UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

From Cities to Series: Complex Networks and Deep Learning for Improved Spatial and Temporal Analytics

Gabriel Spadon de Souza

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Gabriel Spadon de Souza

From Cities to Series: Complex Networks and Deep Learning for Improved Spatial and Temporal Analytics

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. José Fernando Rodrigues Júnior

USP – São Carlos
August 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

```
        Spadon de Souza, Gabriel
S732f      From Cities to Series: Complex Networks and Deep
        Learning for Improved Spatial and Temporal
        Analytics / Gabriel Spadon de Souza; orientador
        José Fernado Rodrigues Júnior. -- São Carlos, 2021.
           171 p.


           Tese (Doutorado - Programa de Pós-Graduação em
        Ciências de Computação e Matemática Computacional) --
         Instituto de Ciências Matemáticas e de Computação,
        Universidade de São Paulo, 2021.


           1. Network Science. 2. Artificial Intelligence.
        3. Urban Systems. 4. Time Series. I. Rodrigues
        Júnior, José Fernado, orient. II. Título.
```

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Gabriel Spadon de Souza

Das Cidades às Séries: Redes Complexas e Aprendizado Profundo para Aprimorar Análises Espaciais e Temporais

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. José Fernando Rodrigues Júnior

USP – São Carlos
Agosto de 2021

To my beloved wife, Patricia Matsumoto.

ACKNOWLEDGEMENTS

Aos meus pais, Aéliton e Vera, agredeço o apoio incondicional que me motivou a buscar e conquistar desafios cada vez maiores; a eles devo não só a vida, mas o respeito e carinho por estarem presentes em todos os desafios que já enfrentei, e por outros que ainda estão por vir. À minha irmã, Julia, agradeço a nossa parceria fraternal, que do seu jeito, me fez observar o mundo e os desafios nele presentes com outras perspectivas que hoje fazem de mim quem sou. À minha esposa e parceira de vida, Patricia, que caminhou comigo durante toda minha jornada acadêmica, que me provocou a superar cada obstáculo que encontrei, que me ensinou muito do que sei hoje e que, diariamente, me encoraja a seguir meus sonhos. Aos meus avôs e avós, que em sua simplicidade e ternura comemoram minhas vitórias e acalentaram meu coração em tempos difíceis. Ao meu orientador, José Fernando, que com sabedoria e paciência me orientou durante o mestrado e doutorado, me proporcionando oportunidades acadêmicas e profissionais antes, por mim, inimagináveis, pautando meu desenvolvimento não só como pesquisador, mas também como pessoa. Aos meus mestres, professores da Faculdade de Ciência e Tecnologia (FCT) da Universidade Estadual Paulista (UNESP) e do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP); ser professor e pesquisador em meio a tantos desafios que a educação de nosso país enfrenta é compromisso com o sonho de um futuro melhor, sendo assim, a todos vocês, meu mais profundo respeito. Aos meus amigos do Grupo de Base de Dados e Imagens (GBdI) pelos bons momentos que passamos juntos nos últimos anos, os quais foram de grande importância para motivar o meu trabalho e a mim mesmo. Bem como agradeço aos meus amigos da minha cidade natal que me proporcionaram experiências de vida incríveis em anos marcantes de amizade. Que a vida nos reserve reencontros calorosos! Por fim, agradeço às agências de fomento que tornaram meu doutoramento possível. ⋆ *O presente trabalho foi realizado com apoio: da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001; do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) – Processo 167967/2017-7; e da, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) – Processos 2016/17078-0, 2017/08376-0 e 2019/04461-9.*

"We are just two lost souls
Swimming in a fish bowl
Year after year (. . .)"
Pink Floyd, 1975

RESUMO

SPADON, G. **Das Cidades às Séries: Redes Complexas e Aprendizado Profundo para Aprimorar Análises Espaciais e Temporais**. 2021. 171 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

A relação entre diferentes entidades de um conjunto de dados é uma propriedade passível de ser representada por um grafo, os quais são conjuntos estruturados formados por entidades (*i.e.*, vértices) e relacionamentos (*i.e.*, arestas). Por muitas vezes grafos foram utilizados para responder questionamentos sobre a interação entre entidades do mundo real pela análise de seus vértices e arestas (*i.e.*, topologia do grafo). As redes complexas, por outro lado, ficaram conhecidas por serem grafos de topologia não trivial. Entre suas aplicações, destaca-se a representação de fenômenos humanos como a urbanização de cidades, o movimento migratório de populações, e a propagação de pandemias. A teoria dos grafos e a ciência de redes, os campos de pesquisa que regem o estudo de grafos e redes complexas, têm sido explorados com sinergia no âmbito da inteligência artificial, no qual transpõe-se a análise da interação entre diferentes entidades para o processo interno de aprendizado computacional dos algoritmos. Neste sentido, a presente tese introduz um ferramental de redes complexas juntamente com técnicas de aprendizado supervisionado de classificação e regressão de modo a contribuir com o entendimento de fenômenos humanos inerentes às malhas viárias, migrações pendulares, e progressões pandêmicas por meio de modelagem e análise computacional. Entre os resultados alcançados, estão: (i) técnicas de identificação de falhas de planejamento urbano ao mesmo tempo em que se auxilia na análise da topologia da rede complexa para diferenciar os vértices mais influentes; (ii) uma metodologia de análise e predição de links em redes complexas no âmbito de mobilidade humana entre cidades por meio de aprendizado de máquina; e, (iii) uma nova arquitetura de rede neural capaz de modelar processos dinâmicos observados em dados variantes no espaço e no tempo, com aplicações de alcance a diferentes domínios. Tais resultados reiteram o potencial dos grafos e das redes complexas na solução de problemas conectados à análise de diferentes fenômenos humanos, bem como a previsão de seus processos evolutivos no espaço e no tempo, quando utilizados conjuntamente com os algoritmos de aprendizado computacional provenientes da inteligência artificial.

Palavras-chave: Ciência de Redes, Inteligência Artificial, Sistemas Urbanos, Séries Temporais.

ABSTRACT

SPADON, G. **From Cities to Series: Complex Networks and Deep Learning for Improved Spatial and Temporal Analytics**. 2021. 171 p. Thesis (Doctorate Candidate in Science) – Institute of Mathematics and Computer Sciences, University of São Paulo, São Carlos – SP, 2021.

The relationship between different entities is a property that can be represented as a graph, structured sets formed by entities (*i.e.*, vertices) and relationships (*i.e.*, edges). Graphs have often been used to answer questions about the interaction between entities from the real world by analyzing their vertices and edges (*i.e.*, the graph's topology). On the other hand, complex networks are known to be graphs of non-trivial topology, capable of representing human phenomena such as cities' urbanization, peoples' movement, and migration, besides epidemic processes. However, graph theory and network science, the research fields that oversee the study of graphs and complex networks, have also been traversed in the realm of artificial intelligence, in which the analysis of the interaction between different entities is transposed to the internal learning process of algorithms. In this sense, this thesis introduces complex networks and supervised learning (classification and regression) techniques to improve understanding of human phenomena inherent to street networks, pendular migration, and pandemics progression through computational analysis and modeling. Accordingly, we contribute with: (i) techniques for identifying inconsistencies in the urban plan while tracking the most influential vertices; (ii) a methodology for analyzing and predicting links in the scope of human mobility between cities through machine learning algorithms; and (iii) a new neural network architecture capable of modeling dynamic processes observed in spatial and temporal data with applications on different domains. These results reiterate the potential of graphs and complex networks in solving problems related to analyzing human phenomena and modeling their evolutive processes across space and time when used together with articial intelligence learning algorithms.

Keywords: Network Science, Artificial Intelligence, Urban Systems, Time Series.

CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Context & Problem

The interaction between entities of different natures is a fundamental property of living systems and a topic that has been studied for centuries under the light of several methods. In the 18th century, for instance, graph theory, a discrete mathematics topic, established the graph as the structured amounting to a set that represents properties from real-world complex systems. That was due to graphs being used for solving the *Seven Bridges of Königsberg* problem [Euler 1741]. Nowadays, in the 21st century, graphs are still as important as in the past, but with the potential to leverage technology to solve more complex and challenging problems. Recent literature goes beyond traditional graph theory and nominates network science [Barabási 2016] as the field dedicated to studying complex networks, which are graphs with unique topological properties [Kim and Wilhelm 2008] considered more sophisticated than traditional ones from discrete mathematics.

According to Boccaletti *et al.* 2006, complex networks represent entity-interaction systems, such as networks of product recommendations, countries cooperation, neuronal connections, subway lines, street meshes, and others. As mathematical models, these networks stand out due to their algebraic properties and computing potential [Barabási 2016], with analytic applicability to brace cognitive processes of knowledge discovery and decision-making. Properties that describe several complex networks are, for instance, the power law of the degree distribution [Barabási and Albert 1999], a high clustering coefficient together with a low diameter in small-world networks [Watts and Strogatz 1998], modular structures [Newman 2006], and network assortativity mixing [Newman 2002].

For example, in the case of street meshes, complex networks[1] can describe factors related to individuals' displacement [González, Hidalgo and Barabási 2009], location-

[1] Hereinafter we use *complex network* (or *network* for short) and *graph* indistinctly.

allocation of facilities [Zhang *et al.* 2016], improvement of tasks related to transport [Scellato *et al.* 2010], and the study of human movement dynamics [Kang *et al.* 2013]. This potential is amplified when the network is provided with additional information about the vertices and edges (*i.e.*, weights). Through specialized structures, such as weighted graphs, the related literature has centered on improving the urban design [Goh *et al.* 2016], cities comprehension [Cui and Han 2015], and human behavior modeling [Pan *et al.* 2013]; mainly in the face of heavy traffic [Iacus *et al.* 2020], epidemics [Kraemer *et al.* 2020], and criminality [Spadon *et al.* 2017, Spadon *et al.* 2018]. These problems reveal the intricacies underneath the human behavior [Song *et al.* 2010, Souza 2017], demonstrating why these phenomena are challenging to understand when one looks from a single perspective.

Graphs are ubiquitous in the real world, but the knowledge represented through them is not yet definitive. Approaches that combine graph theory (and network science) with artificial neural networks (*i.e.*, deep learning [LeCun, Bengio and Hinton 2015]) have demonstrated promising results by solving problems in a wide range of domains [Zhang, Cui and Zhu 2020]. Combining computing techniques and graphs makes it possible to reveal characteristics of interest that are not obvious for human inspections based on reading. Consequently, research on how to utilize those topics together has attracted considerable attention. This is because the networks may be wide (*i.e.*, high number of vertices), intricate (*i.e.*, high number of edges), holding non-trivial patterns and particularities whose observation depends on algorithms. Although end-to-end graph processing with deep learning is still in the early stages of discussion, research on graph-inspired deep learning is currently under the spotlight [Hu *et al.* 2021, Ling *et al.* 2020, Cheng *et al.* 2020, Qiu *et al.* 2019]. Despite being unable to picture a complex system, such as a city, in the way complex networks do, graph-inspired models bring to artificial intelligence algorithms the ability to analyze the topology of a dataset by navigating inside the neighborhood of a graph represented as adjacency matrices through linear algebra operations. Such techniques have already shown outstanding potential in modeling and forecasting dynamic processes on several complex systems [Spadon *et al.* 2021, Oishi *et al.* 2021].

Accordingly, in the context of data science, this thesis presents results derived from classic graph methods to cutting-edge graph-inspired deep-learning techniques. We employ statistics, machine learning, and artificial neural networks to improve the organizational understanding, and to enhance the modeling and analysis of human phenomena inherent to street networks, pendular migration, and pandemics progression, achieving, as a result: (i) a technique capable of tracking urban inconsistencies in street networks – *i.e.*, vertices that suffer from lack of mobility from/to points of interest – using distance functions and graph centrality metrics; (ii) a way to employ machine learning algorithms for link prediction tasks while modeling the dynamics of human mobility and migration between pairs of cities; and (iii) a graph-inspired deep-learning layer and neural network architecture for modeling spatial and temporal dynamic processes over differing granularities.

1.2 Motivation & Rationale

Subsequently, we introduce the motivation and the literature-related reasons that led us to our three research investigations, which have graphs as a modeling formalism.

1.2.1 Network Science

Over the years, network science has asymptotically evolved, resulting in a range of innovations in street meshes modeling and forecasting through complex networks. These networks improve the understanding of entity-interaction systems by analyzing topological properties like shortest paths [Maruhashi *et al.* 2012, Galbrun, Pelechrinis and Terzi 2016] and centrality indicators [Porta *et al.* 2011, Gil 2016]. They were also used in studies concerning cities' geometric aspects [Cardillo *et al.* 2006, Corcoran *et al.* 2015] and spatial organization [Barthélemy and Flammini 2008, Zhong *et al.* 2014], some of which leveraged features from the vertices and edges [Scripps *et al.* 2010, Myronenko, Wenger and Atmazhov 2016]. Some authors focused on the analytical reasoning over cities represented as graphs [Crucitti, Latora and Porta 2006, Costa *et al.* 2010], while others approached the support of urban design planning [Porta *et al.* 2009, Strano *et al.* 2012], apart from the advancements made by analyzing points of interest on street meshes [Song, Merlin and Rodriguez 2013, Li and Parrott 2016]. These studies and others with related contributions provide for the economic, social, political, and environmental spheres of city planning.

From another research perspective, multiple graph metrics have been used to understand the structure of cities [Porta, Crucitti and Latora 2006, Porta, Crucitti and Latora 2006, Travençolo and da F. Costa 2008, Viana and da Fontoura Costa 2011], model street traffic [Zamith *et al.* 2015], and plan tourism routes [Spadon and Rodrigues-Jr 2018]. Some research examined the role of centrality in urban agglomerations [Xiao, Webster and Orford 2014, Hillier 2007], while others focused on mathematical properties found in the network topology [Sarkar 2015, Spadon *et al.* 2018]. In such a case, topological properties refer to shortest paths [Maruhashi *et al.* 2012, Galbrun, Pelechrinis and Terzi 2016, Gil 2016] analyzed from the viewpoint of geometry [Corcoran *et al.* 2015], spatial distribution [Barthélemy and Flammini 2008, Zhong *et al.* 2014], and external data [Davies and Johnson 2015, Sopan, Rey and Shneiderman 2013, Shiode and Shiode 2013].

Studies in the field of street network analysis have roots in urban morphology [Goh *et al.* 2016], which is a research field that studies the shape of cities, describing geometric aspects that impact the population [Barthelemy 2017], foundations of urban evolution [Dibble *et al.* 2017], and the displacement in cities [Boeing 2019], to name a few. These studies are mainly based on quantitative methods and metrics for analyzing urban forms [Boeing 2017]. They have been consistently employed for solving mobility and access-related issues, such as street pattern modeling, facility location-allocation, and inconsistency tracking (*i.e.*, anomaly detection problems). However, a fundamental and yet unsolved problem has to do with the characterization of cities based on their form [Strano *et al.* 2013]. This is because cities are not naturally divided into groups due to historical, economic, and social traits impacting their evolution and increasing their uniqueness [Bettencourt *et al.* 2007]. These traits make it challenging to define an accurate model capable of distinguishing two or more cities, such that studies in this field have been leveraging from similarity analysis [Spadon, Gimenes and Rodrigues 2018, Boeing 2018].

1.2.2 Artificial Intelligence

Beyond the complex network topology, the literature describes the modeling and forecasting of human mobility between places using techniques ranging from traditional mathematics to artificial intelligence. The interest in doing so comes from humans being bounded to move daily [Ravenstein 1889, Stouffer 1940], and understanding human mobility is vital to explain the processes related to human movement [Barbosa *et al.* 2018], which impact the community (nearby people) and environment (cities and nature). The analysis of human mobility networks can assist in urban planning activities [Krueckeberg and Silvers 1974, Batty 2008, Batty and Longley 1994, Benenson and Torrens 2004], modeling the evolution and spread of epidemics [Eubank *et al.* 2004, Colizza *et al.* 2006, Balcan *et al.* 2009], and even forecasting catastrophic events [Carter *et al.* 2002].

The literature on deep learning using graphs is still in its inception, especially in domain-oriented cases. Recently there has been an increased interest in graph-based representation learning through Graph Neural Networks (GNNs) [Li *et al.* 2016, Hamilton, Ying and Leskovec 2017, Kipf and Welling 2017, Velickovic *et al.* 2017, Xu *et al.* 2018], which follows a message passing scheme through a recursive neighborhood aggregation of feature vectors [Xu *et al.* 2018, Gilmer *et al.* 2017]. Along these lines, several variant architectures with mixed neighborhood aggregation and graph-level pooling schemes were proposed [Scarselli *et al.* 2009, Defferrard, Bresson and Vandergheynst 2016, Duvenaud *et al.* 2015, Hamilton, Ying and Leskovec 2017, Kearnes *et al.* 2016, Verma and Zhang 2018, Ying *et al.* 2018, Zhang *et al.* 2018]. Although such studies show that the GNN and variants can achieve state-of-the-art performance in many tasks (*e.g.*, topological classification and link prediction), their architecture is known to be based on intuition, heuristics, and trial-and-error [Xu *et al.* 2019]. Either way, GNNs have been used for multi-agent communication learning in artificial intelligence systems [Sukhbaatar, Szlam and Fergus 2016], physical reasoning using objects-through-vertex representation [Battaglia *et al.* 2016] while leveraging attention to weigh different interactions [Hoshen 2017], and neural networks augmentation for question answering problems [Santoro *et al.* 2017].

In conclusion, the related works lack robustness concerning the analytical procedures and methodologies. Often, they are based on analyzing small parts of cities and not the whole complex network, presenting limitations in evaluating the network topology. Moreover, few of them benefit from multiple graph metrics. Such metrics can define feature vectors suitable for clustering analysis and multidimensional projection, approaches weakly explored in the street network literature. There is still much to be explored regarding computational intelligence related to human phenomena (*i.e.*, mobility and pandemics) as the literature is embryonic, with open flanks to introduce more sophisticated methodologies. Accordingly, this thesis contributes to the state-of-the-art in both complex networks and artificial intelligence by conducting a multi-faceted investigation, including topics that range from the statistical inference over urban structures, the topological analysis of cities in the form of complex networks, up to the modeling of dynamic processes related to human mobility and pandemics progression on georeferenced data. Hence, our investigations and results benefit from an interchangeable underlying structure, the graph.

1.3 Questions & Hypotheses

Concerning the large volume of urban data available from digital maps, urban indicators contributed by governmental censuses, and empirical observations of collective phenomena on georeferenced data, this thesis is motivated by the following questions: **Q1** – *"How is it possible to detect urban inconsistencies through topological features from complex networks to highlight the urban design problems of cities?"*; **Q2** – *"In which way is it possible to use urban indicators from governmental censuses as input of learning algorithms to forecast the human movement observed in pendular migrations?"*; and, **Q3** – *"How can one make use of multiple empirical observations from the real world (e.g., cities, states, and countries) to shape the progression of a pandemic across time and space?"*.

The overall outcome of the thesis, driven by the previous questions, was produced by conducting three different but interconnected research fronts, each handling a particular set of materials and methods. The first front **F1** refers to question **Q1** and is tackled in Chapter 2; the second one **F2** refers to question **Q2** and is discussed in Chapter 3; while the last one **F3** covers question **Q3**, whose investigation is reported in Chapter 4. In the following, we provide details about the research fronts and their initial hypotheses:

F1. We propose using distance and centrality metrics to track inconsistencies in street networks by assuming that cities should provide the shortest routes among different destinations of interest (*e.g.*, hospitals and schools); when considering a vertex that is closer to a target by geodesic distance and closer to another by shortest path distance, the vertex is regarded as an urban inconsistency. The joint of urban inconsistencies and centrality metrics can reveal the importance of elements that are deficient within a city. We departed from the following hypothesis: *using proper* *graph modeling over a digital map makes it possible to design an algorithmic solution based on set theory to detect spatial inconsistencies in the street mesh of a city.*

F2. We propose the use of machine learning to model mobility and migration patterns inherent to urban indicators. The flow of commuters, also referred to as pendular migration, describes the daily movement of people between home and work in dierent cities. To this end, machine learning can predict discrete or continuous values of unknown elements by learning from the data, contributing to the collective human-behavior characterization. Applications along these lines can assist the design of cities, reduce costs related to the planning of highways, and provide an estimation of people's movement in between governmental censuses. We departed from the following hypothesis: *by using official census information, the use of advanced machine learning algorithms can provide a more accurate – as compared to analytical formulations – estimate of the flow of commuters between pairs of cities.*

F3. We propose an artificial intelligence framework for tackling higher-dimensional time-series forecasting problems, which involves looking at multiple time-series and their related variables simultaneously, leveraging from temporal patterns existing between different yet related data. An example where such a paradigm applies is in the pandemics progression modeling of the Coronavirus Pandemic. This is because, despite progressing in different moments and locations, the underlying mechanisms behind the pandemic are supposed to follow similar yet non-identical patterns. Looking individually at the development of the pandemic in each country enables us to describe the problem in terms of multiple variables, like the number of confirmed cases, recovered people, and deaths. However, when looking at all the countries at once, each country becomes a multivariate sample of a broader problem resulting in a multiple multivariate time-series forecasting problem. We departed from the following hypothesis: *by exploring the fact that time series are dependent not only on their inner variables but also on outer variables originated from other concurrent time series sharing the same timestream, it is possible to advance the state-of-the-art in terms of Artificial Neural Network architectures for spatial and temporal problems.*

In summary, this thesis is grounded on graph-based theory to model phenomena arising from different domains related to the human nature. It utilizes data science techniques such as set theory, algorithmic classification and regression, and artificial neural networks to produce analytical artifacts that improve the comprehension of the underlying problems. Our results potentially support decision-making, predicting values, and forecasting events in real-world circumstances as demonstrated in extensive experiments. Comprising three investigation lines, the spinal cord of this work lies in the use of a standard base – *an ensemble framed from the use of complex networks, data science, and* *computational intelligence*. The domains, in turn, are all related to human phenomena that emerge from social interactions of different granularities, such as the ones observed between individuals, societies, and cities. This general background, which granted a convergent thematic, was due to the original more general hypothesis, compiled as follows:

General Hypothesis

The analytic processing by means of complex networks and graph metrics combined with artificial intelligence techniques from the computational intelligence realm can expand the comprehension and, consequently, the capacity of modeling and forecasting different human phenomena, providing us with information for acting on the network topology (i.e., street networks from maps), dynamics (i.e., pendular migration between cities), and inner processes (i.e., pandemics progression over time).

1.4 Summary of Contributions

As our first contribution, we advanced with a set of distance-based pattern-discovery algorithmic instruments to detect vertices that lack access from/to points of interest in a given city (see Chapter 2). We presented a proof of concept and case studies, all of which indicate that our methodology suggests better placements for points of interest at the same time that it improves access to the majority of the vertices of a city by reducing the number of inconsistencies. Our contributions are in the definition of a concept based on intrinsic problems to urban structures; in two algorithms devised to track and reduce inconsistent vertices in complex networks; and, finally, in a case study, in which we show how our toolset and algorithms can aid urban planners. Overall, these contributions introduce a systematic manner to treat a recurrent problem of broad interest in cities.

The second contribution comes from devising a model capable of reconstructing the inter-city commuters network and accurately reproducing the flow of people traveling from residence to work in a different city (and vice-versa) using machine learning algorithms and urban indicators (see Chapter 3). Our results show that gradient-based algorithms can reconstruct the commuters network with 90.4% of accuracy and describe 77.6% of the variance observed in the flow of people between cities. We further identify the essential features to rebuild the network and the indicators that attract workers to commute more intensely. Distance plays a vital role in commuting, but other indicators such as Gross Domestic Product and unemployment rate are also driving forces for people to commute. Because link prediction and network reconstruction are significant challenges in network science, these results shed new light on the role of urban indicators on commute patterns.

This thesis's third contribution is based on an artificial intelligence architecture for tackling higher-dimensional time-series forecasting problems, which involves looking at multiple time-series and all their related variables concurrently, leveraging from temporal patterns existing between different yet related data (see Chapter 4). Thereby, we benefit from multiple multivariate time series to propose a new neural network and layer architecture, which are named, respectively, Recurrent Graph Evolution Neural Network (ReGENN) and Graph Soft Evolution (GSE) layer. The techniques we propose are capable of working on higher-dimension tensors from input to output, finding non-linear relationships between different time series, and making forecasting across time for multiple multivariate time-series. The results we discuss outperformed both statistical and ensemble-learning approaches, showing an improvement of 64.87% over the competing algorithms in the task of epidemiology modeling on the SARS-CoV-2 dataset of the renowned John Hopkins University for 188 countries simultaneously. Besides, we also outperformed the task of climate forecasting on the Brazilian Weather dataset for 253 sensors by at least 11.96% and patient monitoring on Intensive Care Units (ICUs) on the PhysioNet dataset by 7.33% for 11,988 patients. Nonetheless, our results are not limited to achieving state-of-the-art marks in time-aware data modeling but also in interpreting and bringing to light the relationships found by the neural network, which we do by using similarity analysis over the hidden weights of the representation learning layers of the neural network.

1.5 Document Organization

In order to describe our results in detail, this thesis is organized as a *Collection of Articles* divided into five chapters, in which, besides the *Introduction* and *Conclusion* chapters, the intermediary ones reproduce selected articles resulting from the thesis project. The document organization and the articles contained in each chapter is as follows:

◇ Chapter 1 introduces the problem, context, and motivation underneath the thesis;

◇ Chapter 2 reviews the urban design and organization problem with complex networks;

⋆ It covers the following article:

Spadon G., Brandoli B., Eler D. M., and Rodrigues-Jr J. F., *Detecting Multi-Scale Distance-Based Inconsistencies in Cities through Complex-Networks.* Journal of Computational Science. Elsevier, 2018.

□ Further published articles on the same topic are available in Appendix A.

◇ Chapter 3 addresses the human mobility-related problem using machine learning;

⋆ It covers the following article:

Spadon G., Carvalho A. C. P. L. F., Rodrigues-Jr J. F., and Alves L. G. A., *Reconstructing Commuters Network using Machine Learning and Urban Indicators.*

Scientific Reports. Springer Nature, 2019.

☐ Supplementary material is available in Appendix B.

◇ Chapter 4 outlines the modeling and forecasting of spatial and temporal data; and,

⋆ It covers the following article:

Spadon G., Hong S., Brandoli B., Matwin S., Rodrigues-Jr J. F., and Sun J., *Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning.* Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2021.

☐ Supplementary material is available in Appendix B.

◇ Chapter 5 presents the conclusions and final remarks, including a discussion on the practical implications of the thesis contributions focused on the general public understanding.

◇ Appendix A and B present, respectively, further publications not included as chapters and supplementary material required for the understanding of the thesis contributions.

CHAPTER

2

STREET NETWORK ANALYSIS

In this chapter, we reproduce the following article:

Spadon *et al.* 2018, *Detecting Multi-Scale Distance-Based Inconsistencies in Cities through Complex-Networks.* Journal of Computational Science. Elsevier.

The article consists of a distance-based pattern-discovery algorithmic instrument to improve urban structures by detecting vertices that lack access from/to points of interest in a city. The study assumes that the network has streets that render the shortest distance between places. In this regard, our toolset uses two distance functions to track vertices that do not provide shortest-distance routes to vertices of interest. Vertices that fail to provide minimum-length routes are considered inconsistent vertices and evidence of anomalies in the city structure. Accordingly, we introduce a greedy algorithm that can recommend improvements to cities' structures by suggesting where to place points of interest, contributing with a thorough process that ranges from mathematical modeling to algorithmic innovation. These contributions are linked to a sequence of papers also resulting from this thesis but included in Appendix A and listed as follows:

Spadon *et al.* 2018, *A distance-based toolset to track inconsistent urban structures through complex-networks*[1]. In International Conference on Computational Science. Springer, Cham.

Spadon, Gimenes and Rodrigues 2018, *Topological street-network characterization through feature-vector and cluster analysis.* In International Conference on Computational Science. Springer, Cham.

Spadon and Rodrigues-Jr 2018, *Computer-assisted city touring for explorers.* In Workshop on Big Social Data and Urban Computing (BiDU) co-located with the 44th International Conference on Very Large Data Bases (VLDB). CEUR-WS.

1 Awarded paper – Invited for publication at the *Journal of Computational Science.*

The previous listing of papers are considered to be *future works* from a mutual paper:

> Spadon, Gimenes and Rodrigues-Jr 2017, *Identifying Urban Inconsistencies via Street Networks*[2]. Procedia Computer Science 108. Elsevier.

Among those related to the thesis, we report an early formulation for the *Urban Inconsistencies* and present a systematic literature review together with a detailed methodology for optimizing tourist routes in cities, besides providing empirical evidence that: (A) the network topology is a source that can reveal groups of cities with similar characteristics, potentially exposing disparities; (B) although cities may share administrative boundaries with others, they cluster with cities with no apparent similarity; and, (C) there is a correlation between urban and territorial indicators with the features extracted from the street-network topology of cities. Those findings allowed us to better comprehend how cities are organized within the geographical extent of their boundaries.

We reproduce the article from Spadon *et al.* 2018 under the *Rights and Permissions* below on the following pages. Besides the introduction in Section 1, the article is organized as follows: Section 2 reviews closer-related works found in the literature; Section 3 discusses the algebraic formulation of the proposed method; Section 4 presents the algorithmic concepts, solutions, and analysis; Section 5 exposes the results about the applicability of the proposed toolset; and, Section 6 exhibits the conclusions and remarks.

[2] Results from the Masters' dissertation.

Journal of Computational Science 30 (2019) 209–222

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Detecting multi-scale distance-based inconsistencies in cities through complex-networks☆

Gabriel Spadon[a,∗,1], Bruno Brandoli[b], Danilo M. Eler[c], Jose F. Rodrigues-Jr[a]

[a] University of Sao Paulo, Sao Carlos, SP, Brazil
[b] Federal University of Mato Grosso do Sul, Ponta Pora, MS, Brazil
[c] Sao Paulo State University, Presidente Prudente, SP, Brazil

ARTICLE INFO

Article history:
Received 10 September 2018
Received in revised form 5 December 2018
Accepted 27 December 2018
Available online 30 December 2018

2010 MSC:
05C12
90C35
05C85

Keywords:
Complex network
Network analysis
Urban structure

ABSTRACT

We propose an algebraic tool-set and related algorithms to track access problems not obviously observed in urban environments represented as street mashes. Our tool-set assumes that points of interest must be promptly accessible within the paths of a street network. Over that assumption, we introduce formalisms and computational tools to detect and evaluate access-related problems. The output of our methods, in the form of inconsistent nodes found in a given city, has the potential to assist the decision-making process regarding the positioning of resources, building of new services, and outlining the initial design of a city.

1. Introduction

Complex networks are capable of modeling complex systems from social networks to railway systems [1], and also, they can represent cities when linking the network topology with the geographical position of the network nodes. All these systems, when in the form of a complex network, describe the information exchanging through the relationships of their entities — edges connecting nodes. The cities' complex-network can reveal features capable of describing urban problems, which are meaningful indicators for city planners [2]. Such features can reveal, for instance, sites where social activities are more intense, regions where facilities should be placed, and locations that lack street access.

In this article, we contribute with an algebraic tool-set and related algorithms to track multi-scale distance-based inconsistencies by analyzing the complex-network topology of a city using multiple perspectives of displacement. The related literature still lack methods to analyze and enhance the structure, mobility, and street access of cities [2–5]. As such, our results have implications in all the previous points, supporting a finer street planning by improving mobility indicators and providing better city's structural evaluation.

The core assumption of this study is that the network is supposed to provide streets that render the shortest distance between places considering both pedestrian and automobile displacements. In this regard, our tool-set uses three distance-functions while varying the direction of the complex-network edges to track nodes that do not provide shortest-distance routes *to*/*from* nodes of some interest. Nodes that fail in providing minimum-length routes are considered to be inconsistencies, which are evidence of structural problems in the street mesh.

Although we focus on pedestrians and automobiles the methodology is suitable to any kind of on-surface transportation (*e.g.*, bicycles and buses, to name a few) as they are obligated to follow the block-based itineraries intrinsic to street networks. Underground transportation, in turn, defines a totally different problem, since it can explore shortest line itineraries that are not based on street-based constraints. The relevance of our proposal comes from the fact that just 178 cities in 56 countries [6] can count on underground transportation means.

☆ This article is an extended version of our conference paper (published under DOI: 10.1007/978-3-319-93698-7_22 in the ICCS 2018) invited to this JoCS' special issue.
∗ Corresponding author.
E-mail addresses: spadon@usp.br (G. Spadon), bruno.brandoli@ufms.br (B. Brandoli), daniloeler@fct.unesp.br (D.M. Eler), junio@usp.br (J.F. Rodrigues-Jr).
1 http://www.spadon.com.br.

https://doi.org/10.1016/j.jocs.2018.12.015
1877-7503/© 2018 Elsevier B.V. All rights reserved.

Accordingly, this article presents an extended version of Spadon et al. [7], advancing with a more robust algebraic tool-set to track inconsistent urban structures through complex networks. We have extended our previous study by considering different types of displacement from pedestrian and automobiles both regarding the street-mesh of a city; by further discussing the points of interest in the Brazilian city of Sao Carlos; and, by enhancing the definition of inconsistencies in a continuous rather than a categorical manner. Along these lines, in face of inconsistencies of a given city shaped as a complex network, our study is paved into two hypotheses: (A) the network lacks a more appropriate mesh; or, instead, (B) the city lacks its points of interest placed in more appropriate locations. The first one indicates the need for new points of interest to distribute their load. Contrarily, the second one indicates the need for relocating points of interest because the topology of the terrain cannot bear new streets. Along the investigation of such a hypothesis, we consider the dynamic particularities of points of interest regarding how citizens have street-based access to them. As a result, we explore the points of interest according to four categories (*i.e.*, inward, outward, absolute, and walking-based) of street-based urban planning problems (*i.e.*, inconsistencies). Furthermore, notice that this work relates to the concept of street accessibility, which is one of the many variables that influence the flow of a city, and that is a much broader problem.

In order to present our contributions, this article is organized as follows: Section 2 reviews related works found in the literature; Section 3 discusses the algebraic formulation of our proposal; Section 4 presents the algorithmic concepts and solutions; Section 5 exposes the results about the applicability of the proposed tool-set; and, Section 6 exhibits the conclusions and final remarks.

2. Related work

The motivation of our proposal is the desire to improve the knowledge about the mobility of cities regarding its points of interest. To this end, we use spatial properties found in street networks to analyze the access and location of points of interest, supporting decision-making activities that might improve the design of urban structures. Along these lines, other authors pursued similar purposes with a related motivation, as discussed in the following.

Porta et al. [8] performed an investigation about centrality in cities, introducing *Multiple Centrality Assessment* (MCA) as a methodology defined over 1-square-mile street sub-regions, providing a different spatial perspective based on graph metrics. We refrain from using the same approach because the authors do not focus on identifying elements that are critical to urban planning and design. Instead, we define a different goal aiming to characterize an urban-space based on a set of reference nodes (referred to as points of interest) of single cities, and not comparatively among cities.

Following the ideas introduced by the MCA methodology, other studies presented a comparative analysis of cities. This is the case of Crucitti et al. [3], who identified common characteristics between self-organized and planned cities. Others [9–11] studied centrality focusing on the characterization of the peculiarities of several cities by employing visualization functionalities.

The MCA methodology uses the 1-square-mile sampling approach that allows for higher scalability over non-optimized algorithms. This approach is limited in the sense that sampling can inspect local characteristics, but it is not able to characterize macro-structural features, which can be found in cities. Differently, our analyses embrace the whole city, and for each point of interest, our tool-set analyzes it in the context of its surroundings and nearby structures.

Following a different approach, the study performed by Strano et al. [12] investigated the network's geometrical attributes using centrality to highlight discrepancies and relationships. In particular, their results point to the fact that cities share architectural similarities due to their quasi-planarity, but also that each instance reveals several unique geometrical proprieties. As another example, Scellato et al. [13] explained how it is possible to extract the backbone of a city using spanning trees based on edge-betweenness and information centrality. Despite the potential of both, none of them was proficient in tracking irregularities in the street meshes, which allows extending the understanding of important routes that affect public services and retails.

In regard to general transportation, the related literature still lacks from discussions on how to improve access to critical nodes of the network mainly when taking into account the inconsistent nodes found in a city. These former studies have focused on metric-analytical methods applied to cities [14], others have approached the support to urban planning and design [15], and there are still those who have researched on transportation and resource planning [16,17].

Costa et al. [14] focused on the effectiveness of the underground systems to facilitate the connection between distant places, while Viana and Costa [18] explored long-range connections to improve the traffic between two points. The first proposal differs from ours as they consider transportation means beyond the city streets, such an approach is not reasonable to ours because our focus is on characterizing features extracted from a street network. In the second one, long-range connections demand single edges that are far larger than what is feasible in street-based networks, what would demand another research approach.

Travençolo and Costa [19], on the other hand, have first defined the concept of accessibility by means of entropy. They stated the *Outward Accessibility* as the diversity of the access of a node concerning all nodes in the network and the *Inward Accessibility* as the frequency of accesses to a node from all the others. As a result, they demonstrate that hypothetical edges can improve the city's mobility. Their proposal differs from ours as they consider the access a global property of the network; meanwhile, we take it as the ability of displacement though minimum-length paths concerning a set of user-given referential nodes.

Huynh et al. [20] investigated the spatial distribution of bus stops in different cities by employing the percolation cluster from statistical physics. The motivation of the authors was to provide information about the transportation found in cities, so to distinguish them into well-planned or organically grown cities. The analysis of transportation regarding bus stops gives us a view of how easily citizens can move around. However, the routes used by buses are not always adequate by means of minimum-length paths; they relate only to route planners, which cannot benefit from our methodology. That is because their goal is to help in the process of locomotion, but not to fulfill the whole displacement process, in such a way that they minimize the distance between passengers' origins and destinations. In another perspective, we analyze distinct sets of points of interest in a city and how they behave in the whole process of locomotion considering displacement from both pedestrians and automobiles.

The urban analysis is not limited to the previous investigations, for instance, when aiming towards the solution of questions related to it, multiple metrics have been adopted to understand and analyze the structural conditions of the cities [21], the intense traffic of vehicles [22,23], and the collective behavior [24,25]. More specifically, some studies [26–29] analyzed the centrality of the elements acquired from the city representation, and also, others focused on mathematical and geometrical properties originated from its topology; this is the case of shortest paths [30–32] and centrality [33], that can be analyzed from the perspective of geometry [34], spatial disposition [35,36], and either from outside data that can be related together with the network [37–39].

3. Algebraic formulation

3.1. Preliminaries

Along the text, we refer to a complex network as a distance-weighted directed graph $G = \{V, E\}$, which is composed by a set V of $|V|$ nodes and a set E of $|E|$ edges. To model a city as a complex network, we considered streets as the edges and their crossings as the nodes, preserving the city topology. An edge $e \in E$ is an ordered pair $\langle i, j\rangle$, in which $i \in V$ is named *source*, and $j \in V$ is named *target*, $i \neq j$. Each node $i \in V$ has two properties $\{\mathcal{L}^i_{at}, \mathcal{L}^i_{on}\}$ that correspond to their coordinates — $\mathcal{L}^i_{at}$ is the latitude and $\mathcal{L}^i_{on}$ the longitude. Based on such coordinates, we conferred to the edges in E a floating-point weight that refers to the great-circle (or *inline*) distance between their *source* and *target*. The great-circle distance is the shortest distance between two points on the surface of a sphere; in our case, the sphere refers to planet Earth, as detailed in Section 3.3. The gain from using the great-circle distance is related to the accuracy in approximating the real-world distance. There is no loss of performance in using such an equation because we compute the length of the streets only once per city. Furthermore, its asymptotic complexity is sublinear.

3.2. Problem constraints

The algebraic tool-set we devised tracks nodes that fail to provide street access with regard to a set of points of interest, such nodes demand access to pedestrians and automobiles. In this sense, it is known that both pedestrians and automobiles must obey traffic rules and regulations. Vehicles must follow the streets' direction while pedestrians can walk back and forth in any direction.

The idea then is to analyze a city from multiple points of view considering that *closer nodes must be walkable* and *distant nodes must be drivable*. It is noteworthy that we use the terms walkable (referring to pedestrians) and drivable (referring to automobile vehicles) when referring to nodes that can render minimum-length paths *to/from* points of interest in a given city.

Yong and Diez-Roux [40] stated that the mean distance that pedestrians usually walk to reach some place is about 0.7 miles ($\approx$1.13 km), in such a case, we are assuming that any point of interest up to this threshold must be walkable, and the ones beyond the threshold must be drivable. Along these lines, our tool-set begins tracking a set $P \subset V$ of points of interest; the idea, then, is to determine three sets of nodes that surrounds a point of interest $p \in P$. These sets are the *perimeter set of p*, the *driving set of p*, and the *walking set of p*.

3.3. Grouping nodes in the surroundings of the points of interest

The first set refers to the closest nodes to a point of interest p according to the great-circle (or *inline*) distance, which is called the *perimeter set of p*. The walking threshold (w_t) divides this set into two, such that the first one consists of nodes that are inside the walking range and the other one is formed by nodes that are inside the driving range. Such a threshold is a parameter of our model, we have defined it based on the results of Yong and Diez-Roux [40] (see Section 3.2), but it can be changed to adapt to another scenario based on specific knowledge of urban planners. This means that nodes pertaining to the *walking-perimeter set of p* (W^E_p) must have their great-circle distance to p less than or equal to the threshold w_t and the ones that fail in fitting this constraint are known to be in the *driving-perimeter set of p* (D^E_p), both formalized as follows:

$$W^E_p = \{v \in V | d^E_{vp} \leq w_t, d^E_{vp} < d^E_{v\bar{p}}, \forall p, \bar{p} \in P, p \neq \bar{p}\} \quad (1)$$

$$D^E_p = \{v \in V | d^E_{vp} > w_t, d^E_{vp} < d^E_{v\bar{p}}, \forall p, \bar{p} \in P, p \neq \bar{p}\} \quad (2)$$

where d^E_{ij} is the great-circle distance between i and j in the surface of Earth. The great-circle distance is an undirected distance function $d^E_{ij} \equiv d^E_{ji}$ given by:

$$d^E_{ij} = \mathcal{R} \times \text{arcos}\left(\sin(\mathcal{L}^i_{at})\sin(\mathcal{L}^j_{at}) + \cos(\mathcal{L}^i_{at})\cos(\mathcal{L}^j_{at})\cos(\Delta_{\mathcal{L}^{ij}_{on}})\right) \quad (3)$$

where $\mathcal{L}^i_{at}$ and $\mathcal{L}^j_{at}$ are the latitudes, $\Delta_{\mathcal{L}^{ij}_{on}}$ is the difference between the longitudes $\mathcal{L}^i_{on}$ and $\mathcal{L}^j_{on}$, both of the nodes i and j. Also, $\mathcal{R}$ is the radius of Earth (6378 km), and all values are represented in radians. Given a graph $G = \{V, E\}$ and a set of points of interest P, according to our formulation, a node $v \in V$ pertains to one subset (W^E_p or D^E_p) from the perimeter set of only one $p \in P$, meaning that they are mutually disjoint among their equals and also to the opposite subset:

- $W^E_p \bigcap W^E_{\bar{p}} = \emptyset$ and $D^E_p \bigcap D^E_{\bar{p}} = \emptyset - \forall p \in P, \forall \bar{p} \in P, p \neq \bar{p}$; and,
- $W^E_p \bigcap D^E_{\bar{p}} = \emptyset - \forall p \in P, \forall \bar{p} \in P$.

The following two sets are composed of nodes closest to a point of interest p according to the length of their shortest paths. The *walking set of p* (W^N_p) uses the shortest **undirected** path (d^U_{ij}) as the distance function; while the *driving set of p* (D^N_p) uses the shortest **directed** path (d^N_{ij}) instead. This is because pedestrians can walk in any direction (see Section 3.2). These sets are used to group nodes that are closest to points of interest when moving through streets.

$$W^N_p = \{v \in V | d^E_{vp} \leq w_t, d^U_{vp} < d^U_{v\bar{p}}, \forall p \in P, \forall \bar{p} \in P, p \neq \bar{p}\} \quad (4)$$

$$D^N_p = \{v \in V | d^E_{vp} > w_t, d^N_{vp} < d^N_{v\bar{p}}, \forall p \in P, \forall \bar{p} \in P, p \neq \bar{p}\} \quad (5)$$

We consider a path S between two nodes i and j as an ordered sequence of connected nodes $S^n = \langle v_1, v_2, \ldots, v_{q-1}, v_q\rangle$, in which the consecutive ones are connected through an edge $\langle S^n_m, S^n_{m+1}\rangle \in E$, $\forall m \in [1, |S^n|[$. In turn, the shortest directed path d^N_{ij} consists of minimizing the real-valued weight function $f : E^n \to \mathbb{R}$ that describes the cost of the paths among all the existing paths $\mathbb{S} = \{S^1, S^2, \ldots, S^n\}$ between nodes i and j, such that $\sum_{m=1}^{|S|-1} f(\langle S^n_m, S^n_{m+1}\rangle)$ must be of minimum cost [41]. The cost is related to the weight of the edges given by the straight-line distance between their nodes through Eq. (3) and, along these lines, the directed shortest path distance is defined as follows:

$$d^N_{ij} = \min\left(\sum_{m=1}^{|S|-1} f(\langle S^n_m, S^n_{m+1}\rangle), \forall S^n \in \mathbb{S}\right) \quad (6)$$

Also, d^U_{ij} measures the shortest undirected paths that refer to Eq. (6) but now considering that $\langle S^n_m, S^n_{m+1}\rangle \equiv \langle S^n_{m+1}, S^n_m\rangle$, $\forall m \in [1, |S^n|[$. In other words, the direction between consecutive connected nodes in a path is not restrictive.

We refer to the shortest path length as *direct* and *undirected network distance*, in the sense that one must necessarily (in the best case) move across this path to go from the source node to the target node. Additionally, notice that any node $v \in V$, according to our formulation, pertains to one subset (W^N_p or D^N_p) from the network set of only one $p \in P$, meaning that they are all mutually disjoint among their similar and opposite subsets, formally defined as follows:

- $W_p^N \bigcap W_{\bar{p}}^N = \emptyset$ and $D_p^N \bigcap D_{\bar{p}}^N = \emptyset - \forall p \in P, \forall \bar{p} \in P, p \neq \bar{p}$; and,
- $W_p^N \bigcap D_{\bar{p}}^N = \emptyset - \forall p \in P, \forall \bar{p} \in P$.

The *direct network-distance* ***TO*** *a point of interest* is not necessarily the same as the *direct network-distance* ***FROM*** *a point of interest*, which may result in different driving sets for the same *p*. This specialization is addressed in the following section, where we define the driving set from a point of interest to the nodes in *V* by means of the *reverse driving-set of p*, defined as follows:

$$\bar{D}_p^N = \{v \in V | d_{vp}^E > w_t, d_{pv}^N < d_{\bar{p}v}^N, \forall p \in P, \forall \bar{p} \in P, p \neq \bar{p}\} \qquad (7)$$

3.4. Compartmentalizing inconsistencies using the network direction

Consider different public services of a city as points of interest; such services may have different ways to assist the population, but all of them must require locomotion as a condition for assistance. For instance, in the case of doctors' clinics, it is desired that patients get there efficiently. In turn, police stations require that their police officers efficiently reach the house of the citizens. In the case of schools, the daily routine demands an efficient back-and-forth transit to students; along with other services that can be fitted with this assumption. Notice that, we are referring to efficient paths as the ones with minimum length.

In the first example, there is an implicit displacement from a node *v* to a node *p*; in the second one, the displacement is from the node *p* to the node *v*; and, in the third case, there is a bi-directional displacement between *v* and *p*, in which *v* is an ordinary node and *p* is a specific point-of-interest. Based on the direction of the network edges, those three cases led to the following definitions:

1. *Inward inconsistency:* nodes that are inline-closest to a point of interest, but network-closest (from *v* to *p* — Eq. (5)) to another one:

$$\Psi_p^I = D_p^E - D_p^N \qquad (8)$$

2. *Outward inconsistency:* the same as the previous category, but in the opposite direction of the network-closest set (from *p* to *v* — Eq. (7)):

$$\Psi_p^O = D_p^E - \bar{D}_p^N \qquad (9)$$

3. *Absolute inconsistency:* nodes that are, simultaneously, known to be inward and outward inconsistencies, *i.e.*, nodes in the intersection of sets:

$$\Psi_p^A = \Psi_p^I \cap \Psi_p^O \qquad (10)$$

In cases where the direction of the edges is not taken into account, there will be no minimum-length divergence between paths of a round trip; yet, the inconsistencies can be tracked by calculating the difference between the walking-perimeter set W_p^E and the walking set W_p^N of *p*, resulting in the following definition:

4. *Walking inconsistency:* nodes inside the walking range that are inline-closest to a point of interest, but undirected network-closest to another:

$$\Psi_p^W = W_p^E - W_p^N \qquad (11)$$

3.5. Quantifying inconsistencies in a range

Inconsistencies of a city have been discussed categorically, but further algebra can aid in identifying the severity of a network inconsistency in a continuous, rather than a categorical, manner. The motivation is to assess the inconsistency degree of the nodes and to focus on nodes that are more inconsistent than others, improving the decision-making in the design and planning of urban structures.

Generally speaking, the tool-set for inconsistency tracking relies on the great-circle distance as the ground truth of displacement between a node $v \in V$ and a point of interest $p \in P$. The idea is to use such distance to measure the deviation between the great-circle distance (d_{ij}^E) to the *inline-closest p* and the network distance (d_{ij}^N or d_{ij}^U) to the *network-closest p* using Eqs. (3) and (6).

For example, consider $i \in V$ as an inconsistent node inside the walking range that is *inline-closest* to $p \in P$ and *network-closest* to $\bar{p} \in P$, such that $p \neq \bar{p}$. According to Section 3.3, it is true that $d_{ip}^E > d_{i\bar{p}}^U$ and the distance deviation is:

$$x_i = d_{ip}^E - d_{i\bar{p}}^U \qquad (12)$$

Such an approach can calculate the distance deviation to all the nodes in Ψ_p^W:

$$x^W = \left\{ d_{ip}^E - d_{i\bar{p}}^U | d_{i\bar{p}}^U < d_{i\bar{\bar{p}}}^U, \forall i \in \Psi_p^W, \forall p, \bar{p}, \bar{\bar{p}} \in P \right\} \qquad (13)$$

Once we have the distance deviation of all the inconsistencies, we can, for instance, discard the ones that are too close to the expected distance to their *inline-closest p* and also, we can perform a min–max unity-based normalization bringing all values into the range [0, 1]; in which values close to 0 indicate less or non-inconsistent nodes and values close to 1 indicate highly inconsistent nodes:

$$\mathfrak{x}_i^W = \frac{x_i^W - \min(x^W)}{\max(x^W) - \min(x^W)}, \forall i \in \left[1, \left|x^W\right|\right] \qquad (14)$$

This discussion used the set of walking inconsistencies $\left(\Psi_p^W\right)$, but it can be directly extended to the other types of inconsistencies presented in Section 3.4.

4. Algorithmic solution

Algorithm 1 tracks distance-based inconsistencies in complex networks by using a set *P* of |*P*| of points of interest. The algorithm instantiation requires the user to inform the type of inconsistency to be tracked regardless of the definition of inconsistency that is segmented into four types as in Section 3.4.

The algorithm starts by filling a set of empty sets, each one reserved to store the inconsistencies of a single point of interest (see lines 1–3). Subsequently, it uses p^E and p^c to store, respectively, the inline-closest and network-closest points of interest to an arbitrary node $v \in V$ (see line 5). As the algorithm runs for one inconsistency type at a time, p^c can refer to any closest point of interest, such as walking-closest or direct/undirected network-closest ones.

For simplicity purposes, the algorithm calls for two external functions — referred to as *InlineClosest* and *NetworkClosest* (see lines 7, 9, 11, and 12) — to extract the closest point of interest to a node by implementing all distance functions in Section 3.3. *InlineClosest* implements d_{ij}^E following Eq. (3) and **NetworkClosest** implements both d_{ij}^N and d_{ij}^U by adapting Eq. (6). Any additional variable to the previous ones is conditional to the inconsistency type that is being tracked, which is handled by the if–else used in lines 8 and 10.

Once the closest points of interest have been already identified (see line 14), the algorithm checks whether a node is an inconsistency or not using two possible validations. The first one (see line 16) is dedicated to inward, outward, and walking inconsistencies and the other one (see line 18) is used just in cases of absolute inconsistencies. The first test of the algorithm considers a node inconsistent if the inline-closest point of interest p^E and the network-closest point of interest p^c are not the same, in turn, the second one requires the node to be an inward and outward inconsistency so to be an absolute inconsistency. The output then is a set of inconsistencies of |*P*| points of interest (see line 21).

Algorithm 1. Inconsistency tracker

```
Data: $G = \{V, E\}$, $P \subset V$, and $c \in \{I, O, A, W\}$ — c is the inconsistency
      type
Result: $\{\Psi^c_p, \forall p \in P\}$ — inconsistencies set for all points of interest
      $p \in P$
1  $\Psi^c \leftarrow \emptyset$
2  for each $p \in P$ do
3      $\Psi^c_p \leftarrow \emptyset$ // notice that $\Psi^c_p \subset \Psi^c, \forall p \in P$, therefore $|\Psi^c| = |P|$
4  for each $v \in V$ do
5      $p^E \leftarrow p^c \leftarrow \emptyset$
6      for each $\bar{p} \in P$ do
7          $p^E \leftarrow$ InlineClosest$(v, \langle p^E, \bar{p} \rangle)$
8          if $c \in \{I, O, W\}$ then
9              $p^c \leftarrow$ NetworkClosest$(v, \langle p^c, \bar{p} \rangle, c)$
10         else
11             $p^I \leftarrow$ NetworkClosest$(v, \langle p^c, \bar{p} \rangle, I)$
12             $p^O \leftarrow$ NetworkClosest$(v, \langle p^c, \bar{p} \rangle, O)$
13             $p^c \leftarrow \{p^I, p^O\}$
14     if $p^E \neq \emptyset$ and $p^c \neq \emptyset$ then
15         if $c \in \{I, O, W\}$ then
16             if $p^c \neq p^E$ then
17                 $\Psi^c_{p^E} \leftarrow \Psi^c_{p^E} \cup \{v\}$ // v should be closer to $p^E$
18         else
19             if $p^c_1 \neq p^E$ and $p^c_2 \neq p^E$ then
20                 $\Psi^c_{p^E} \leftarrow \Psi^c_{p^E} \cup \{v\}$ // see comment on line 17
21 return $\Psi^c$
```

Given a graph $G = \{V, E\}$, a set *P* of |*P*| points of interest, and an inconsistent node $v \in V$; such node is known to be an inconsistency to one and only one $p \in P$.

- $\Psi^c_p \bigcap \Psi^c_{\bar{p}} = \emptyset - \forall p \in P, \forall \bar{p} \in P, p \neq \bar{p}$ — meaning mutual disjointedness.

It is possible to derive two other sets from a point of interest *p*: (i) the inconsistency set Ψ^c_p; and, (ii) the set of consistent nodes $\bar{\Psi}^c_p = \left(W^E_p \cup D^E_p\right) - \Psi^c_p$, such that $\bar{\Psi}^c_p \cap \Psi^c_p = \emptyset$. The consistent nodes are fundamental to the process of suggesting locations to points of interest because they provide a smaller average distance to the nodes in their perimeter, different than an inconsistent node.

Finally, we introduce Algorithm 2, that based on the concept of multi-scale distance-based inconsistencies was designed to suggest alterations in urban structures seeking improvements to the access *to*/*from* points of interest. Such an algorithm uses centrality to find locations that are equally accessible to all network nodes using a greedy approach instead of searching for ideal locations that would demand to test all possible combinations through an in-depth search.

Along these lines, we decided to use *Straightness Centrality* [3] as the centrality metric of Algorithm 2 because it analyzes the nodes of the network by combining both inline and network distances. Nevertheless, any distance-based centrality metric could be employed, as well as multiple metrics together; however, different metrics tend to provide dubious or wrong spots for relocation.

Algorithm 2. Inconsistency reducer

```
Data: $G = \{V, E\}$, $P \subset V$, and $c \in \{I, O, A, W\}$ — c is the inconsistency
      type
Result: R — set of suggested positions for points of interest
1  $R \leftarrow \emptyset,\ \bar{P} \leftarrow \emptyset,\ \tilde{P} \leftarrow \emptyset$
2  $\Psi^c \leftarrow$ InconsistencyTracker$(G, P, c)$
3  $\Phi^c \leftarrow \Psi^c$ // copy of the original set
4  while $|P| - |\bar{P}| > 0$ and $\left(\sum_{i=1}^{|P|} |\Psi_i^c| \geq \sum_{i=1}^{|P|} |\Phi_i^c|\right)$ do
5      $old_p \leftarrow \emptyset,\ new_p \leftarrow \emptyset$
6      for each $p \in (P - \bar{P})$ do
7          if $c \in \{I, O, A\}$ then
8              $G_p^E \leftarrow G\left(D_p^E - \Psi_p^c\right)$ // driving-consistent subgraph
9          else
10             $G_p^E \leftarrow G\left(W_p^E - \Psi_p^c\right)$ // walking-consistent subgraph
11         $\bar{p} \leftarrow$ extract_central$(G_p^E)$
12         $\mathbb{P} \leftarrow (((P - \bar{P}) \cup \tilde{P}) - \{p\}) \cup \{\bar{p}\}$
13         $\Omega^c \leftarrow$ InconsistencyTracker $(G, \mathbb{P}, c)$
14         if $\left(\sum_{i=1}^{|P|} |\Phi_i^c| > \sum_{i=1}^{|P|} |\Omega_i^c|\right)$ then
15             $\Phi^c \leftarrow \Omega^c$ // new lowest number of inconsistencies
16             $old_p \leftarrow p,\ new_p \leftarrow \bar{p}$
17     if $old_p \neq \emptyset$ and $new_p \neq \emptyset$ then
18         $R \leftarrow R \cup \{old_p, new_p\}$ // $old_p$ was transferred to $new_p$
19         $\bar{P} \leftarrow \bar{P} \cup \{old_p\}$ // old location
20         $\tilde{P} \leftarrow \tilde{P} \cup \{new_p\}$ // new location
21     else
22         break // achieved the best greedy result
23 return R
```

The algorithm starts by initializing auxiliary variables (see line 1) and by tracking inconsistencies from the original network (see line 2). Following, the algorithm enters in a loop (see line 4) until all points of interest have been moved once, or until there are no more inconsistencies to be reduced on the network. Inside the loop, the algorithm tries to change one point of interest at a time (see line 6) using candidate nodes that pertains to the induced subgraph G_p^E of consistent nodes (see lines 8 and 10) that can be nodes inside the walking or driving range, both of which are related to the inconsistency type $c \in \{O, I, A, W\}$.

The algorithm searches for the node that has the highest centrality value among all the other ones (see line 11) testing it as the new location to the point of interest. The algorithm, then, temporally replaces the node (see line 12) to collect information about the inconsistencies of such network configuration (see line 13). Subsequently, it tests whether the new configuration causes fewer inconsistencies then the former one (see line 14) before tagging the node for relocation (see lines 15 and 16). In a greedy manner the algorithm selects the point of interest that by being replaced will lead to the highest elimination of inconsistencies; after that, it performs integrity tests, marks the node as relocated, and it removes the old point of interest (see lines 17–20).

The output of the algorithm is a set R of new locations (see line 23) returned when there are no more profitable changes (see line 22) for the remaining points of interest; each element $r \in R$ is an ordered pair $r = \langle old_p, new_p \rangle$ of the current (old_p) node where a point of interest is and a better node (new_p) for placing it.

Algorithm 2 runs in $O(|V||P|^3)$ in the average case, where $|P|$ is the number of points of interest and $|V|$ is the number of nodes, $|P| \ll |V|$. The algorithm was proved finite and correct by Spadon et al. [7] as it never increases the number of inconsistencies in a city. It is straightly paralyzed and cache values to avoid recomputing, running in less than a minute over a 200,000-inhabitant's city. It noteworthy that the algorithm holds for any city size, from hundreds of thousands to millions of inhabitants. The algorithm performance scales linearly.

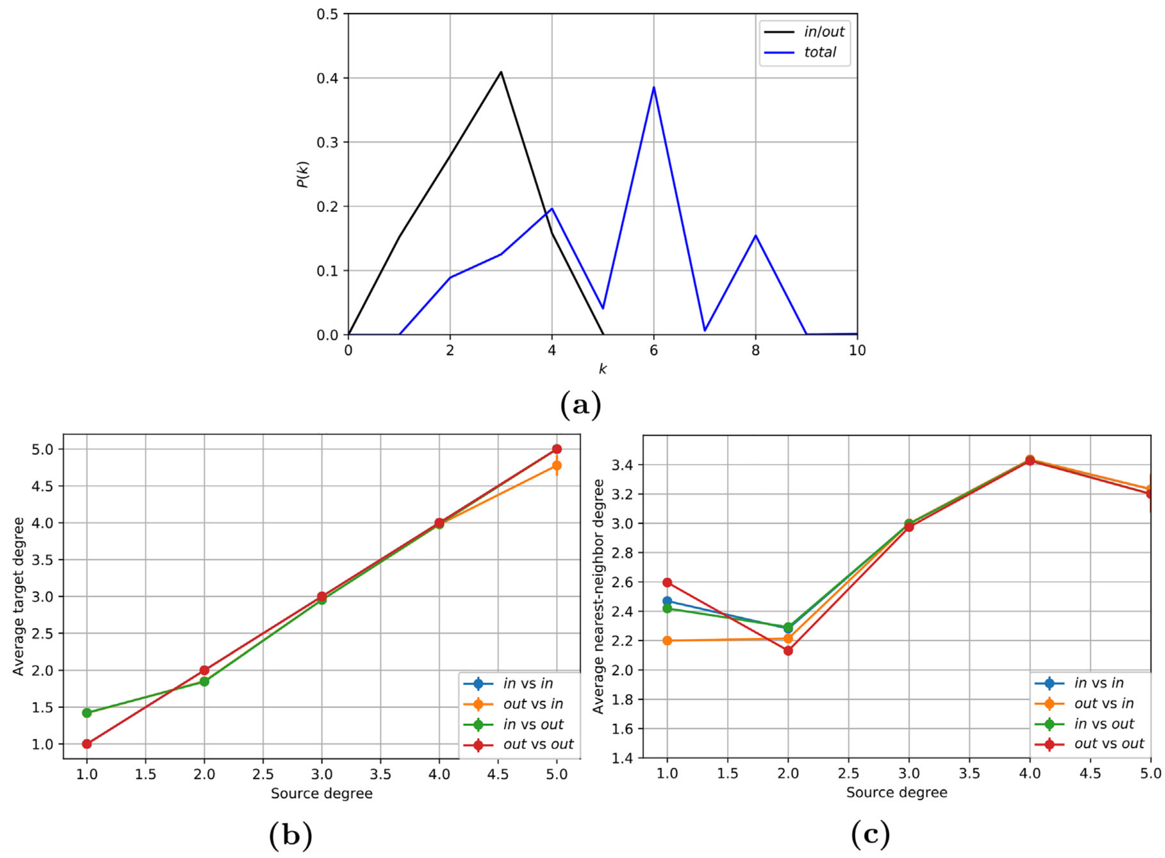


Fig. 1. Topological organization of Sao Carlos considering: (a) the in–out/total degree distribution of the network nodes; (b) the average target neighborhood degree, considering the source degree of a given node; and, (c) the average nearest neighbor degree correlation.

4.1. *Proposal generalization*

For simplicity's sake, we assume that: (i) any displacement goes through the city streets; (ii) both origin and destination of paths are network nodes; and, (iii) cities with uniform population distribution. Nevertheless, our tool-set is even for scenarios where these assumptions are not true.

More specifically, we use a general concept of weight, which can refer to travel time, edge capacity, route cost, and others. About the population distribution, it is possible, for instance, to use a normal distribution peaked at the center of the city, multimodal distributions, or census data. This information can aid in the analysis of urban structures if it is used to assign values to sets of nodes corresponding to the population's density of the area they belong to.

Also relevant is that fact that, despite being central to our problem formulation, the redesign and relocation of points of interest is not possible in most cases. However, the output of our tool-set can support decision-making in different ways: rethinking the allocation of resources, the building of new services, reestablishing the limits of services, and outlining the initial design of a city or neighborhood that, still, is in its early stage of project. More broadly, the set of inconsistencies, the output of our method, can benefit from the analysis of a specialist rather than being a categorical result.

Regarding other domains, for instance, our tool-set is also suitable to the model of computer networks, when it is necessary to add an extra switch, or router, or even when there is a need to reallocate such equipment. It can be used in the topological design of electronic circuits when it is possible to save on tracks just by redistributing specific components. It works for transportation networks in situations when it is possible to save on resources if the warehouse needs to be transferred to an adequate node of the trucks' network. In fact, the proposed tool-set fits many real-world scenarios modeled as complex networks.

5. Results and discussions

Our tool-set was validated over the Brazilian city of Sao Carlos. Such a city was instantiated as a complex network through a digital map from *OpenStreetMap* (OSM).[1] Given a map file extracted from OSM, there are two possibilities to build a network from it. The first one is through a primal graph [8], which considers streets as edges, and street crossings as nodes (as discussed in Section 3.1). The other one is through a dual graph [28] in which the streets are nodes, and street crossings are edges. Our study is based strictly on primal graphs because one cannot compute distances using non-georeferenced data as provided by dual graphs.

5.1. *Topological characterization*

The degree distribution of the city is depicted in Fig. 1a, which shows the portion — or simple probability $P(k)$ — of nodes that have a given degree k. The image shows that the city has a majority of nodes with total degree 2, 4, 6, and 8; in which, nodes with degree 6 are the most recurrent ones followed by nodes with degree 4, 8, and 2. Such topological configuration is compatible with a city in which: nodes with degree 6 indicate streets forming triangles or T-like structures; nodes with degree 4 indicate the existence of straight bi-directed paths used to link distant places; nodes with degree 8 point to well-designed block structures; and, those with degree 2 are the city limits or dead end streets.

[1] www.openstreetmap.org.

In addition, Fig. 1b and c shows the degree correlation and the average neighborhood connectivity of the network nodes. The first one depicts that, on average, the neighbors of a given node tend to have the same degree, pointing out a recurrent urban-design pattern. Such recurrence fails when correlating the in-degree of a source node with the out-degree of a target node when the source degree is 1 or 2, and also when correlating the opposite case when the source degree is 4 or 5. The second figure, in turn, depicts that the nearest connected neighbor tends to have a lower degree than the source node, meaning a trade-off between a higher probability of movement and shortest distances. This implies that movement through the shortest paths can lead the citizen to inconsistent regions with fewer alternative routes causing difficulty in moving out of these regions. In such a way, the trade-off has an important impact by explaining some of the city's inconsistencies, as we review in the next section.

5.2. *Inconsistencies among urban structures*

In this section, we discuss some of the inconsistent urban structures found in the city of Sao Carlos. Among all the existing points of interest, we chose to discuss the results about post offices, supermarkets, hospitals, police stations, and schools. The number of points of interest and related case studies is small because of the difficulty in acquiring maps with points of interest previously mapped within the street mesh, but the set of points of interest we selected are known to be relevant because they are part of the daily life of the citizens.

5.2.1. *Visualizing inconsistent regions*

Initially, we discuss post offices and supermarkets (grocery stores in general) because they are affected by walking inconsistencies. When using a supporting visualization, it is possible to detect regions that are full of inconsistencies, indicating neighborhoods that lack access regarding such points of interest.

In our visualizations, we depict inconsistencies at the same time that we omit consistent nodes. The inconsistencies follow a color scale called *Inconsistency Degree* defined in the range [0.0, 1.0]; such a scale was built following the definitions of Section 3.5. It is noteworthy the process of feature normalization considered the distance deviation of the inconsistencies found in both scenarios so to afford an accurate comparison between post-offices and supermarkets.

Consider Fig. 2, which depicts the walking inconsistencies of post offices. The majority of post offices — black stars — are located at the bottom of the figure, indicating that they are centered in a small part of the city. This same behavior can be observed regarding the inconsistent nodes, that seems to be neighboring the location of the post offices. Additionally, it is possible to notice that the most inconsistent nodes (according to the scale of inconsistency) are separated from their inline-closest point of interest by a big empty area. Such a case can indicate both a miss-connected region and an area surrounded by constructions that are obstructing the building of new streets. Either way, the inconsistencies are not limited to this case, but when alone or in small concentration, they should represent no major problem to the city access.

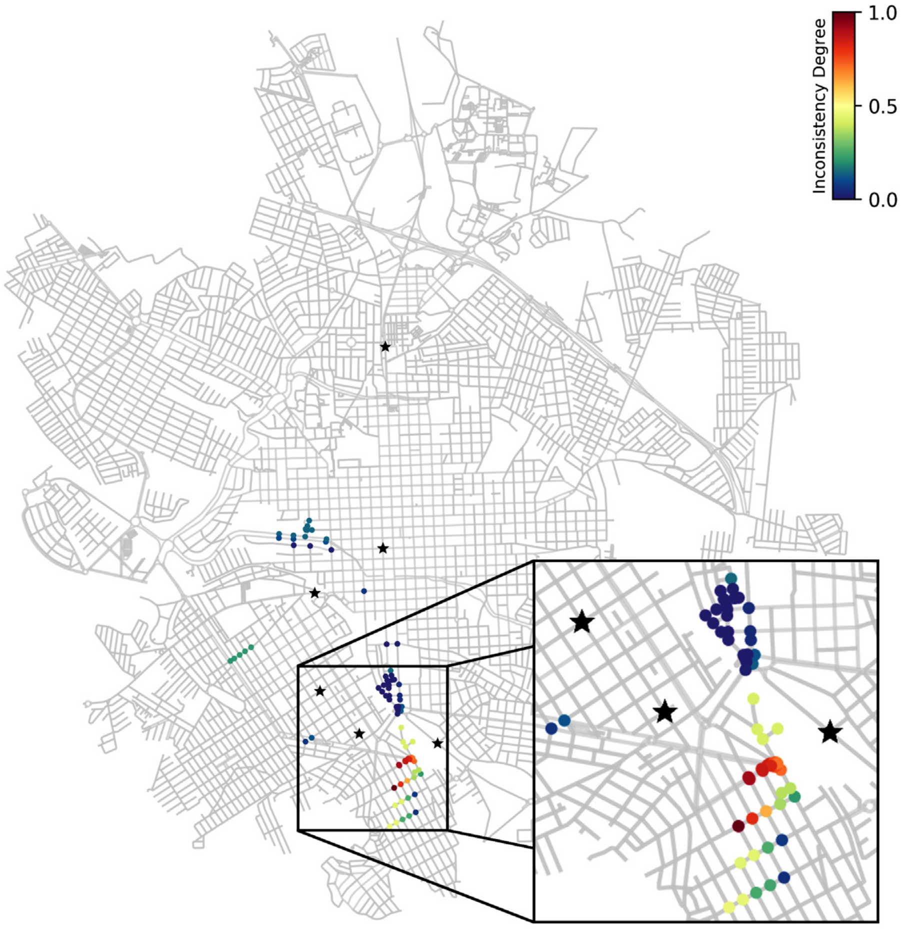


Fig. 2. The inconsistency degree of the walking inconsistencies when considering post offices as points of interest. We can observe that the inconsistencies are neighboring the points of interest (black colored nodes) and that the joining of inconsistent nodes in the bottom of the image indicates a whole area that lacks access when regarding such points of interest.

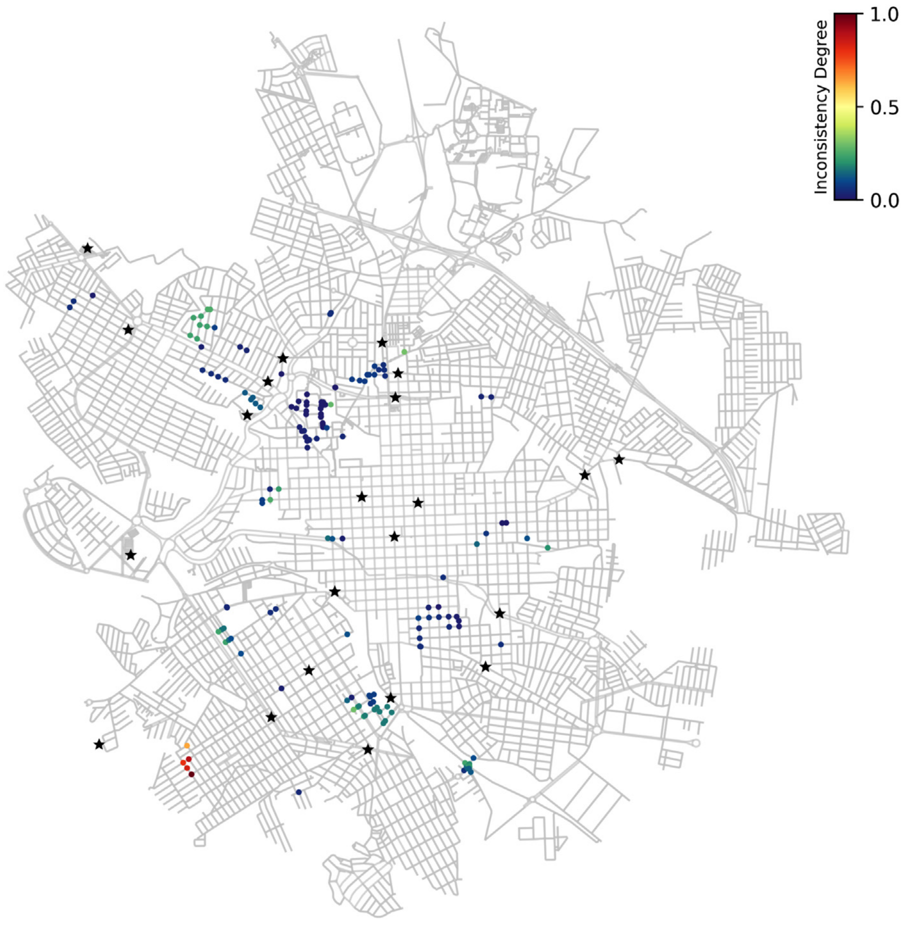


Fig. 3. The walking inconsistency degree when considering supermarkets as points of interest. The points of interest are spread all over the city creating more possibilities of locomotion to the citizens and reducing the degree of inconsistency of the nodes.

The results of the walking inconsistencies in the case of supermarkets are in Fig. 3. In this scenario, the points of interest — black stars — are spread all over the city along with their inconsistencies. However, it is possible to see that the inconsistent nodes have a lower degree in the scale of inconsistency. This means that the divergence between the expected distance to the inline-closest point of interest and the real distance to the network-closest point of interest is not as significant as we saw in the case of the post offices. This is due to a higher number of points of interest that provide more possibilities for the citizens.

Nonetheless, when joining the supermarkets' inconsistencies that are inside the walking-perimeter with those inside the driving-perimeter — as in Fig. 4, it is possible to notice that these points of interest still require attention. In such a scenario the mobility issue is spread all over the network, jeopardizing the access of the city as a whole. Such a result let us conclude that the nodes inside the walking range tend to have better access to the points of interest in a city rather than the ones that are within the driving range. This is because the further the citizens need to go, the more inconsistencies they tend to find in their routes. This is a characteristic of Sao Carlos, which has a trade-off between a higher probability of movement and shortest distances (see Section 5.1) leading the citizens to paths with more inconsistencies and fewer possibilities of movement.

5.2.2. *Assessing inconsistency recovery*

In this experiment, we have analyzed a set of 13 hospitals, 4 police stations, and 16 schools. The inconsistencies we tracked to all and each one of these cases are in Table 1 together with the total number of inconsistencies for hospitals, police stations, and schools that are respectively 559, 342, and 663. Regarding the order of magnitude from the point of interest with more inconsistencies to the one with fewer, we have schools followed by hospitals and then police stations, which is the same as the order of sets with more points of interest, suggesting that the occurrence of inconsistencies is connected to the number of points of interest. In fact, inconsistencies appear whenever different perimeters meet; as a consequence, more points of interest mean more perimeters, what tends to increase their number. Then, the challenge becomes to find suitable places to points of interest that reduce the number of inconsistencies in the city.

By using Algorithm 2, we were able to reduce the inconsistencies of Sao Carlos, suggesting the relocation of 6 hospitals, 2 police stations, and 9 public schools. As a result, the algorithm decreased 160 inconsistencies from hospitals (from 559 to 399), 123 inconsistencies from police stations (from 342 to 219), and 179 inconsistencies from public schools (from 663 to 484) — as shown in the right side of Table 1. Despite the inconsistencies of some points of interest raised in number when comparing the original and the enhanced city, the total number of inconsistencies were always smaller, as guaranteed by our algorithm.

5.2.3. *Supporting the designing of urban structures*

The tool-set we proposed is not only to be used in the automatic recovery of inconsistencies but also to aid human-made urban-planning decisions. This is the case when a specialist designs a city by having ground truth knowledge about citizens. In this case,

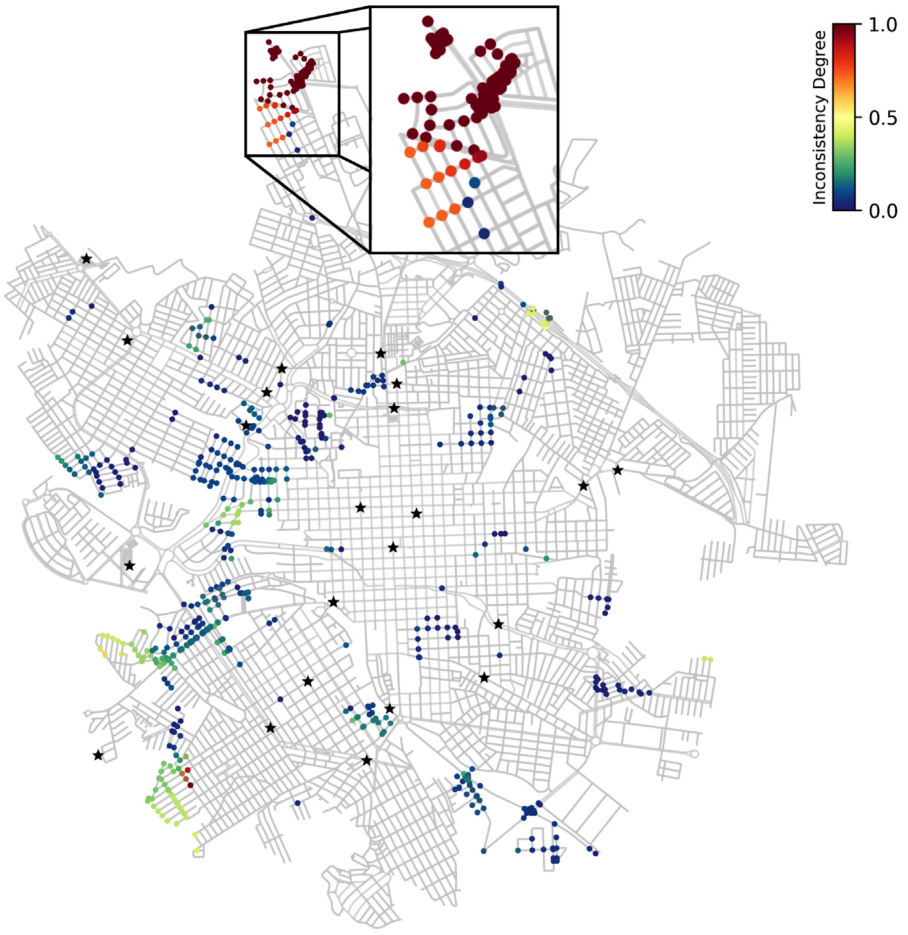


Fig. 4. A view of the inconsistency degree of the city when the points of interest are supermarkets that are both in the walking or driving range of the network nodes. In such a case we can observe that major sets of nodes at the top of the image are suffering from the highest degree of inconsistency, which indicates the lack of access to a whole residential area.

Table 1
Inconsistencies of the city of Sao Carlos, in which the points of interest are police stations, hospitals, and public schools; we use # to refer to the total number of inconsistencies and % to their percentage (reprinted by permission from *Springer Customer Service Centre GmbH*: Springer Nature, A Distance-Based Tool-Set to Track Inconsistent Urban Structures Through Complex-Networks [Spadon et al. 2018], Springer International Publishing AG).

*n*th POI	Original city						Enhanced city						*n*th POI
	Hospitals		Police Stations		Schools		Hospitals		Police Stations		Schools		
	#	%	#	%	#	%	#	%	#	%	#	%	
1	13	2.3%	32	9.3%	15	2.2%	14	3.5%	30	13.7%	19	3.9%	1
2	2	0.3%	4	1.1%	77	11.6%	2	0.5%	48	21.9%	13	2.6%	2
3	12	2.1%	86	25.1%	43	6.4%	18	4.5%	96	43.8%	37	7.6%	3
4	19	3.4%	29	8.4%	71	10.7%	4	1.0%	32	14.6%	58	11.9%	4
5	30	5.3%	191	55.8%	114	17.1%	87	21.8%	13	5.9%	57	11.7%	5
6	49	8.7%	–	–	3	0.4%	51	12.7%	–	–	1	0.2%	6
7	145	25.9%	–	–	8	1.2%	26	6.5%	–	–	1	0.2%	7
8	39	6.9%	–	–	15	2.2%	22	5.5%	–	–	18	3.7%	8
9	12	2.1%	–	–	78	11.7%	31	7.7%	–	–	77	15.9%	9
10	43	7.6%	–	–	51	7.6%	63	15.7%	–	–	48	9.9%	10
11	72	12.8%	–	–	38	5.7%	45	11.2%	–	–	41	8.4%	11
12	95	16.9%	–	–	15	2.2%	17	4.2%	–	–	11	2.2%	12
13	28	5.0%	–	–	56	8.4%	19	4.7%	–	–	10	2.0%	13
14	–	–	–	–	8	1.2%	–	–	–	–	16	3.3%	14
15	–	–	–	–	60	9.0%	–	–	–	–	51	10.5%	15
16	–	–	–	–	11	1.6%	–	–	–	–	26	5.3%	16
Total	559	100%	342	100%	663	100%	399	100%	219	100%	484	100%	Total

Algorithms 1 and 2 can provide for the analysis and suggestion of distance-efficient locations that are feasible to points of interest.

This section introduces three hypothetical case studies that depict our tool-set in practice. The first two were conducted considering a subset of hospitals and public schools of the Brazilian city of Sao Carlos (see Section 5.2.2). The last one introduces a discussion on how to improve police stations in cases where reducing inconsistencies by relocating them is not possible. Nonetheless, our

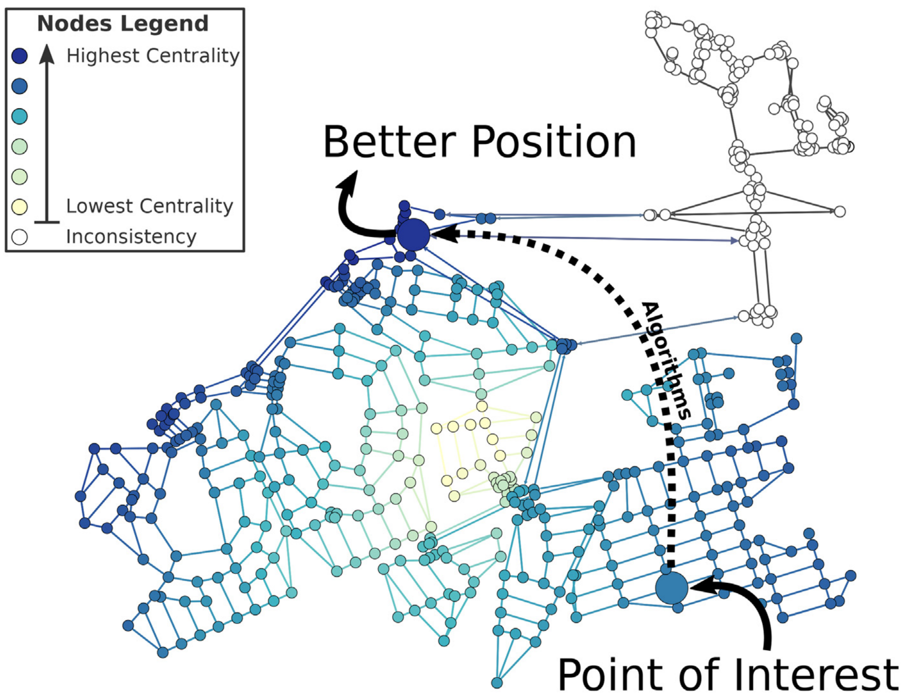


Fig. 5. Illustration of the process of designing urban structures under the light of centrality metrics. This process starts by identifying nodes of interest, then it follows by tracking their inconsistencies, and it ends by suggesting locations to these nodes that reduce the total number of inconsistencies of the city (reprinted by permission from *Springer Customer Service Centre GmbH*: Springer Nature, A Distance-Based Tool-Set to Track Inconsistent Urban Structures Through Complex-Networks [Spadon et al. 2018], Springer International Publishing AG).

tool-set is expandable to any point of interest since it is identical to all of them.

The reader must consider that the case studies considering only hospitals, schools, and police stations are canonical; that is, they do not take into account the more intricate nature of each point of interest. This was a choice to improve the briefness and generalization of the presentation; nevertheless, real-world settings should consider more intricate points of interest. For example, it is common to find communities that have schools of different grades separated just by a few blocks. Similarly, multiple hospitals dedicated to specific religious communities close to each other. Our canonical modeling approach would find such cases to be inconsistencies. These are not limitations to the proposal, but limitations to a given instance (implementation) of it. A reasonable implementation would consider the possibility of customizing the points of interest according to several possibilities, as the ones just mentioned.

The case studies from bellow follow as in Fig. 5, in which we start analyzing the problems of a given point of interest, next we try to solve it by ourselves, and then we use our algorithms, under the light of centrality metrics, to improve the results. It is noteworthy that all case studies are depicted by the induced subgraph of the point of interest under analysis and, although we have illustrated some inconsistent nodes in Fig. 5, in the case studies they are not visible because they do not provide useful visual information to the images. The following case studies refrain from using the same supporting visualization from Section 5.2.1 because we are assuming to have urban-planning knowledge, so we use the inconsistencies as a metric helping to solve an already known issue.

5.2.4. *Case study 1: creating a new hospital to reduce the demand*

From the set of hospitals of the city of Sao Carlos, we identified one that, when compared to the others, has excessive nodes in its perimeter (see Fig. 6a). There is no specific explanation about the hospital's location and; we can hypothesize that the city may have grown after the hospital has been built or the planners did not take the surroundings of the hospital into account. One thing is for sure, an extensive area with an ill-positioned point of interest will deprive the access of the nodes; in this case, when points of interest are healthcare facilities, time-critical activities, as the transportation of patients in a critical state, can be jeopardized by the lack of street access. Hence, the problem becomes where to build a new hospital and how to avoid inconsistencies.

First, we tried to solve the problem manually, visually choosing a location that could provide nodes with similar characteristics in the perimeters of both hospitals. Fig. 6a shows a possible place to the new hospital as well as the resulting perimeter of both of them, which are defined by a line that cuts the image in half. We inserted the proposed location in the set of hospitals, and we used Algorithm 1 to track the inconsistencies in the resulting configuration. Such a configuration leads us to 615 inconsistencies, which is a higher value than the original city. We succeeded in building a hospital that splits the perimeter into two, but we failed in providing better access to the old and new hospitals.

In a second approach, we analyzed the nodes' centrality together with a supporting visualization. We colored the nodes by their centrality, what allowed us to notice that the selected location for the new hospital is a node with low centrality. Then, we used Algorithm 2 to suggest a better place for the new hospital while keeping the location of the old one. Doing so, the city inconsistencies were reduced from 615 to 352 (see Fig. 6b), which positively reflected in the mobility of this area by distributing the demand between both hospitals. Thus, creating a new hospital in a specific location was able to reduce almost half of the inconsistencies of the city without relocating the old hospital. Along these lines, it is our expectation that through Algorithms 1 and 2 city planners can rely on a computer-based mechanism able to "crunch" digital maps that are way too big and complex for a human being to exhaustively evaluate.

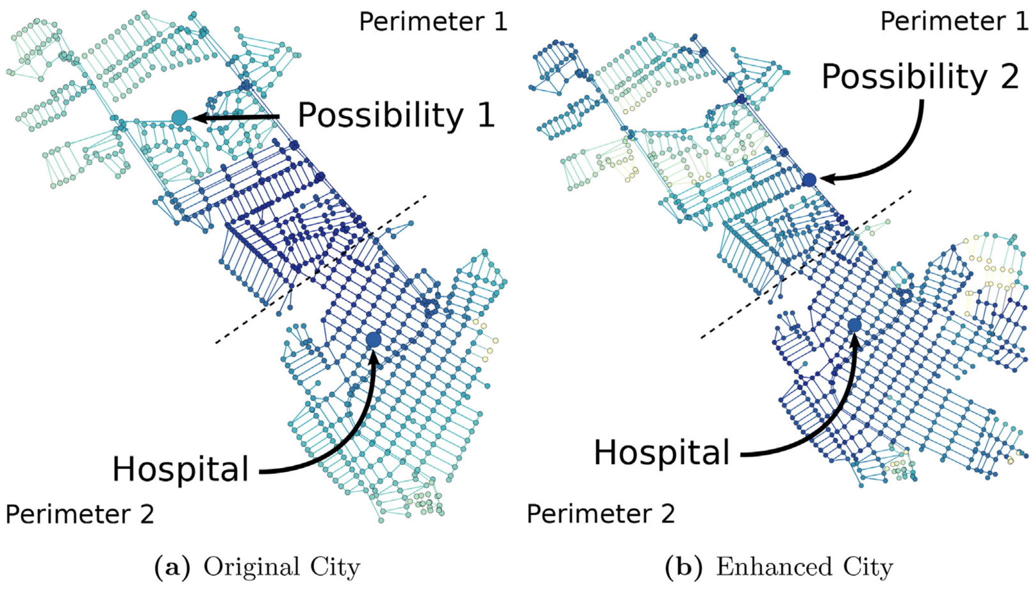


Fig. 6. Illustration of the assisted urban planning task from the first case study, in which the point of interest is a hospital and the color of the nodes indicates their centrality — the darker, the higher. Fig. (a) shows a hospital's perimeter that is too large, causing lack of access. This way, we placed a new hospital in a central location in that same area to solve the issue. Afterward, we used the algorithm to reduce inconsistencies, which suggested relocating the new hospital to a more central location, reducing the inconsistencies. Fig. (b) depicts the result (reprinted by permission from *Springer Customer Service Centre GmbH*: Springer Nature, A Distance-Based Tool-Set to Track Inconsistent Urban Structures Through Complex-Networks [Spadon et al. 2018], Springer International Publishing AG).

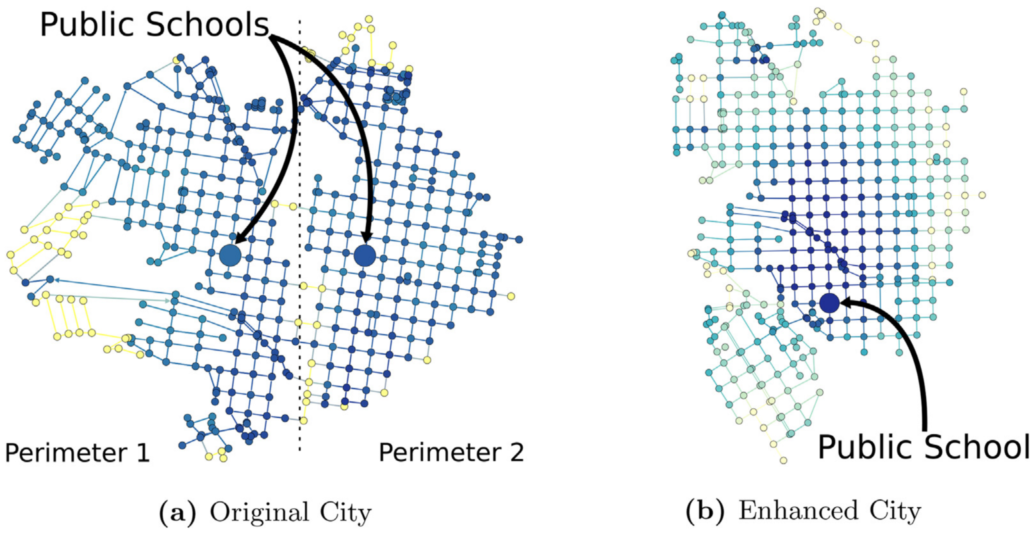


Fig. 7. Illustration of the second case study, in which the points of interest are public schools and the color of the nodes denotes their centrality — the darker, the higher. In this case study, we treated a problem related to the waste of resources that was caused by having two schools near each other; Fig. (a) shows the area, which is small, increasing the drawbacks related to access. By merging both schools, we achieved a better coverage of nodes, as depicted in Fig. (b) (reprinted by permission from *Springer Customer Service Centre GmbH*: Springer Nature, A Distance-Based Tool-Set to Track Inconsistent Urban Structures Through Complex-Networks [Spadon et al. 2018], Springer International Publishing AG).

5.2.5. Case study 2: merging schools to centralize public resources

In the second case study, we identified two public schools that are close to each other and serve a small set of nodes. In this case, the proximity of the schools (see Fig. 7a) is a problem since none of them is used up to its capacity, implying a waste of public resources. In a first approach, by using Algorithm 2 to relocate them, the number of inconsistencies was reduced from 663 to 635.

Considering the size of the perimeter of both schools, we decided to remove one school to improve the utility of the one that remained. By centralizing the schools in a single node, we can reduce inconsistencies because there will be fewer perimeters bordering each other; hence, the inconsistencies, located whenever two of them meet, will be naturally decreased. To further enhance this process, we used the color-coded centrality metric to choose a candidate to be the new sole school. Afterward, we used Algorithm 2 to find a better location (see Fig. 7b), which reduced the number of inconsistencies from 635 to 445.

5.2.6. Case study 3: how to decision-making based on the detection of inconsistencies

Previously, we revised cases in which the location of points of interest was altered to enhance the city's mobility by reducing the number of inconsistencies. However, some scenarios cannot bear such structural changes, but we can still use the concept of inconsistency to help in planning urban logistics. In this case, inconsistent nodes are not considered to be problems but, rather, indicators.

For example, consider the case of police stations, assuming that only five stations must assist all the city. The problem here is that the police stations are not equally distributed along the city, and despite reducing inconsistencies, we have a majority of network nodes closest just to a few stations.

Instead of redistributing such points of interest, this information can be useful to improve the service provided by them. For instance, the coverage of each station can be defined manually, in such a way that some stations will face more inconsistencies, but they will cover ranges similar to the other ones, or instead, indicating stations that require more human resources and supplies.

Hence, the decision-making process would join knowledge about the logistics of the police stations and also about the topology and mobility of the network. The expected result of such a process would be an equilibrium between the theoretical and practical knowledge. This process hardly would be able to reduce inconsistencies in its algebraic formulation, but instead, it would enhance the logistics related to the service provided by all the police stations in the city.

6. Conclusion

In this article, we proposed a set of algebraic formalisms and algorithms to track and reduce distance-based inconsistencies considering multiple means of displacement so to improve mobility issues *to/from* points of interest in a city. We provided quantitative and qualitative results, all of which provided useful and unprecedented knowledge for decision-making activities in cities aiding in reducing the number of inconsistencies and improving the mobility of a city. Our results are based on the analysis of street networks from digital maps provided by *OpenStreetMap* that is a source extensively used in the related literature, and through them, we provide results as precise as the quality of the digital maps.

More specifically, our contributions elucidate intrinsic problems found in urban structures, which are based on the location of points of interest through considering multiple types of displacements. We materialized our developments in the form of two algorithms to track and reduce inconsistent nodes in complex networks. Finally, we demonstrated our methodology in quantitative and qualitative real-world studies considering points of interest from the Brazilian city of Sao Carlos.

In summary, we explained how to treat urban structures from the perspective of complex networks, from data modeling to computer-assisted urban design, contrasting our contribution to others in the related literature. The proposed methods were algebraically formalized and empirically demonstrated, granting to our tool-set potential for prompt contribution and for opening new research questions. As a future work, we shall embrace link prediction methods for suggesting changes in the network topology — *i.e.*, proposing variations in the flow's direction, while searching for better urban topological configurations.

Acknowledgment

We would like to thank the Brazilian agencies CNPq (grant 167967/2017-7), FAPESP (grants 2016/17078-0, 2016/02557-0, and 2017/08376-0), and CAPES.

References

[1] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D. Hwang, Complex networks: Structure and dynamics, Phys. Rep. 424 (4-5) (2006) 175–308, http://dx.doi.org/10.1016/j.physrep.2005.10.009.
[2] S. Porta, V. Latora, F. Wang, E. Strano, A. Cardillo, S. Scellato, V. Iacoviello, R. Messora, Street centrality and densities of retail and services in Bologna, Italy, Environ. Plan. B: Plan. Des. 36 (3) (2009) 450–465, http://dx.doi.org/10.1068/b34098.
[3] P. Crucitti, V. Latora, S. Porta, Centrality measures in spatial networks of urban streets, Phys. Rev. E: Stat. Nonlinear Soft Matter Phys. 73 (3) (2006), http://dx.doi.org/10.1103/physreve.73.036125.
[4] G. Pan, G. Qi, W. Zhang, S. Li, Z. Wu, L.T. Yang, Trace analysis and mining for smart cities: issues, methods, and applications, IEEE Commun. Mag. 51 (6) (2013) 120–126, http://dx.doi.org/10.1109/mcom.2013.6525604.
[5] W. Zhang, K. Cao, S. Liu, B. Huang, A multi-objective optimization approach for health-care facility location-allocation problems in highly developed cities such as Hong Kong, Computers, Environ. Urban Syst. 59 (2016) 220–230, http://dx.doi.org/10.1016/j.compenvurbsys.2016.07.001.
[6] International Association of Public Transport (IAPT), World Metro Figures 2018, September 2018, URL: https://www.uitp.org/world-metro-figures-2018.
[7] G. Spadon, B.B. Machado, D.M. Eler, J.F. Rodrigues, A distance-based tool-set to track inconsistent urban structures through complex-networks, in: Y. Shi, H. Fu, Y. Tian, V.V. Krzhizhanovskaya, M.H. Lees, J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2018, Springer International Publishing, Cham, 2018, pp. 288–301.
[8] S. Porta, P. Crucitti, V. Latora, The network analysis of urban streets: a primal approach, Environ. Plan. B: Plan. Des. 33 (5) (2006) 705–725, http://dx.doi.org/10.1068/b32045.
[9] C. Cui, Z. Han, Spatial patterns of retail stores using POIs data in Zhengzhou, China, in: 2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM), Institute of Electrical and Electronics Engineers (IEEE), 2015, http://dx.doi.org/10.1109/icsdm.2015.7298031.
[10] Y. Xiao, C. Webster, S. Orford, Identifying house price effects of changes in urban street configuration: an empirical study in Nanjing, China, Urban Stud. 53 (1) (2014) 112–131, http://dx.doi.org/10.1177/0042098014560500.
[11] T. Davies, S.D. Johnson, Examining the relationship between road structure and burglary risk via quantitative network analysis, J. Quantit. Criminol. 31 (3) (2014) 481–507, http://dx.doi.org/10.1007/s10940-014-9235-4.
[12] E. Strano, M. Viana, L.F. Costa, A. Cardillo, S. Porta, V. Latora, Urban street networks, a comparative analysis of ten European cities, Environ. Plan. B: Plan. Des. 40 (6) (2013) 1071–1086, http://dx.doi.org/10.1068/b38216.
[13] S. Scellato, A. Cardillo, V. Latora, S. Porta, The backbone of a city, Eur. Phys. J. B: Condens. Matter Complex 50 (1–2) (2006) 221–225, http://dx.doi.org/10.1140/epjb/e2006-00066-4, arXiv:0511063.
[14] L.F. Costa, B.A.N. Travençolo, M.P. Viana, E. Strano, On the efficiency of transportation systems in large cities, Europhys. Lett. 91 (1) (2010), http://dx.doi.org/10.1209/0295-5075/91/18003.
[15] E. Strano, V. Nicosia, V. Latora, S. Porta, M. Barthélemy, Elementary processes governing the evolution of road networks, Sci. Rep. 2 (2012), http://dx.doi.org/10.1038/srep00296.
[16] M.B. Lowry, R.J. Balling, An approach to land-use and transportation planning that facilitates city and region cooperation, Environ. Plan. B: Plan. Des. 36 (3) (2009) 487–504, http://dx.doi.org/10.1068/b34055.
[17] B. Huang, W. Zhang, Sustainable land-use planning for a downtown lake area in central China: multiobjective optimization approach aided by urban growth modeling, J. Urban Plan. Dev. 140 (2) (2014) 04014002, http://dx.doi.org/10.1061/(asce)up.1943-5444.0000186.
[18] M.P. Viana, L.F. Costa, Fast long-range connections in transportation networks, Phys. Lett. A: Gen. Atom. Solid State Phys. 375 (15) (2011) 1626–1629, http://dx.doi.org/10.1016/j.physleta.2011.03.006.
[19] B. Travençolo, L.F. Costa, Accessibility in complex networks, Phys. Lett. A: Gen. Atom. Solid State Phys. 373 (1) (2008) 89–95, http://dx.doi.org/10.1016/j.physleta.2008.10.069.
[20] H.N. Huynh, E. Makarov, E.F. Legara, C. Monterola, L.Y. Chew, Characterisation and comparison of spatial patterns in urban systems: a case study of u.s. cities, J. Comput. Sci. 24 (2018) 34–43, http://dx.doi.org/10.1016/j.jocs.2017.12.001.
[21] A.P. Masucci, K. Stanilov, M. Batty, Limited urban growth: London's street network dynamics since the 18th century, PLoS ONE 8 (8) (2013), http://dx.doi.org/10.1371/journal.pone.0069469.
[22] A.T. Kaczynski, M.J. Koohsari, S.A.W. Stanis, R. Bergstrom, T. Sugiyama, Association of street connectivity and road traffic speed with park usage and park-based physical activity, Am. J. Health Promot. 28 (3) (2014) 197–203, http://dx.doi.org/10.4278/ajhp.120711-QUAN-339.
[23] M. Zamith, R.C.P. Leal-Toledo, E. Clua, E.M. Toledo, G.V. de Magalhães, A new stochastic cellular automata model for traffic flow simulation with drivers' behavior prediction, J. Comput. Sci. 9 (2015) 51–56, http://dx.doi.org/10.1016/j.jocs.2015.04.005.
[24] A. Sopan, P.J. Rey, B. Shneiderman, The dynamics of web-based community safety groups: lessons learned from the nation of neighbors, IEEE Signal Process. Mag. 30 (6) (2013) 157–162, http://dx.doi.org/10.1109/MSP.2013.2276513.

[25] D. Edwards, T. Griffin, Understanding tourists' spatial behaviour: GPS tracking as an aid to sustainable destination management, J. Sustain. Tour. 21 (4) (2013) 580–595, http://dx.doi.org/10.1080/09669582.2013.776063.
[26] P. Crucitti, V. Latora, S. Porta, Centrality in networks of urban streets, Chaos 16 (1) (2006) 015113, http://dx.doi.org/10.1063/1.2150162.
[27] B. Hillier, Space is the Machine: A Configurational Theory of Architecture, Space Syntax, 2007.
[28] S. Porta, P. Crucitti, V. Latora, The network analysis of urban streets: a dual approach, Physica A 369 (2) (2006) 853–866, http://dx.doi.org/10.1016/j.physa.2005.12.063.
[29] A. Cardillo, S. Scellato, V. Latora, S. Porta, Structural properties of planar graphs of urban street patterns, Phys. Rev. E: Stat. Nonlinear Soft Matter Phys. 73 (6) (2006), http://dx.doi.org/10.1103/physreve.73.066107.
[30] K. Maruhashi, J. Shigezumi, N. Yugami, C. Faloutsos, EigenSP: a more accurate shortest path distance estimation on large-scale networks, 2012 IEEE 12th International Conference on Data Mining Workshops, Institute of Electrical and Electronics Engineers (IEEE) (2012), http://dx.doi.org/10.1109/ICDMW.2012.110.
[31] G. Spadon, G. Gimenes, J.F. Rodrigues Jr., Identifying Urban Inconsistencies via Street Networks, vol. 108, Elsevier BV, 2017, pp. 18–27, http://dx.doi.org/10.1016/j.procs.2017.05.103, International Conference on Computational Science, ICCS 2017, 12–14 June 2017, Zurich, Switzerland.
[32] E. Galbrun, K. Pelechrinis, E. Terzi, Urban navigation beyond shortest route: the case of safe paths, Inf. Syst. 57 (2016) 160–171, http://dx.doi.org/10.1016/j.is.2015.10.005.
[33] J. Gil, Street network analysis "edge effects": examining the sensitivity of centrality measures to boundary conditions, Environ. Plan. B: Plan. Des. (2016), http://dx.doi.org/10.1177/0265813516650678.
[34] P. Corcoran, M. Jilani, P. Mooney, M. Bertolotto, Inferring semantics from geometry, in: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems – GIS'15, Association for Computing Machinery (ACM), 2015, http://dx.doi.org/10.1145/2820783.2820822.
[35] M. Barthélemy, A. Flammini, Modeling urban street patterns, Phys. Rev. Lett. 100 (13) (2008), http://dx.doi.org/10.1103/PhysRevLett.100.138702.
[36] C. Zhong, S.M. Arisona, X. Huang, M. Batty, G. Schmitt, Detecting the dynamics of urban structure through spatial network analysis, Int. J. Geograph. Inf. Sci. 28 (11) (2014) 2178–2199, http://dx.doi.org/10.1080/13658816.2014.914521.
[37] S. Shiode, N. Shiode, Network-based space-time search-window technique for hotspot detection of street-level crime incidents, Int. J. Geograph. Inf. Sci. 27 (5) (2013) 866–882, http://dx.doi.org/10.1080/13658816.2012.724175.
[38] S. Myronenko, A. Wenger, S. Atmazhov, Capacity analysis of the street and road network of modern regional center, Odes'kyi Politechnichnyi Universytet, Pratsi (3) (2016) 21–25, http://dx.doi.org/10.15276/opu.3.50.2016.06.
[39] G. Spadon, G. Gimenes, J.F. Rodrigues, Topological street-network characterization through feature-vector and cluster analysis, in: Y. Shi, H. Fu, Y. Tian, V.V. Krzhizhanovskaya, M.H. Lees, J. Dongarra, P.M.A. Sloot (Eds.), Computational Science – ICCS 2018, Springer International Publishing, Cham, 2018, pp. 274–287.
[40] Y. Yang, A.V. Diez-Roux, Walking distance by trip purpose and population subgroups, Am. J. Prev. Med. 43 (1) (2012) 11–19, http://dx.doi.org/10.1016/j.amepre.2012.03.015.
[41] M. van Steen, Graph Theory and Complex Networks: An Introduction, Maarten van Steen, 2010.

Gabriel Spadon is a Ph.D. candidate at the University of Sao Paulo, Brazil. His research focus is the analysis and characterization of real-world systems in the form of complex networks. He has published on topics in knowledge discovery in graphs and data-driven decision making. He is presently working on the analysis and optimization of urban systems using complex networks, data mining, and related concepts.

Bruno B. Machado is an Associate Professor in the Computer Science department at the Federal University of Mato Grosso do Sul, Brazil. He obtained his M.Sc. and Ph.D. in Computer Science from the University of Sao Paulo, in 2010 and 2016, respectively. His main research interests include machine learning, computer vision, and complex networks. He is currently leading projects focused on developing softwares to enable visual recognition of objects using deep learning methods.

Danilo M. Eler is Assistant Professor in the Department of Mathematics and Computation at the Sao Paulo State University, Brazil. He obtained his M.Sc. and Ph.D. in Computer Science from the University of Sao Paulo. His main research interests include information visualization, visual data mining, visual analytics, data science and computer vision.

Jose F. Rodrigues-Jr is an Associate Professor at the University of Sao Paulo, Brazil. He received his Ph.D. from this same university, part of which was carried out at Carnegie Mellon University, USA, in 2007. Rodrigues is a regular reviewer and author in his field, having contributed with publications in major journals and conferences. His topics of research include data science, machine learning, content-based data retrieval, visualization, and the application of such techniques in the analysis of medical, agriculture, and e-learning domains.

CHAPTER

3

HUMAN MOBILITY FORECASTING

In this chapter, we reproduce the following article:

Spadon *et al.* 2019, *Reconstructing Commuters Network using Machine Learning and Urban Indicators*. Scientific Reports. Springer Nature.

The investigation covered in this chapter proposes a model for predicting the number of people commuting from one city to another when taking into account the urban indicators from both source and target cities. As a result, we contribute with a model based on a single machine learning algorithm capable of predicting the flux and the number of people commuting between pairs of cities. Our results are based on modeling commuters fluxes using the Brazilian population censuses of 2010. The dataset was built considering one unity of flux as a person who does not work in the same city where he/she lives and that daily commute between cities. Such data was used with previously published models from the literature so to compare their performance with ours. The tests revealed that our proposal is about four times more accurate than others devised with the same purpose, achieving 90.4% of accuracy and 77.6% of R^2 Score. Furthermore, we show that the other models cannot make predictions consistent with the observed data. More specifically, our contributions are in: (1) reconstructing the Brazilian network of commuters fluxes via machine learning and urban indicators; (2) showing that the modeling of human mobility is dependent on variables beyond the population size and distance between cities; and, (3) the analysis of the impact of urban indicators in choosing a city for work and residence.

We reproduce the article from Spadon *et al.* 2019 under the *Rights and Permissions* below on the following pages. To present the related contribution: Section 1 reviews other models from the related literature used to predict commuters fluxes; Section 2 discusses the results of the proposed machine learning technique; Section 3 exposes the conclusions and final remarks; and, Section 4 presents our dataset and methodology. Additionally, in Appendix B, we provide the supplementary material linked with the discussed article.

SCIENTIFIC REPORTS
natureresearch

OPEN

Reconstructing commuters network using machine learning and urban indicators

Received: 17 May 2019
Accepted: 1 August 2019
Published online: 13 August 2019

Gabriel Spadon[1], Andre C. P. L. F. de Carvalho[1], Jose F. Rodrigues-Jr[1] & Luiz G. A. Alves[1,2]

Human mobility has a significant impact on several layers of society, from infrastructural planning and economics to the spread of diseases and crime. Representing the system as a complex network, in which nodes are assigned to regions (*e.g.*, a city) and links indicate the flow of people between two of them, physics-inspired models have been proposed to quantify the number of people migrating from one city to the other. Despite the advances made by these models, our ability to predict the number of commuters and reconstruct mobility networks remains limited. Here, we propose an alternative approach using machine learning and 22 urban indicators to predict the flow of people and reconstruct the intercity commuters network. Our results reveal that predictions based on machine learning algorithms and urban indicators can reconstruct the commuters network with 90.4% of accuracy and describe 77.6% of the variance observed in the flow of people between cities. We also identify essential features to recover the network structure and the urban indicators mostly related to commuting patterns. As previously reported, distance plays a significant role in commuting, but other indicators, such as Gross Domestic Product (GDP) and unemployment rate, are also driven-forces for people to commute. We believe that our results shed new lights on the modeling of migration and reinforce the role of urban indicators on commuting patterns. Also, because link-prediction and network reconstruction are still open challenges in network science, our results have implications in other areas, like economics, social sciences, and biology, where node attributes can give us information about the existence of links connecting entities in the network.

Humans move daily to work, do business, have leisure, meet people, and perform routine activities. Modeling human mobility is vital to better allocate resources and to improve the impacts of human activities in the community (nearby people) and the environment (cities and nature). From physics and mathematics to geography and social sciences, several researchers have tried different approaches to understand the impacts of human movement on society and vice-versa[1,2]. Human mobility is shaped by the urban organization[3] and can also change cities as an effect of traffic congestion[4]. Mobility patterns are associated with energy use[5,6], the spread of diseases[7–10], the occurrence of crimes[11,12], and others. Therefore, good predictive models can, for instance, help improve daily human activities with better urban planning, and also help policy-makers with more informed decisions to intervene in the disease spreading and crime.

Prediction of migration from one area to another, at different time scales, is one of the most critical challenges in human mobility. It is common to represent the system as a spatial complex network, where each node represents a region (*e.g.*, city) and the links between two regions indicate the flow of people. Thus, physics-inspired models such as the gravitation[13,14] and radiation models[15,16] have been used to predict the edges of the network as well as their weights (the number of people migrating). These models assume that the number of people going from one region/node to another decays with the distance separating them and is proportional to the populations' masses of these regions[13–16]. However, this assumption often fails to accurately describe the flow of people because of other factors that can increase or decrease mobility, such as the underlying transportation network[17], socio-economic aspects[3,18–20], and transport congestion[4]. Usually, these models are limited to predicting the weights of existing links, and they overestimate the number of connections between nodes when dealing with sparse

[1]University of Sao Paulo, Institute of Mathematics and Computer Sciences, Sao Carlos, SP, 13566-590, Brazil. [2]Northwestern University, Department of Chemical and Biological Engineering, Evanston, IL, 60208-3112, USA. Correspondence and requests for materials should be addressed to G.S. (email: spadon@usp.br) or L.G.A.A. (email: lgaalves@northwestern.edu)

mobility networks. Therefore, these limitations make it hard to generalize the model on unseen data and to reconstruct the structure of human mobility networks.

Link prediction and network reconstruction is a very active area of research in network science[21]. Most of the literature on link prediction is based on evaluating the similarity between nodes and suggesting missing links. For instance, the metrics used to evaluate the existence of links include, but are not limited to, the number of common neighbors[22,23], the existence of short paths between nodes[24], measures of centrality[25], and hierarchical structures within the network[26]. Reconstruction techniques also include methods based on maximum-entropy distribution regarding the node degree and the flow strength as constraints[27,28] and Bayesian inference of links based on edge data information[29,30]. In general terms, these methods use network-based metrics to infer the existence (or non-existence) of links, which significantly differs from predictive models based on meta-data attributes such as the population size or the distance between nodes.

A few models can be found in the literature where the node's attributes are used as complementary information to predict links. For instance, in the context of social contact networks, mobility patterns such as co-location of individuals was used to evaluate the probability of individuals to be connected in a social network[31], restaurant reviews were used to predict links in a taste similarity network[32], and the frequency of programming code commits was used to predict the mobility in cyberspace of projects[33]. The focus of our work is in the class of methods that can use node's attributes, such as urban metrics, as input data to fit models that can predict links and can be further extended to other data sets where the dependent variable is unknown.

Recently, an unprecedented amount of data related to human behavior and cities became available, from GPS tracking of mobile phones to census data of thousands of people[14,34–37]. These data promoted intense research on human mobility[1], including transportation networks[38], commuters networks[17], and network models of migration[39]. On the other hand, urban indicators were used to describe scaling in cities[3], to measure the performance of cities[40] and the similarity among different municipalities[41,42], and to describe crime-related phenomena[3,11,43]. However, a connection between human mobility and urban indicators, such as unemployment rate and Gross Domestic Product (GDP) is still missing. Understanding the influence of these indicators on the individual choices on daily commuting to work could help us to predict the flow of people between different areas and reconstruct the structure of the commuters network. Such a gap has led us to frame the question we want to answer: *how to quantify the number of people commuting in between cities taking into account a more comprehensive set of indicators*, *beyond distance and population*, *that might attract or repel commuters?*

In this study, we propose an alternative approach to deal with the reconstruction of commuters networks using supervised machine learning to understand the relationships between urban indicators and the structure of commuters networks. We perform an analysis based on 22 urban indicators of 5,565 Brazilian municipalities, together with the daily number of people commuting between every city in the data set. We show that the gravitation and radiation models have limited predictive power to classify whether there is a link between two cities or not and to quantify the flow (number of people commuting) between cities. In contrast, we show that predictions based on machine learning algorithms and urban indicators can reconstruct the commuters network with 90.4% of accuracy and describe 77.6% of the variance observed in the flow of people between cities. Further, we interpret the machine learning results using SHapley Additive exPlanations (SHAP) values[44] to quantify the importance of the urban indicators in predicting human mobility. We show that distance is a critical metric for predicting human mobility, but other indicators, like GDP and unemployment rate, also play a significant role in attracting/repelling people to a particular area. Our approach provides a better way to quantify human mobility and shed new lights on the role of urban indicators on commuting patterns. Because link-prediction and network reconstruction are still open challenges in network science, our results have implications in other areas, like economics, social sciences, and biology, where node attributes can give us information about the existence of links connecting network's entities.

Results

We started our study from the observation that people move from one city s to work in another city t, taking into consideration the advantages of living nearby or far away from where they work. In Fig. 1A each node represents a city that offers different opportunities (in terms of jobs), costs of living (number of houses available), and other factors, such as hospitals, schools, and urban parks, to name a few. Two given cities, s and t, are separated by a distance r_{st}, and people have a cost to commute from their home city to the city where they work. Existing migration models assume that the flow of people from city s to city t is proportional to their populations and decays with the distance. However, plenty of other factors play an important role when deciding where to live or work.

To quantify the number of people commuting in between cities taking into account a more comprehensive set of indicators, beyond distance and population, we collected data about the number of people commuting from one city to another (pendular migration), in 2010, considering all the 5,565 Brazilian cities, together with 22 urban indicators related to these municipalities. This data is provided and maintained by the Brazilian Institute of Geography and Statistics (IBGE)[45]. The data consists of the number of people that are living in a city s and commuting to work in a city t daily, and of urban indicators describing the cities in terms of population, labor, tuition rate, economy, sanitation, infrastructure, and violence, to name a few. We suppose that these indicators can be associated with a high or low number of commuters. Thus, less economically developed cities, usually those with fewer job opportunities, inferior infrastructure, and possibly higher indices of violence, would attract a lower number of people. On the other hand, those with a higher income, larger population, better social development, and infrastructure, might offer better job opportunities and would attract a higher number of people.

To investigate the flow of people commuting from one city to another, we assigned a node for each city and, for each non-zero flow, an edge with weight w_{st} equal to the number of people commuting from city s to t in the year 2010. Figure 1B illustrates the Brazilian commuters network using a kernel-based edge bundling technique[46]. To illustrate our urban indicators, in Fig. 1C, we present the Pearson correlation coefficient between the 22 urban indicators.

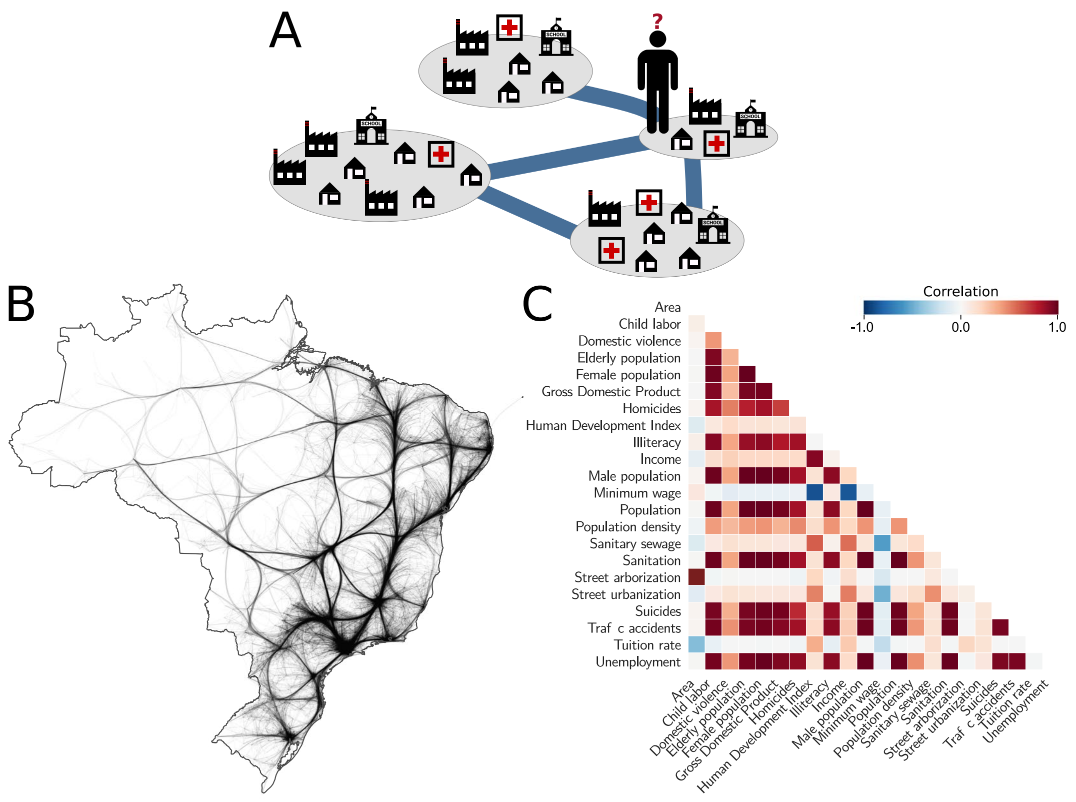


Figure 1. Commuters network and urban indicators related to human mobility. (**A**) Illustration of the work-mobility problem when people have to choose where to work based on distance, job opportunities, housing prices, and other variables related to urban systems. (**B**) Brazilian commuters network illustrated through a kernel-based Edge Bundling technique[46] where the thickness of the edges represents the flow intensity. (**C**) Correlation analysis of Brazilian urban indicators considering those that describe the population, labor, tuition rate, economy, sanitation, infrastructure, violence, and others. Reddish colors correspond to positively correlated indicators and bluish colors to negatively correlated ones.

State-of-the-art models. Next, we investigate the limitations of two models widely used to describe the flow of people between cities, namely, the gravitation model and the radiation model.

Gravitation model. Inspired by Newton's law of universal gravitation, it has been first used to predict the trading between nations; the model is defined as:

$$T_{st}^{GM} = C\frac{m_s^{\alpha} n_t^{\beta}}{r_{st}^{\gamma}}, \tag{1}$$

where m_s and n_t are the supply and demand forces of nations s and t, respectively[47]. This model was widely applied to predict population movement[14], economic relations between countries[48], cargo shipping volume[49], and long-distance phone calls[50]. In the context of migration, T_{st}^{GM} is the number of people commuting, s and t are cities, m_s and n_t are the population masses of these cities, and r_{st} is the distance between them. Thus, the interaction between two municipalities is directly proportional to the population masses and inversely proportional to the distance. Such a model depends on the parameters C, α, β, and γ, which can be estimated through Ordinary Least Squares (OLS) applying the logarithm operation on both sides of the formula. Although the gravitation model provides a good fit for a wide variety of scenarios, there are still some unsolved problems. For instance, Simini *et al.*[15] pointed out that:

- The model's equation lacks a rigorous mathematical derivation;
- The augmentation of the model is unrestricted, what would scale the number of parameters;
- The model fails when the data is not sufficient to estimate its parameters;
- The flow is only a function of population and distance, not including any other characteristic inherent to the cities and their neighbors;
- The number of commuters increases without limit as the population of the target city grows, becoming higher than the total population of the source city; and,
- The model is deterministic, being unable to estimate fluctuations in the number of people commuting.

Radiation model. Formulated in terms of the radiation and absorption processes from physics, this model was proposed to solve the problems inherent to the gravitation model[15]. It is based on a diffusion process in which an object in a specific location emits particles, having nearby objects with a different probability of absorbing these particles, which varies according to the forces acting upon them. In the context of commuters, the objects are the cities, and the particles are the people commuting. Thus, cities are radiating commuters, which are absorbed by neighboring cities.

In contrast to the gravitation model, in the radiation model, the distance between cities is not the major limitation to commuters; differently, the limitation arises from the supply and demand of commuters concerning the neighboring cities. The idea of supply and demand comes from the theory of intervening opportunities; this theory defines that mobility patterns are more influenced by the commuters' opportunity to establish in the target city than the distance to such a city[19]. The approximate number of intervening opportunities p_{st} is calculated according to the population of the source s and target t cities. It also takes into account the population of all the neighbors of the source city in a maximum radius of up to the distance to the target city. Thus, the radiation model can be defined as:

$$T_{st}^{Rad} = T_s \frac{m_s n_t}{(m_s + p_{st})(m_s + n_t + p_{st})}, \tag{2}$$

where m_s is the population of the source city, n_t the population of the target city, and p_{st} is the total population in the circle of radius r_{st} centered at city s.

The only parameter of the radiation model, T_s, is the scaling factor of the outgoing number of commuters from the source city. The value of T_s can be estimated by adjusting the equation $T_s = \alpha m_s$ to the data using OLS, in which m_s is the population of the source city and α is the intersection point of the fitting between the population size and the number of commuters from every city in the whole data set. The major limitation of the model is that T_s depends on the number of commuters. Thus, the model requires the correct information (or an estimation) about the number of people commuting between cities, which makes it difficult to generalize the model to unseen data.

Similarly, Ren *et al.*[17] proposed the cost-based radiation model to predict the flow of commuters in spatial networks. Although the model has been reported as more efficient than its predecessor, its predictive ability is still limited to cases where there is information about the existence of links. As a consequence, the radiation model and its cost-based version are not able to reconstruct the structure of the commuters network.

It is also worth to mention that most of these models (gravitation and radiation models) were validated through correlation analysis, rather than using the variance between the predicted and real values; this fact causes an overestimation of the capability of the model in predicting the flow of people between cities. The variance is more error-sensitive than the correlation coefficient (*e.g.*, Pearson, Spearman, or Kendall). Moreover, when assessing the results, some authors take into account only existing edges to compare with the predictions, causing, once more, an optimistic estimation of their predictive performance.

Next, to further confirm our claims, we applied the gravitation and radiation models to predict the number of people in the Brazilian commuters network. Our first observation is that these models (Eqs 1 and 2) are not able to reconstruct the unweighted projection of the commuters network structure, because any pair of cities with non-zero population would have an edge, resulting in a fully connected network. If one considers faraway cities, it is possible to have flow close to zero, but for typical distances between Brazilian cities (average value of ~1,000 *km*), we would still find a non-zero flow. The resulting networks generated by these models are far from the sparse networks observed in real data. Thus, we quantified the number of commuters considering only edges that we know beforehand to have a non-zero flow. We considered the distance to be the length of the geodesic path (in kilometers) between two given cities, which is calculated from their coordinates provided by the Brazilian census data[45].

Our evaluation started by fitting the models (Eqs 1 and 2) to the data through OLS to estimate their descriptive parameters. Next, we evaluated the results using R^2-score (coefficient of determination) and Pearson correlation coefficient. The R^2-score measures the variance of a dependent variable that was predicted using independent variables; the metric is defined in the range $]-\infty, \ 1]$. An R^2-score closer to 1, means that the variance of the prediction concerning the real value is low[51]. Notice that this metric can be negative since the model can be arbitrarily wrong. It is worth mentioning that, along with the text, we express the R^2-score through percentage (*e.g.*, 14% instead of 0.14) because we use it to discuss the proportion of variance explained by the correlation between the predicted and observed flow. The Pearson correlation coefficient ρ, in turn, measures the linearity of two variables regarding each other. The metric is defined in the range $[-1, \ 1]$, in which -1 indicates a perfect negative correlation, 0 indicates no linear correlation and 1 indicates a perfect positive correlation[52]. With very wrong predictions, one could find results that are very correlated with the empirical values ($\rho \approx 1$), misleading the evaluation of the predictive model performance.

Figure 2 presents the results related to the fitting of the gravitation and radiation models to our data. From this figure, we verify that the values predicted by both models are positively correlated (values greater than 0.60) with the real data, but the R^2-score is meaningless regardless of the model (values below 0.25). Despite the good correlation, the models were not able to correctly predict the data, and the values differ sharply from the real ones. The same conclusion was brought by Masucci *et al.*[16], who analyzed the ability of generalization and universality of both models with a data set of commuters in the surroundings of London, UK. In agreement with what we observed when evaluating the model using the Brazilian data set, neither model could satisfactorily fit their data. Thus, we confirmed that the gravitation and radiation models could not correctly describe the mobility pattern of the intercity commuters network.

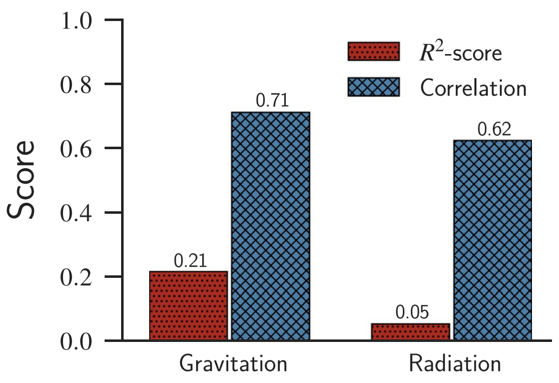


Figure 2. Evaluating the predictive ability of the gravitation and radiation models. We used two metrics, the coefficient of determination (referred to as R^2-score), and the Pearson correlation coefficient; both metrics consider the predicted flow and the observed flow of the models. Although the results show a linear correlation between the predicted and observed flows, the R^2-score reveals that the predicted values are still very noisy when compared to the real ones, suggesting that neither model is accurate enough in reconstructing the commuters network.

Alternative modeling using machine learning. Next, we propose the use of predictive models induced by machine learning algorithms to predict the number of intercity commuters and to reconstruct the structure of the corresponding commuters network. Our approach differs from the regular link prediction because it does not take into account any network-based topology metric as an independent variable. Instead, we reconstruct the network based on the distance r between two cities, their populations' size and a set of urban indicators related to each city. To do so, we investigate two predictive tasks:

- Classification: induce a binary classifier able to predict whether a link between a pair of cities exists or not;
- Regression: induce a regressor able to predict the number of commuters between a pair of cities.

Reconstructing the commuters network structure using machine-learning classification. In our context, a binary classifier predicts whether a link between a city s and a city t exists or not based on the distance r_{st}, populations sizes m_s and n_t, and more 21 urban indicators from each city, $U_i = \{u_{i_0}, \ldots, u_{i_{20}}\}$, $i \in \{s, t\}$, and $u_{i_j} \in \mathbb{R}$ for $0 \leq j \leq 20$, resulting in a total of 45 urban indicators (referred to as features). That is, given an ordered pair $\langle city_s, city_t\rangle$ whose urban indicators define a set $S_{st} = \{r_{st}, m_s, n_t, U_s, U_t\}$, $S_{st} \in \mathbb{R}^{|S|}$, a binary classifier refers to a function

$$Class: \mathbb{R}^{|S|} \rightarrow \{0, 1\} \tag{3}$$

in which, 0 indicates no link between s and t, and 1 indicates the opposite.

To find the best classifier, we first employed the holdout approach, splitting the data into 70% for training and 30% for testing. Then, we sampled the training data using stratified k-fold cross-validation, with $k = 5$. The training data was used in the model selection, feature selection, and hyperparameter tuning. The test set was used only in the final step, after the hyperparameter tuning, to evaluate the predictive performance, generalization, and universality of the model in an unseen data set.

Subsequently, we tested 34 classification algorithms with default hyperparameter values from the scikit-learn[53] and eXtreme Gradient Boosting (XGBoost)[54] libraries, from which only 27 were able to fit the data and provide the resulting accuracy. The 7 removed algorithms (see Supplementary Table 1) fail to fit the data (*i.e.*, to converge) without a pre-tuning of hyperparameters, which is intentionally not covered by our methodology. We do not perform the pre-tuning of hyperparameter because it would exponentially increase the processing time of the experiments' pipeline with no improvement guaranteed.

Following, to select the best among the 27 remaining classifiers, we used bootstrapping sampling to evaluate their predictive performance following an accuracy-based perspective. This procedure consists in feeding the model with randomly selected samples to assess its variance. Figure 3A shows the results of the classifiers that passed our first test. The algorithms were sorted in ascending order according to their predictive accuracy. It is worth to mention that the best algorithms (rightmost) are the CatBoost, XGBoost, and Light Gradient Boosting Machine (LGBM), and the ones with the worst performance are the Perceptron, Passive Aggressive Classifier, and Gaussian Naive Bayes (NB). The average accuracy score in the experiment varies from 50.2% in the worst case to 87.9% in the best case — see Supplementary Table 3. Notice that, the algorithm with the highest median score, lowest variance, and fewer outliers is the CatBoost. However, the difference between the CatBoost (first-placed) and XGBoost (second-placed) is irrelevant (see Supplementary Table 3), and the XGBoost is around fifty times faster per train iteration than the CatBoost. For this reason, we have chosen the XGBoost as the best algorithm for this problem.

Following, we evaluated the learning curve considering only the best classifier (*i.e.*, XGBoost), which showed an average accuracy score of 87.5% on the training set, with little variance and no significant accuracy outliers. The learning curve assesses the model predictability by varying the size of the training set. The curve in Fig. 3B

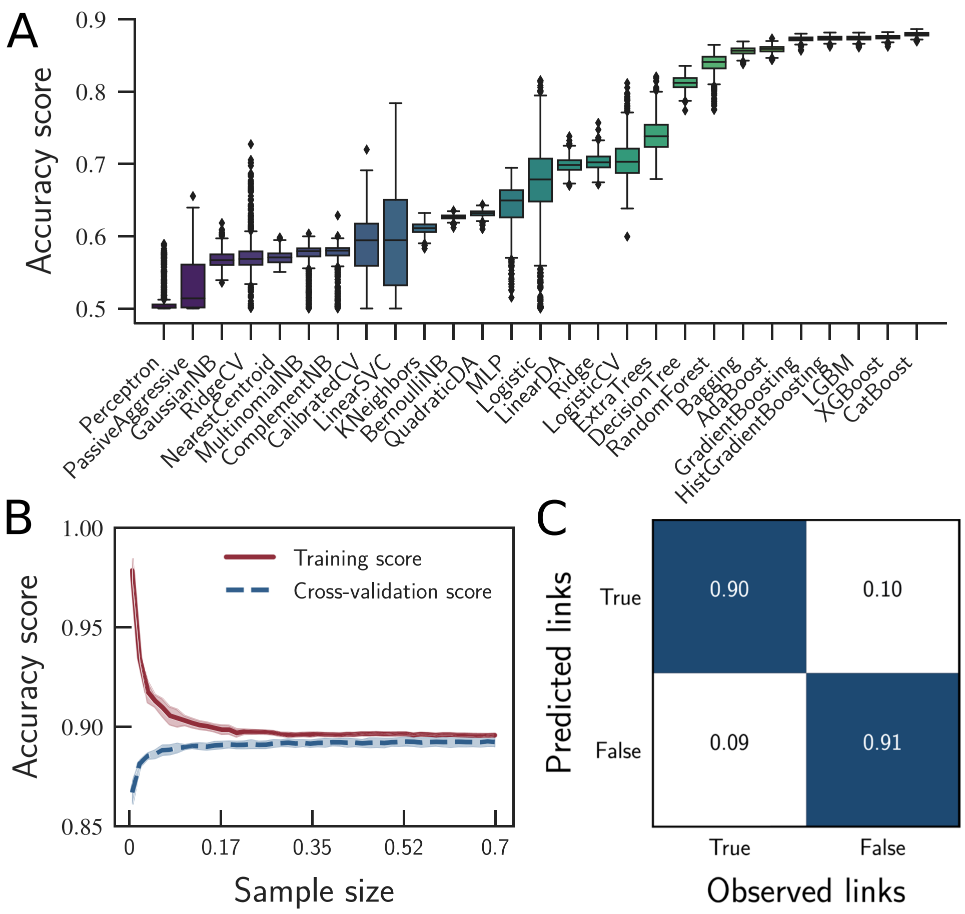


Figure 3. Performance of the classification algorithms in reconstructing the unweighted projection of the commuters network; see Supplementary Table 1 for a list of the classifiers' acronyms. (**A**) The accuracy of the classification algorithms in determining whether a link between a given pair of cities exists or not. (**B**) XGBoost learning curves varying the sample size. (**C**) XGBoost confusion matrix after hyperparameter tuning, showing an overall accuracy of 90.4%. Specifically, the results showed 90% of true positives, 10% of false negatives, 9% of false positives, and 91% of true negatives.

shows that 89% of the training set is the right amount of data required to train the classifier and a classifier trained up to such amount of data yields an accuracy of about 89.2%.

To reduce the computational cost in the induction of the predictive model using XGBoost, we used a threshold-based feature selection (see Methods), measuring the accuracy of the model by progressively removing features. We considered the degree of importance of a feature as the number of times the feature is used to split a node into two trees in the XGBoost algorithm. By doing so, we were able to remove 24 predictive features without reducing the model's predictive accuracy (see Supplementary Table 5).

Finally, we used the test set (the remaining 30% of the data previously not used) to test the induced model. We first calculated the predictive accuracy of our classifier on the test data using the model with no grid-search optimization, finding 89.3% of accuracy. The grid-search was able to improve this value by 1.1% (see Methods), which, in a scenario with thousands of cities with millions of possibilities of commuters flowing between them, means more hundreds of thousands of correctly predicted links. Therefore, such an improvement resulted in an overall accuracy of 90.4%, as shown in the confusion matrix in Fig. 3C.

Reconstructing the weighted commuters network using machine-learning regression. Deeper in the analytical scenario, we are interested not only in knowing whether a link exists or not but also in the weight w_{st} of each link, which represents the flow of people from city s to t. That is, given an edge $\langle city_s, city_t\rangle$ and its corresponding set of indicators $S_{st} = \{r_{st}, m_s, n_t, U_s, U_t\}, S_{st} \in \mathbb{R}^{|S|}$, we seek a function

$$Weigh: \mathbb{R}^{|S|} \to \mathbb{N} \tag{4}$$

where *Weight* predicts the number of commuters between cities s and t.

Similarly to the prediction of links, we first used holdout to split the data into 70% for training and 30% for testing. Subsequently, the training set was used for 5-fold cross-validation, model selection, feature selection, and hyperparameter tuning, and the test set to evaluate our model in an unseen data set.

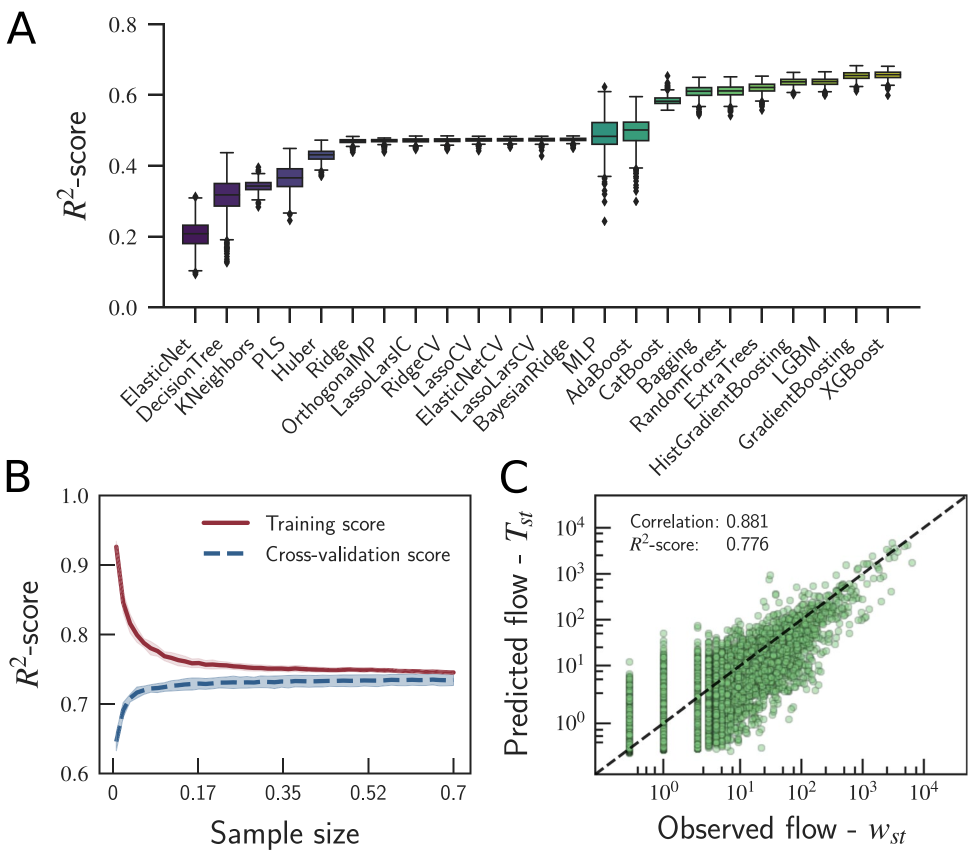


Figure 4. Performance of the regression algorithms in reconstructing the weighted projection of the commuters network; see Supplementary Table 2 for a list of the regressors' acronyms. (**A**) R^2-score of the regressors' performance in quantifying the number of people (*i.e.*, *Weight$_{st}$*) commuting from city *s* to city *t*. (**B**) XGBoost learning curves varying the sample size. (**C**) Predictions of *Weight$_{st}$* made by the XGBoost regressor, after hyperparameter tuning, compared with the actual flow w_{st}. Specifically, our model as able to achieve an R^2-score of 77.6% and a Pearson correlation coefficient of 0.881.

We used bootstrapping sampling to assess the variance of different regressors in terms of the R^2-score. From the 44 models, 21 were not able to provide all values (including outliers) of R^2-score higher than 0 (see Supplementary Table 2). Figure 4A shows the results obtained from the 23 models that passed the first test, all of which were tested with default hyperparameter values from the scikit-learn[53] and XGBoost[54] libraries. The models are presented in ascending order according to the value of their R^2-score. The results reveal that the three best algorithms are XGBoost, Gradient Boosting, and LGBM, and the three worst are Elastic-Net, Decision Tree, and K-Nearest Neighbors. The average R^2-score in the test varies from 20.6% in the worst case (*i.e.*, Elastic-Net) to 65.6% in the best case (*i.e.*, XGBoost), see Supplementary Table 4.

From now on, our focus will be on the best regression model, which was induced by XGBoost with an average R^2-score of 65.6%, with no significant variance after a thousand predictions using different random samples of the training set. The further reason to choose the XGBoost as the best algorithm is that it is fast, as it provides a parallel tree boosting that solves problems faster than its competitors and has support to solve problems on high-performance computing environments. Thus, we evaluated the learning curve of the XGBoost regressor (see Fig. 4B), assessing the model predictability by varying the size of the training set, and we found that 100% of the training data is required to train the regressor, yielding an R^2-score of 73.4%. One can see that the regressors' predictions are more challenging than those of the binary classification algorithms, requiring more data during the training phase and yielding more errors.

Finally, we used the 30% data reserved for testing the model. The results indicated that the regressor was able to predict 73.1% of the variance observed in the data. After the hyperparameter tuning (see Methods), the XGBoost regressor described 77.6% of the variance. Although the gains seem to be small, our model is intended to be used on data sets of thousands of cities, which answer for millions of possibilities of commuters flowing between pairs of cities. In such a case, 4.5% improvement means weights closer to the real ones among the actual links. Figure 5C compares the predictions about the number of commuters (*Weight$_{st}$*) with the actual values (w_{st}) observed in the data. Notice that, our model can recover the weighted network structure and predict the number of commuters with much higher precision than the gravitation and radiation models. In our case, even considering links with $w_{st} = 0$, our predictions are much more accurate than those from the gravitation and radiation models (see Fig. 2), which were induced using only existing flows.

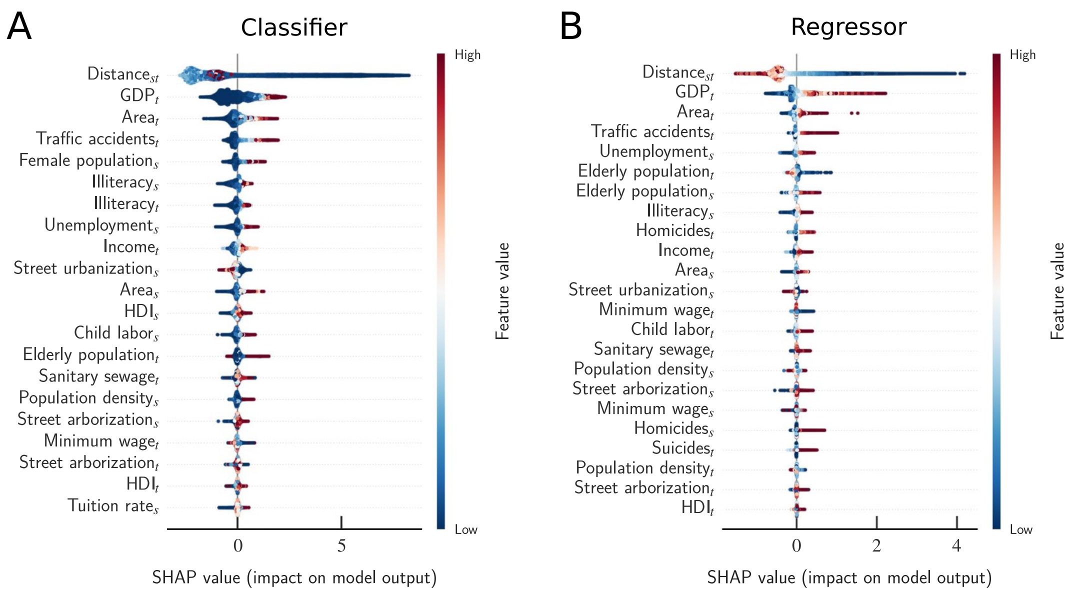


Figure 5. Interpreting the relationship between flow and features. Analysis of feature importance in the XGBoost (**A**) classifier and (**B**) regressor using SHAP values. The features are ranked by importance in descending order based on the sum of the SHAP values over all the samples. The violin-shaped plots show the distribution of the impacts of each feature on every point of the model output. The colors represent the SHAP value, varying from low values (blueish colors) to high values (reddish colors).

Interpreting machine learning models using SHapley additive exPlanations (SHAP). In contrast with the gravitation and radiation models, the proposed models can accurately reproduce the structure of the network, predicting whether a link between a given pair of cities exists or not (XGBoost classification) and accurately estimate the number of commuters between two cities (XGBoost regressor), reconstructing the weighted structure of the commuters network. The price of having a more complex model is that it becomes harder to interpret the relationship between the independent variables and our predictor. However, if we could understand how the algorithm makes decisions, the decision process would be a valuable resource to learn about the equations that describe our systems. Thus, instead of assuming mathematical relationships and trying to fit the model to the data, we could look into the results and try to figure out what the data tell us about the model that describes our system and the relationships between the commuter's flow and our features.

To turn our black-box algorithm in a more interpretative model, we calculated the features' importance and the impact of the features on individual predictions using the SHapley Additive exPlanations (SHAP) values[44,55,56]. The SHAP metric is based on the Shapley values introduced by Lloyd Shapley in 1953 in the context of game theory[57]; it helps one to understand how the model decides to make a prediction and which features contribute to improving the accuracy of the model. The course of action of SHAP is to calculate the importance of a feature by comparing what the model predicts with and without the feature. Notice that the order in which we add new features to the model can affect its predictions. Thus, we have to permute over all the possible feature orderings to fully capture the impact of a feature on the model.

In Fig. 5A, we show the features considered important for the XGBoost classifier, as well as the distribution of the impacts of each feature on the model output. While high values of distance $Distance_{st}$ (also known as r_{st}) are not good predictive features for the presence of links (values smaller than 0), high GDP_t values increase the predictive accuracy of the model (values larger than 0). High rates of the elderly population in the target city also strongly affect the model accuracy. Finally, the total urban area or traffic accidents are more important than the population density.

In Fig. 5B, we show the features' importance for the XGBoost regressor, as well as the distribution of the impacts of each feature on the model output. In this case, high values of distance $Distance_{st}$ have a low impact on the predictive performance of the model (values smaller than 0), whereas high GDP_t increases the predictive performance of the model (values larger than 0). Higher values of the elderly population in the target city decrease the performance of the algorithm. Again, we verify that several urban indicators have a role in defining the flow of people from one city to others and that urban indicators, such like GDP, area, and traffic accidents, are more useful to improve the predictive performance of the model than population size.

Discussion

Predicting decisions related to individual choices, such as housing and working places, is a difficult task. These decisions are tied to many variables that are often based on personal reasons and cannot be easily measured. However, the analysis of SHAP values can help us to understand how urban indicators and distance can influence this decision process and what makes people commute from one area to another to work. The SHAP values point four features that are mutually important in both classification and regression models, *i.e*., Distance, GDP, Area, Traffic accidents in the target city, followed by other less important features.

Distance is the most important feature in predicting the existence of a link between a pair of cities and quantifying the number of people commuting from one city to the other. It is traditionally included in migration models. Distant cities make the journey to work unfeasible, as workers must return home by the end of the day, limiting the commuting to a certain distance. However, because of other factors, cities close to each other may have an insignificant flow, either because they offer similar opportunities or because the infrastructure and transport limit the commuting.

GDP (Gross Domestic Product) is the second most prominent feature. The cities with the highest GDP have the highest demand for commuters and the best job opportunities. As a consequence, they attract more commuters. Looking at Fig. 5, it is possible to observe that the higher the GDP value, the higher its contribution to the predictive performance of the XGBoost induced models. Another possible explanation is that these cities are more productive because they attract more workers and, therefore, they have higher GDP values, as suggested by Keuschnigg *et al*.[58] for long-term migration.

Area is related to the physical size of a city, and it is the third most influential variable. Larger cities, like metropolis and megalopolis, usually are hubs of industries and enterprises and, consequently, tend to offer more job opportunities. This variable fails to affect the flow prediction when cities have a wide territorial extension but are used for other purposes (*e.g*., natural reserves, forest, and huge urban parks) rather than urban expansion. Our data set reveals just a few records of cities occupying a large area with no significant number of commuters. These are the cases of cities with SHAP values close to 0, as we can observe in Fig. 5B.

Traffic accident is the fourth most important variable. This indicator was selected by the model because it has a high correlation with a higher flow of people and automobile vehicles (see Fig. 1C), which is also a trait related to the economics of the city, as is the case of GDP. Recall that, although the features we used are correlated, the feature selection process chose to keep the ones that it considers to be important, removing the other ones without negatively impact the predictive performance of the model (classifier and regressor).

The following features are ordered differently, depending on the predictive task, whether it is classification or regression. In the classification task, the classifier does not require much to predict a fair number of links correctly. For example, using just the previously discussed selected features on the training set, the classifier presented an accuracy close to 87.6%. The other features improved the model accuracy by 1.6%, also on the training set. Among all variables, the *unemployment* indicator of the source city (*i.e*., where people live), seems to have a significant impact on the existence of flow between two cities. High rates of unemployment seem to make commuters leave the city where they live (*i.e*., repel) to work in nearby cities with better job opportunities (*i.e*., absorb).

Differently from the classification task, in the regression task, the algorithm demands more information to estimate the value of flow close to the actual value. Using only the four most important features, the model describes 70.6% of the variance of the data on the training set. The remaining features are responsible for improving the regressor's predictive performance by 2.8% on the training set. The *elderly population* indicator has an interesting behavior: the larger the number of elderly people in the home city and the smaller their number in the city of work, the higher the flow of commuters between the two.

The remaining urban indicators showed little relevance in predicting the flow of people, despite having a significant impact on the predictions of the classification and regression tasks. This set of features together provides a better prediction and highlights the importance of taking into account a more comprehensive set of indicators, in addition to the typical metrics used in migration models, such as population and distance. Further, such models could be enhanced by adding other indicators that were not explored in our data set, which could potentially improve the predictions of links and flow of commuters.

Finally, we believe that our approach is not only to be used on the task of commuters network reconstruction but also to complex networks of different domains where node's attributes can give us information about the existence of links connecting them. For instance, this methodology could be applied in economic trade networks[59,60], where the amount of money exchanged could be predicted in terms of indicators as GDP, unemployment, interest rate, production, corporate profits, and other macroeconomic variables; in social networks[61], where friendship connections could be unveiled by individual node's preferences, such as music or sport preferences, language or individual socioeconomic status and education; in metabolic networks[62], where biochemical reactions (links) could be predicted in terms of metabolites chemical, physical, and biological properties. Thus, further insights from these networks could be obtained by exploring the SHAP values to identify decisive indicators to reconstruct each network topology.

Methods

Data set. We collected data about the number of people commuting from a home city to another city to work (pendular migration) in the year of 2010, considering all the 5,565 Brazilian cities as well as 22 yearly-updated urban indicators. This data is provided and maintained by the Brazilian Institute of Geography and Statistics (IBGE)[45]. The data was modeled as a complex network represented as a directed graph $G = \{V, E\}$ composed of 5,565 vertices and 55,247 edges with non-zero weights. A vertex $s \in V$ is considered to be a city, and an edge $e \in E$ is an ordered pair $e = \langle s, t \rangle$ representing the flow from the source city $s \in V$ to the target city $t \in V$. The existence of commuters from city s to city t is noted as $Class_{st} = 1$, and $Class_{st} = 0$ indicates the opposite; the flow $Weight_{st}$ is an integer representing the number of commuters that move daily from city s to city t. The urban indicators shape an initial 45-value feature vector. This vector is composed of urban indicators (a single value per indicator) from the source and target cities (22 values for each city) and their distance (in kilometers).

Data split. The data set composed by the number of commuters (target feature) and the 2×22 urban indicators plus the distance between pairs of cities (predictive features) was split in a 70/30 ratio preserving the ratios of existing and non-existing links in both sets. Thus, 70% was used as a training set, and the other 30% was used as a test set to evaluate the predictive performance, generalization, and universality of the model induced after all training and selection steps. To assure that the model could learn all inherent nuances of our data (*e.g.*, different intercity road systems), we also stratified the data by the Federal Brazilian States to feed the model with a proportional amount of information about every state. This also saves computational time to train the models by eliminating most of the non-existent links across states.

Class balancing. Because imbalanced data sets favor the majority class, we balanced the classes of the resulting data set keeping 50% of the data labeled with existing links, $w_{st} \neq 0$, and the other with non-existing links $w_{st} = 0$, resulting in 110,494 pairs of cities (edges). For each existing connection between a city from state A and another from state B, our data set has a non-existing link between cities (chosen at random) from these states. This process was carried out for both tasks, classification, and regression. In the classification, the labels are $Class_{st} = 1$ if there is a non-zero flow and $Class_{st} = 0$ otherwise, whereas, in the regression task, the data was labeled with the number of commuters in the flow from one city to the other.

Training set stratification. We used a label-based stratified *k*-fold with 5 folds for the classification tasks (categorical data) and a regular *k*-fold with 5 folds for the regression tasks (number of commuters).

Model selection. Using the training set, which represents 70% of the whole data set and is composed of 77,345 edges, we tested 78 algorithms of machine learning, including 34 classifiers and 44 regressors. In this step, we used the validation subset of the training data to select the best classifier to predict the existence of a non-zero flow (link) between cities and the best regressor to predict the number of commuters flowing between them (links' weight). This task was carried out using Bootstrapping, which is a statistical resampling method that consists of sampling with replacement to define the upper and lower bound of a statistical evaluation metric (accuracy and R^2-score)[63]. We used this method to evaluate different models over random samples of the data set, without compromising the significance of the results. From all classifiers and regressors, we only used those that could yield a predictive performance without needing early hyperparameter tuning. Almost one-third of all algorithms did not meet such constraints, leaving 27 classifiers and 23 regressors for the experiments. The remaining algorithms went through a bootstrapping sampling by training each one with tiny random samples (1% of the data chosen with replacement, *i.e.*, 7,734 edges) of the training set (70% of the whole data set, *i.e.*, 77,345 edges) to access the variance of predictions over a thousand iterations. Through the bootstrapping sampling results, we defined the upper and lower bound of the scoring metric (accuracy and R^2-score). Using these metrics as criteria, we select the classifier and regressor with the highest median, lowest variance, and fewer outliers. It is important to note that bootstrap sampling uses random samples of 1% of the training set to increase the randomness within each test sample so to capture all nuances related to the variance and outliers among thousands of predictions.

eXtreme gradient boosting (XGBoost). XGBoost is a machine learning library that provides a set of techniques to deal with and model data through both classifiers and regressors. Such a library was designed to be highly efficient, flexible, and portable. All the model-inferring algorithms were implemented under the Gradient Boosting framework providing further support to parallel and distributed processing[54]. Gradient Boosting, on the other hand, renders a predictive model through tree ensembles, which are submitted to optimization through an arbitrary differentiable loss function[64].

Learning curves. We used analysis of learning curves to assess the prediction capability of each model (*i.e.*, XGBoost classifier, and regressor) when increasing the size of the training data in increments of 1% up to 70%. The right amount of data to train each model is the one with the best trade-off between bias and variance, that is, where the training score curve and the cross-validation curve have the lowest deviation.

Feature selection. We performed a threshold-based feature selection to decrease the complexity of the induced model by removing features from the least important to the most important one up to a dynamically-defined importance-threshold. Such a process provides a subtler interpretation of the features' impact on the model's output and is carried out together with the XGBoost algorithm, which provides an importance score to each feature during the training phase. The importance score is defined in the interval $[0, 1]$, where 0 indicates a feature with minimum or no importance, and 1 indicates the opposite. The importance score consists of counting the number of times each feature is used to split a node into two trees by the XGBoost algorithm during the training phase. The feature selection process consists of measuring the model's prediction score while progressively removing the features from the least important to the most important one. These iterations define a threshold that splits the interval into two, such that features with importance between 0 and the threshold are removed. The threshold is defined by iteratively increasing the value from 0 up to 1 until the model's prediction score using all features starts decreasing. It is important to note that the features removed are not relevant to the induced model because they do not add useful information to predict the target data. This comes from the fact that urban indicators are collinear to each other, which means each indicator might contain traces of other indicators within it. Bettencourt[3] already discussed such phenomena on urban indicators, however, there still no way to avoid or remove it completely from such type of data. The threshold-based feature selection works around this problem by removing non-relevant and collinear features, which absence impacts the model nor positively neither negatively. Therefore, the benefit of using feature selection, in this case, is its ability to reduce the model complexity by decreasing the number of dependable variables.

Hyperparameter tuning. For XGBoost classifier and regressor, we performed hyperparameter tuning; a brute-force technique that seeks for the set of hyperparameters within a pre-defined hyperparameters' sub-space that optimizes a provided machine learning model. For the sake of experimentation, we tested 3 million possibilities out of a hyperparameters space that is exponentially larger. Such a fine-tuning is considerably time-consuming and can take several weeks of intensive computing for each model. On the other hand, the time is proportional to the number of tested hyperparameters, which is a fair trade-off, and that can be decreased with high-performance computing.

For our experiments, we found that the best hyperparameter values for the XGBoost classifier were: *(i)* learning rate = 0.05; *(ii)* number of trees = 300; *(iii)* maximum tree depth = 9; *(iv)* fraction of observations used as random samples by each tree = 0.85; *(v)* subsample ratio of features when constructing each tree = 0.8; and, *(vi)* regularization term of the model's weights = 0.75. In the case of the XGBoost regressor, the best hyperparameter values were: *(i)* learning rate = 0.05; *(ii)* number of trees = 900; *(iii)* maximum tree depth = 7; *(iv)* fraction of observations used as random samples by each tree = 0.85; *(v)* regularization term of the model's weights = 1.25; *(vi)* minimum loss reduction required to split a node into two trees = 0.25; and, *(vii)* minimum weight needed in a child node = 2.

Model evaluation. After all the training and selection steps, we used the testing set to compare the non-optimized and optimized models, reporting the results in a confusion matrix for the classifier and a scatter plot for the regressor. The classifiers were evaluated through the balanced accuracy score (*i.e.*, the ratio between the correct classified instances and all instances, normalized by the number of elements per label) and the regressors through the R^2-score (*i.e.*, the variance between the predicted and observed values).

SHAP values and feature importance. Given a specific prediction $f(x)$, we calculate the importance of a feature by comparing what a model predicts with and without the feature. Because the order in which we add new features to the model can affect its predictions, we permuted over all possible feature ordering. Mathematically, the Shapley value for a particular feature i (out of M total features), given a prediction x is:

$$\phi_i(f, x) = \sum_{S \subseteq M \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!}[f_x(S \cup \{i\}) - f_x(S)], \tag{5}$$

where S is the set of features used to induce the model, and f_x is the prediction considering the indicated set of features[55]. In practice, this is too difficult to be calculated because there are far too many possible combinations. Instead, we used the SHAP library to calculate ϕ_i, which is optimized to take advantage of different model's structures[44]. Thus, the rank of feature importance is given by the sum of SHAP value magnitudes ϕ_i over all samples. This procedure allows a more in-depth interpretation of the impact of the features on the prediction of individual data, turning the black-box algorithm in a more interpretable model.

References

1. Barbosa, H. *et al.* Human mobility: Models and applications. *Physics Reports* **734**, 1–74, https://doi.org/10.1016/j.physrep.2018.01.001 (2018).
2. Ullman, E. L. *Geography as spatial interaction* (University of Washington Press, 1980).
3. Bettencourt, L. M. A., Lobo, J., Strumsky, D. & West, G. B. Urban scaling and its deviations: Revealing the structure of wealth, innovation and crime across cities. *PLoS One* **5**, 1–9, https://doi.org/10.1371/journal.pone.0013541 (2010).
4. Louf, R. & Barthelemy, M. Modeling the polycentric pransition of cities. *Physical Review Letters* **111**, 198702, https://doi.org/10.1103/PhysRevLett.111.198702 (2013).
5. Trenchard, H. & Perc, M. Energy saving mechanisms, collective behavior and the variation range hypothesis in biological systems: A review. *Biosystems* **147**, 40–66, https://doi.org/10.1016/j.biosystems.2016.05.010 (2016).
6. Helbing, D. *et al.* Saving human lives: What complexity science and information systems can contribute. *Journal of Statistical Physics* **158**, 735–781, https://doi.org/10.1007/s10955-014-1024-9 (2015).
7. Eubank, S. *et al.* Modelling disease outbreaks in realistic urban social networks. *Nature* **429**, 180, https://doi.org/10.1038/nature02541 (2004).
8. Colizza, V., Barrat, A., Barthélemy, M. & Vespignani, A. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences* **103**, 2015–2020, https://doi.org/10.1073/pnas.0510525103 (2006).
9. Balcan, D. *et al.* Multiscale mobility networks and the spatial spreading of infectious diseases. *Proceedings of the National Academy of Sciences* **106**, 21484–21489, https://doi.org/10.1073/pnas.0906910106 (2009).
10. Yang, H.-X., Tang, M. & Wang, Z. Suppressing epidemic spreading by risk-averse migration in dynamical networks. *Physica A: Statistical Mechanics and its Applications* **490**, 347–352, https://doi.org/10.1016/j.physa.2017.08.067 (2018).
11. Caminha, C. *et al.* Human mobility in large cities as a proxy for crime. *PLoS One* **12**, 1–13, https://doi.org/10.1371/journal.pone.0171609 (2017).
12. Spadon, G. *et al.* Behavioral characterization of criminality spread in cities. vol. 108, 2537–2541, https://doi.org/10.1016/j.procs.2017.05.118, International Conference on Computational Science, ICCS 2017, 12–14 June 2017, Zurich, Switzerland (2017).
13. Zipf, G. K. The P1P2/D hypothesis: on the intercity movement of persons. *American Sociological Review* **11**, 677–686, https://doi.org/10.2307/2087063 (1946).
14. Jung, W.-S., Wang, F. & Stanley, H. E. Gravity model in the Korean highway. *EPL (Europhysics Letters)* **81**, 48005, https://doi.org/10.1209/0295-5075/81/48005 (2008).
15. Simini, F., González, M. C., Maritan, A. & Barabási, A.-L. A universal model for mobility and migration patterns. *Nature* **484**, 96, https://doi.org/10.1038/nature10856 (2012).
16. Masucci, A. P., Serras, J., Johansson, A. & Batty, M. Gravity versus radiation models: On the importance of scale and heterogeneity in commuting flows. *Physical Review E* **88**, 022812, https://doi.org/10.1103/PhysRevE.88.022812 (2013).
17. Ren, Y., Ercsey-Ravasz, M., Wang, P., González, M. C. & Toroczkai, Z. Predicting commuter flows in spatial networks using a radiation model based on temporal ranges. *Nature Communications* **5**, 5347, https://doi.org/10.1038/ncomms6347 (2014).
18. Ravenstein, E. G. The laws of migration. *Journal of the Statistical Society of London* **48**, 167–235, https://doi.org/10.2307/2979181 (1885).

19. Stouffer, S. A. Intervening opportunities: A theory relating mobility and distance. *American Sociological Review* **5**, 845–867, https://doi.org/10.2307/2084520 (1940).
20. Louail, T. *et al.* Uncovering the spatial structure of mobility networks. *Nature Communications* **6**, 6007, https://doi.org/10.1038/ncomms7007 (2015).
21. Lü, L. & Zhou, T. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications* **390**, 1150–1170, https://doi.org/10.1016/j.physa.2010.11.027 (2011).
22. Newman, M. E. Clustering and preferential attachment in growing networks. *Physical Review E* **64**, 025102, https://doi.org/10.1103/PhysRevE.64.025102 (2001).
23. Kossinets, G. Effects of missing data in social networks. *Social Networks* **28**, 247–268, https://doi.org/10.1016/j.socnet.2005.07.002 (2006).
24. Jeh, G. & Widom, J. Simrank: a measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 538–543, https://doi.org/10.1145/775047.775126 (ACM, 2002).
25. Fu, C. *et al.* Link weight prediction using supervised learning methods and its application to yelp layered network. *IEEE Transactions on Knowledge and Data Engineering* **30**, 1507–1518, https://doi.org/10.1109/TKDE.2018.2801854 (2018).
26. Clauset, A., Moore, C. & Newman, M. E. Hierarchical structure and the prediction of missing links in networks. *Nature* **453**, 98, https://doi.org/10.1038/nature06830 (2008).
27. Mastrandrea, R., Squartini, T., Fagiolo, G. & Garlaschelli, D. Enhanced reconstruction of weighted networks from strengths and degrees. *New Journal of Physics* **16**, 043022, https://doi.org/10.1088/1367-2630/16/4/043022 (2014).
28. Squartini, T., Mastrandrea, R. & Garlaschelli, D. Unbiased sampling of network ensembles. *New Journal of Physics* **17**, 023052, https://doi.org/10.1088/1367-2630/17/2/023052 (2015).
29. Guimerà, R. & Sales-Pardo, M. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences* **106**, 22073–22078, https://doi.org/10.1073/pnas.0908366106 (2009).
30. Peixoto, T. P. Reconstructing networks with unknown and heterogeneous errors. *Physical Review X* **8**, 041011, https://doi.org/10.1103/PhysRevX.8.041011 (2018).
31. Wang, D., Pedreschi, D., Song, C., Giannotti, F. & Barabasi, A.-L. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, 1100–1108, https://doi.org/10.1145/2020408.2020581 (ACM, New York, NY, USA, 2011).
32. Xuan, Q. *et al.* Modern food foraging patterns: Geography and cuisine choices of restaurant patrons on yelp. *IEEE Transactions on Computational Social Systems* **5**, 508–517 (2018).
33. Xuan, Q., Okano, A., Devanbu, P. & Filkov, V. Focus-shifting patterns of oss developers and their congruence with call graphs. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, 401–412, https://doi.org/10.1145/2635868.2635914 (ACM, New York, NY, USA, 2014).
34. Makse, H. A., Havlin, S. & Stanley, H. E. Modelling urban growth patterns. *Nature* **377**, 608, https://doi.org/10.1038/377608a0 (1995).
35. Thiemann, C., Theis, F., Grady, D., Brune, R. & Brockmann, D. The structure of borders in a small world. *PLoS One* **5**, 1–7, https://doi.org/10.1371/journal.pone.0015422 (2010).
36. Roth, C., Kang, S. M., Batty, M. & Barthélemy, M. Structure of urban movements: Polycentric activity and entangled hierarchical flows. *PLoS One* **6**, 1–8, https://doi.org/10.1371/journal.pone.0015923 (2011).
37. Barthélemy, M. Spatial networks. *Physics Reports* **499**, 1–101, https://doi.org/10.1016/j.physrep.2010.11.002 (2011).
38. Guimerà, R., Mossa, S., Turtschi, A. & Amaral, L. A. N. The worldwide air transportation network: Anomalous centrality, community structure, and cities global roles. *Proceedings of the National Academy of Sciences* **102**, 7794–7799, https://doi.org/10.1073/pnas.0407994102 (2005).
39. Lee, S. H., Ffrancon, R., Abrams, D. M., Kim, B. J. & Porter, M. A. Matchmaker, matchmaker, make me a match: Migration of populations via marriages in the past. *Physical Review X* **4**, 041009, https://doi.org/10.1103/PhysRevX.4.041009 (2014).
40. Alves, L. G. A., Mendes, R. S., Lenzi, E. K. & Ribeiro, H. V. Scale-adjusted metrics for predicting the evolution of urban indicators and quantifying the performance of cities. *PLoS One* **10**, 1–17, https://doi.org/10.1371/journal.pone.0134862 (2015).
41. Domingues, G. S., Silva, F. N., Comin, C. H. & da F Costa, L. Topological characterization of world cities. *Journal of Statistical Mechanics: Theory and Experiment* **2018**, 083212, https://doi.org/10.1088/1742-5468/aad365 (2018).
42. Spadon, G., Gimenes, G. & Rodrigues, J. F. Topological street-network characterization through feature-vector and cluster analysis. In *International Conference on Computational Science*, 274–287, https://doi.org/10.1007/978-3-319-93698-7_21 (Springer, 2018).
43. Alves, L. G. A., Ribeiro, H. V., Lenzi, E. K. & Mendes, R. S. Distance to the scaling law: A useful approach for unveiling relationships between crime and urban metrics. *PLoS One* **8**, 1–8, https://doi.org/10.1371/journal.pone.0069580 (2013).
44. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 4768–4777 (Curran Associates Inc., USA, 2017).
45. Brazilian Institute of Geography and Statistics (IBGE). Accessed: 2017-09-01 (2017).
46. Moura, D. C. 3D Density Histograms for Criteria-driven Edge Bundling. *ArXiv:1504.0268* (2015).
47. Leibenstein, H. Shaping the world economy: Suggestions for an international economic policy. *The Economic Journal* **76**, 92–95, https://doi.org/10.2307/2229041 (1966).
48. Helpman, E., Melitz, M. & Rubinstein, Y. Estimating trade flows: Trading partners and trading volumes. *The Quarterly Journal of Economics* **123**, 441–487, https://doi.org/10.3386/w12927 (2008).
49. Kaluza, P., Kölzsch, A., Gastner, M. T. & Blasius, B. The complex network of global cargo ship movements. *Journal of The Royal Society Interface* **7**, 1093–1103, https://doi.org/10.1098/rsif.2009.0495 (2010).
50. Expert, P., Evans, T. S., Blondel, V. D. & Lambiotte, R. Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences* **108**, 7663–7668, https://doi.org/10.1073/pnas.1018962108 (2011).
51. Carpenter, R. Principles and procedures of statistics, with special reference to the biological sciences. *The Eugenics Review* **52**, 172, https://doi.org/10.1002/bimj.19620040313 (1960).
52. Chiang, C. *Statistical Methods of Analysis*. Statistical Methods of Analysis (World Scientific, 2003).
53. Pedregosa, F. *et al.* Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).
54. Chen, T. & Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 785–794, https://doi.org/10.1145/2939672.2939785 (ACM, 2016).
55. Lundberg, S. M. *et al.* Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering* **2**, 749, https://doi.org/10.1038/s41551-018-0304-0 (2018).
56. Lundberg, S. M., Erion, G. G. & Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888* (2018).
57. Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games* **2**, 307–317 (1953).
58. Keuschnigg, M., Mutgan, S. & Hedström, P. Urban scaling and the regional divide. *Science Advances* **5**, https://doi.org/10.1126/sciadv.aav0042 (2019).
59. Alves, L. G. A., Mangioni, G., Rodrigues, F., Panzarasa, P. & Moreno, Y. Unfolding the complexity of the global value chain: Strength and entropy in the single-layer, multiplex, and multi-layer international trade networks. *Entropy* **20**, 909, https://doi.org/10.3390/e20120909 (2018).

60. Alves, L. G. A. *et al.* The nested structural organization of the worldwide trade multi-layer network. *Scientific Reports* **9**, 2866, https://doi.org/10.1038/s41598-019-39340-w (2019).
61. Adamic, L. A. & Adar, E. Friends and neighbors on the web. *Social Networks* **25**, 211–230, https://doi.org/10.1016/S0378-8733(03)00009-1 (2003).
62. Guimera, R. & Amaral, L. A. N. Functional cartography of complex metabolic networks. *Nature* **433**, 895, https://doi.org/10.1038/nature03288 (2005).
63. Efron, B. Bootstrap methods: Another look at the jackknife. In Kotz, S. & Johnson, N. L. (eds) *Breakthroughs in Statistics: Methodology and Distribution*, 569–593 (Springer New York, New York, NY, 1992).
64. Friedman, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* **29**, 1189–1232 (2001).

Acknowledgments
The authors would like to thank Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001; Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), through grants 2013/07375-0, 2014/25337-0, 2016/16987-7, 2016/17078-0, 2016/18615-0, 2017/08376-0, and 2019/04461-9; Conselho Nacional de Desenvolvimento Cientifico e Tecnológico (CNPq), through grants 303694/2015-7, 404870/2016-3, 167967/2017-7, and 305580/2017-5; and Intel for their financial support. They also would like to thank Daniel M. Lima, Lucas C. Scabora, and Willian D. Oliveira for their help with the Edge Bundling algorithm.

Author Contributions
G.S. and L.G.A.A. conceived the experiments. G.S. conducted the experiments. G.S. and L.G.A.A. analyzed the results. G.S., A.C.C., J.F.R. and L.G.A.A. validated the results. G.S. and L.G.A.A. wrote the original draft. All authors reviewed and approved the manuscript.

Additional Information
Supplementary information accompanies this paper at https://doi.org/10.1038/s41598-019-48295-x.

Competing Interests: The authors declare no competing interests.

CHAPTER

4

DYNAMIC PROCESSES MODELING IN TIME

In this chapter, we reproduce the following article:

Spadon *et al.* 2021, *Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning.* Transactions on Pattern Analysis and Machine Intelligence. IEEE.

In the article, we introduce the Recurrent Graph Evolution Neural Network (ReGENN), which is a graph-inspired time-aware auto-encoder with linear and non-linear components that work together to provide future predictions based on observations from the past. The linear part is the autoregression, and the non-linear component is made of an auto-encoder powered by a pair of Graph Soft Evolution (GSE) layers, a further contribution of this study. The GSE holds for a graph-based learning-representation layer that improves the encoding and decoding processes by learning a shared graph over several time series and timestamps. We selected a range of time-series, machine learning, and deep learning algorithms for contrasting performance, from conventional to cutting-edge ones, totaling 49 algorithms. ReGENN surpassed all the baselines and remained effective after three rounds of 30 ablation tests through distinct hyperparameters. The experiments were carried out over the SARS-CoV-2, Brazilian Weather, and 2012 PhysioNet Computing in Cardiology datasets. In the task of epidemiology modeling on the SARS-CoV-2 dataset, we had improvements of at least 64.87%. We outperformed the task of climate forecasting on the Brazilian Weather dataset by at least 11.96%, and the task of patient monitoring on Intensive Care Units (ICUs) on the PhysioNet dataset by 7.33%.

We reproduce the article from Spadon *et al.* 2021 under the *Rights and Permissions* below on the following pages. To present the contribution related to this chapter: Section 1 presents the problem and related literature; Section 2 discusses the proposed network architecture; Section 3 displays the results in detail; Section 4 goes through the overall discussions; and, Section 5 presents the conclusions and final remarks. Additionally, in

Appendix B, we provide the supplementary material linked with the discussed article.

Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning

Gabriel Spadon, Shenda Hong, Bruno Brandoli,
Stan Matwin, Jose F. Rodrigues-Jr, and Jimeng Sun

Abstract—Time-series forecasting is one of the most active research topics in artificial intelligence. It has the power to bring light to problems in several areas of knowledge, such as epidemiological studies, healthcare inference, and climate change analysis. Applications in real-world time series should consider two factors for achieving reliable predictions: modeling dynamic dependencies among multiple variables and adjusting the model's intrinsic hyperparameters. An open gap in the literature is that statistical and ensemble learning approaches systematically present lower predictive performance than deep learning methods. The existing applications consistently disregard the data sequence aspect entangled with multivariate data represented in more than one time series. Conversely, this work presents a novel neural network architecture for time-series forecasting that combines the power of graph evolution with deep recurrent learning on distinct data distributions, named after Recurrent Graph Evolution Neural Network (ReGENN). The idea is to infer multiple multivariate relationships between co-occurring time-series by assuming that the temporal data depends not only on inner variables and intra-temporal relationships (*i.e.*, observations from itself) but also on outer variables and inter-temporal relationships (*i.e.*, observations from other-selves). An extensive set of experiments was conducted comparing ReGENN with tens of ensemble methods and classical statistical ones. The results outperformed both statistical and ensemble-learning approaches, showing an improvement of 64.87% over the competing algorithms on the SARS-CoV-2 dataset of the renowned John Hopkins University for 188 countries simultaneously. For further validation, we tested our architecture in two other public datasets of different domains, the PhysioNet Computing in Cardiology Challenge 2012 and Brazilian Weather datasets. We also analyzed the *Evolution Weights* arising from the hidden layers of ReGENN to describe how the variables of the dataset interact with each other; and, as a result of looking at inter and intra-temporal relationships simultaneously, we concluded that time-series forecasting is majorly improved if paying attention to how multiple multivariate data synchronously evolve.

Index Terms—Time Series, Graph Evolution, Representation Learning

1 Introduction

Time series refers to the persistent recording of a phenomenon along time, a continuous and intermittent unfolding of chronological events subdivided into past, present, and future. In the last decades, time series analysis has been vital to predict dynamic phenomena on a wide range of applications, such as climate change [1]–[4], financial market [5]–[7], land-use monitoring [8]–[10], anomaly detection [11]–[13], energy consumption, and price forecasting [14]–[16], apart from epidemiology and healthcare-related studies [17]–[22]. On such applications, an effective data-driven decision requires precise forecasting based on time series [23]. A prime example is the SARS-CoV-2, COVID-19, or Coronavirus Pandemic [24], which is known to be highly contagious and cause increased pressure on healthcare systems worldwide [25]. In this case, time-series analysis plays a vital role in planning a safe retake of fundamental activities by preventing economic systems' collapse.

Time series can be regarded as univariate or multivariate describing, respectively, single and multiple variables varying over time [26]. Recent techniques in time series have roots in the use of Artificial Neural Networks [27], which contain a non-linear functioning that enables it to outperform classical algorithms [28]. Such techniques evolved into deep learning models for time-series forecasting, such as Haoyi *et al.* [29] that used an informer component to enhance long-sequence time-series predictions but disregarded the inter-dependencies existing within different multivariate time-series, besides others from the spatiotemporal forecasting field. For example, Seo *et al.* [30] proposed the Graph Convolutional Recurrent Network (GCRN) from graph-structured and time-varying data by combining a Convolutional Neural Network (CNN) [31] that identifies spatial structures and a Recurrent Neural Network (RNN) [32] that learns dynamic patterns; Li *et al.* [33] introduced a spatiotemporal model for traffic forecasting with Diffusion over Convolutional Recurrent Neu-

- **G. Spadon** *and* **J. F. Rodrigues-Jr** *are with the University of Sao Paulo, Institute of Mathematics and Computer Sciences, Sao Carlos – SP, 13566-590, Brazil;* **G. Spadon** *and* **J. Sun** *are with the Georgia Institute of Technology, College of Computing, Atlanta – GA, 30332-0365, USA;* **B. Brandoli** *and* **S. Matwin** *are with the Dalhousie University, Institute for Big Data Analytics, Halifax – NS, B3H 1W5, Canada;* **S. Hong** *is with the National Institute of Health Data Science and the Institute of Medical Technology, both at the Peking University, Beijing, 100191, China;* **S. Matwin** *is with the Institute of Computer Science of the Polish Academy of Sciences, Warsaw, 525-000-94-01, Poland;* **J. F. Rodrigues-Jr** *is with Université Grenoble Alpes, Faculty of Science, Saint-Martin-d'Hères, 38400, France; and,* **J. Sun** *is with the University of Illinois Urbana-Champaign, Department of Computer Science, Champaign – IL, 61801-2302, USA.*

- **S. Hong** *and* **B. Brandoli** *contributed equally.*
- *Corresponding Author:* **J. Sun** *(jimeng@illinois.edu).*
- *Digital Object Identifier no. 10.1109/TPAMI.2021.3076155.*

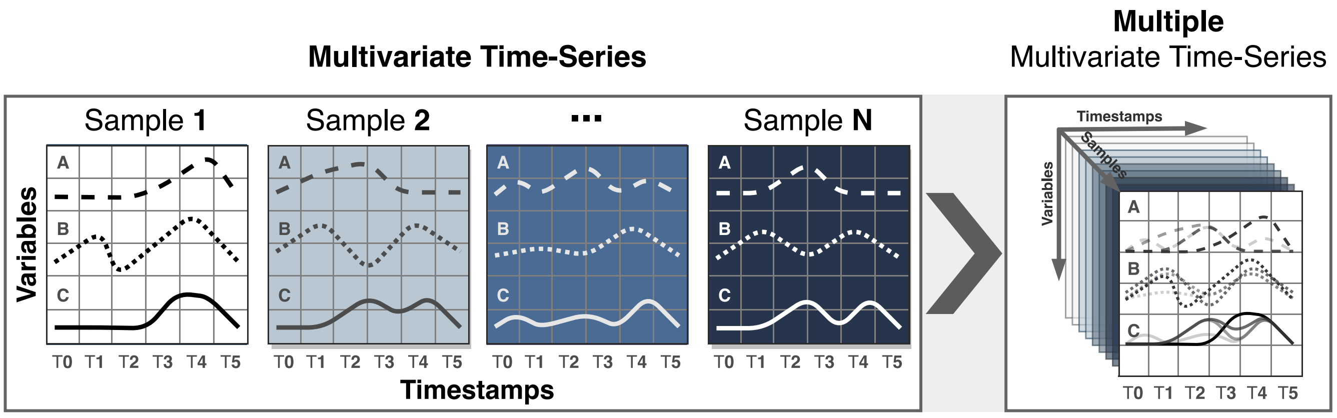


Fig. 1: A multiple multivariate time-series forecasting problem, where each multivariate time-series (*i.e.*, sample) shares the same domain, timestream, and variables. When stacking the time-series together, we assemble a tridimensional tensor with the axes describing samples, timestamps, and variables. The multiple samples have equal variables recorded during the same timestamps, meaning that samples are unique but all observed in the same way. By tackling the problem altogether, we leverage inner and outer variables besides intra- and inter-temporal relationships to improve forecasting.

ral Network (DCRNN); and, Zhang *et al.* [34] presented a Gated Attention Network (GaAN) for forecasting traffic speed using a Graph Gated Recurrent Unit (GGRU) with a CNN controlling the attention head's importance. Further contributions on the spatiotemporal field [35]–[37] employ traditional recurrent units, attention mechanisms, and even Graph Convolution Network (GCN) [38]. However, due to being designed to deal with spatial data, those models often fail to frame temporal dependencies as they aim to achieve state-of-the-art generalization across space and time at once.

Moreover, the LSTNet [39] encodes short-term data into low dimensional vectors by using a CNN for later decoding through an RNN; leverages from a recurrent-skip Gated Recurrent Unit for capturing long-term dependencies within the temporal data; and, incorporates an Autoregressive (AR) component in parallel to the non-linear neural network for preserving the scale of the output. Similarly, the DSANet [40] integrates an AR component with a dual self-attention network [41] with parallel convolutional components, a versatile idea for modeling global and local temporal patterns. More recently, the MLCNN [42] proposed the use of short and long-term prediction strategies for modeling temporal behavior through a multi-layer CNN together with Long Short-Term Memory (LSTM) [43] recurrent units. However, although LSTNet, DSANet, and MLCNN are cutting-edge multivariate time-series forecasting algorithms, they do not explicitly address per-variable and inter-time-series dependencies, which weakens their forecasting ability in the face of higher-dimensional data. Therefore, the state-of-the-art in time-series forecasting is bounded to a bidimensional space in which we understand the forecasting process by a non-linear function between time and variables.

Differently, we hypothesize that *time-series are dependent on their inner variables, which are observations from themselves, and from outer variables provided by different time series that share the same timestream*. For instance, the evolution of a biological species is not solely related to observations from itself but also from other species that share the same habitat, as they are all part of the same food chain. The time series gains an increased dimensionality by considering the variables and the dependency aspect during the analysis. Consequently, a previously considered bidimensional problem, in which a model's forecasting ability comes from observing relationships of variables over time, now becomes tridimensional, where forecasting means understanding the entanglement between variables of different time-series that co-occur in time. Accordingly, time-series define an event that is not a consequence of a single chain of observations but a set of synchronous observations of many time-series.

For example, during the Coronavirus Pandemic, it is paramount to understand the disease's time-aware behavior in every country. Despite progressing in different moments and locations, the pandemic's underlying mechanisms are supposed to follow similar (and probably interconnected) patterns. Along these lines, looking individually at the development of the pandemic in each country, one can describe the problem in terms of multiple variables, like the number of confirmed cases, recovered people, and deaths. However, when looking at all countries at once, the problem yields an additional data dimension, and each country becomes a multivariate sample of a broader problem, such as depicted in Fig. 1. In linguistic terms, we refer to such a problem as multiple multivariate time-series forecasting.

Along with these premises, in this study, we contribute with an unprecedented neural network that emerges from a graph-based time-aware auto-encoder with linear and non-linear components working in parallel to forecast multiple multivariate time-series simultaneously, named after Recurrent Graph Evolution Neural Network (REGENN). *We refer to evolution as the natural progression of a process where the neural network iteratively optimizes a graph representing observations from the past until it reaches an evolved version of itself that generalizes on future data still to be observed*. Accordingly, the underlying network structure of REGENN is powered by two Graph Soft Evolution (GSE) layers, a further contribution of this study. The GSE stands for a graph-based learning-representation layer that enhances the encoding and decoding processes by learning a shared graph across different time-series and timestamps.

The results we present are based on an extensive set of experiments, in which ReGENN surpassed a set of 49 competing algorithms from the fields of deep learning, machine learning, and time-series; among of which are single-target, multi-output, and multi-task regression algorithms in addition to univariate and multivariate time-series forecasting algorithms. Aside from surpassing the state-of-the-art, ReGENN remained effective after three rounds of 30 ablation tests through distinct hyperparameters. All experiments were carried out over the SARS-CoV-2, Brazilian Weather, and PhysioNet datasets. In the task of epidemiology modeling on the SARS-CoV-2 dataset, we had improvements of at least 64.87%. We outperformed the task of climate forecasting on the Brazilian Weather dataset by at least 11.96% and patient monitoring on intensive care units on the PhysioNet dataset by 7.33%. Furthermore, we analyzed the results using the *Evolution Weights* from the GSE layers, which are the intermediate hidden adjacency matrices that arise from the graph-evolution process after going through the cosine similarity activation, showing that graphs shed new light on the understanding of non-linear black-box models. Since multiple multivariate time-series is an ascending research topic, we understand ReGENN has implications in multiple domains, like economics, social sciences, and biology, in which different time-series share the same timestream and co-occur in time, mutually influencing one another.

In order to present our contributions, this paper is further divided into four sections. We begin by proposing a layer and neural network architecture, besides detailing the methods used along with this study. Subsequently, we display the experimental results compared to previous literature. Next, we provide an overall discussion on our proposal and the achieved results. Finally, we present the conclusions and final remarks. Alongside, the Supplementary Material presents extended methods and additional results.

2 Method

2.1 Preliminaries

Hereinafter, we use bold uppercase letters to denote multidimensional matrices (*e.g.*, $\mathbf{X}$), bold lowercase letters to vectors (*e.g.*, $\mathbf{x}$), and calligraphic letters to sets (*e.g.*, $\mathcal{X}$). Matrices, vectors, and sets can be used with subscripts. For example, the element in the i-th row and j-th columns of a matrix is $\mathbf{X}_{ij}$, the i-th element of a vector is $\mathbf{x}_i$, and the j-th element of a set is $\mathcal{X}_j$. The transposed matrix of $\mathbf{X} \in \mathbb{R}^{m\times n}$ is $\mathbf{X}^{\mathrm{T}} \in \mathbb{R}^{n\times m}$, and the transposed vector of $\mathbf{x} \in \mathbb{R}^{m\times 1}$ is $\mathbf{x}^{\mathrm{T}} \in \mathbb{R}^{1\times m}$, where m and n are arbitrary dimensions. We display in Tab. 1 a summary of all context-specific notations.

2.2 Graph Soft Evolution

Graph Soft Evolution (GSE) stands for a representation-learning layer that, given a training dataset, builds a graph in the form of an adjacency matrix, as in Fig. 2. The GSE layer receives no graph as input but a set of multiple multivariate time-series. The graph is built by tracking pairs of co-occurring variables, one sample at a time, and merging the results into a single co-occurrence graph shared among samples and timestamps. We define co-occurring variables as two variables, from a multivariate time-series, with a non-zero value in the same timestamp – in that case, we say one variable influences another and is influenced back. The co-occurrence graph is the projection of a tridimensional tensor, $\mathbf{T}$, $\mathbf{T} \in \mathbb{R}^{s\times w\times v}$, into a bidimensional one, $\mathbf{A}$, $\mathbf{A} \in \mathbb{R}^{v\times v}$, describing variables' pair-wise time-invariant relationships.

The co-occurrence graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ is symmetric and weighted. It is composed of a set $\mathcal{V}$ of $|\mathcal{V}|$ nodes equal to the

TABLE 1: Summary of context-specific notations.

Notation	Definition
$\omega \in \mathbb{N}^{+}$	Sliding window size
$w, z \in \mathbb{N}^{+}$	Number of training and testing (*i.e.*, stride) timestamps
$s, t, v \in \mathbb{N}^{+}$	Number of samples, timestamps, and variables
$\mathbf{T} \in \mathbb{R}^{s\times t\times v}$	Tensor of multiple multivariate time-series
$\mathbf{Y} \in \mathbb{R}^{s\times \omega\times v}$	Batched input of the first GSE and the Autoregression layers
$\mathbf{Y}_\alpha \in \mathbb{R}^{s\times \omega\times v}$	Output of the first GSE and input of the encoder layers
$\mathbf{Y}_\varepsilon \in \mathbb{R}^{s\times \omega\times v}$	Output of the encoder and input of the decoder layers
$\widetilde{\mathbf{Y}}_\varepsilon \in \mathbb{R}^{s\times z\times v}$	Output from the first recurrent unit and input to the second one
$\widetilde{\mathbf{Y}} \in \mathbb{R}^{s\times z\times v}$	Output of the second recurrent unit and input of the second GSE layer
$\mathbf{Y}_\psi \in \mathbb{R}^{s\times z\times v}$	Non-linear output yielded by the second GSE layer
$\mathbf{Y}_\lambda \in \mathbb{R}^{s\times z\times v}$	Linear output provided by the Autoregression layer
$\widehat{\mathbf{Y}} \in \mathbb{R}^{s\times z\times v}$	The final result from the merging of the linear and non-linear outputs
$\widehat{\mathbf{T}} \in \mathbb{R}^{s\times z\times v}$	The ground truth expected from merging the linear and non-linear outputs
$\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$	Graph in which $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ the set of edges
$\mathbf{A} \in \mathbb{R}^{v\times v}$	Adjacency matrix of co-occurring variables
$\mathbf{A}_\mu \in \mathbb{R}^{v\times v}$	Neighbor-smoothed per-variable embedding shared between GSE layers
$\mathbf{A}_\phi \in \mathbb{R}^{v\times v}$	Evolved adjacency matrix produced by the second GSE layer
$\mathbf{U} \circ \mathbf{V}$	Batch-wise Hadamard product between matrices $\mathbf{U}$ and $\mathbf{V}$
$\mathbf{U} \cdot \mathbf{V}$	Batch-wise scalar product between matrices $\mathbf{U}$ and $\mathbf{V}$
$\lVert \cdot \rVert_{\mathrm{F}}$	The Frobenius norm of a given vector or matrix
$\varphi(\cdot)$	Dropout regularization function
$\sigma_g(\cdot)$	Sigmoid activation function
$\sigma_h(\cdot)$	Hyperbolic tangent activation function
$\cos_\theta(\cdot)$	Cosine matrix-similarity activation function
$\mathrm{ReLU}(\cdot)$	Rectified exponential linear unit activation function
$\mathrm{SOFTMAX}(\cdot)$	Normalized exponential function activation function

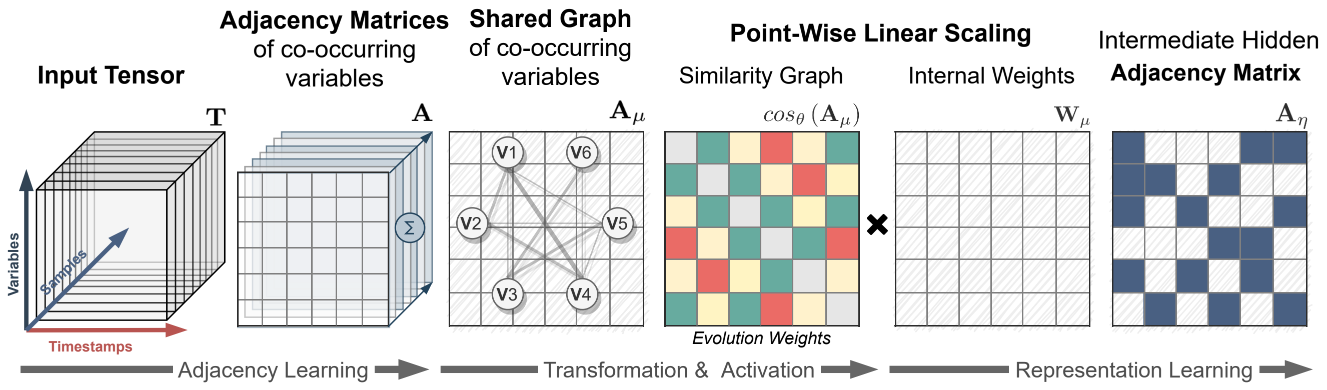

Input Tensor
Adjacency Matrices of co-occurring variables
Shared Graph of co-occurring variables
Point-Wise Linear Scaling
Similarity Graph
Internal Weights
Intermediate Hidden Adjacency Matrix
T
A
A_μ
cos_θ (A_μ)
W_μ
A_η
Variables
Samples
Timestamps
V1
V2
V3
V4
V5
V6
Evolution Weights
Adjacency Learning
Transformation & Activation
Representation Learning

Fig. 2: Graph Soft Evolution representation-learning, in which the set of multiple multivariate time-series is mapped into adjacency matrices of co-occurring variables. The matrices are element-wise summed to generate a shared graph among samples, which, after a linear transformation, goes through a similarity activation function and is scaled by an element-wise multiplication to produce an intermediate hidden adjacency-matrix with similarity properties inherent to the shared graph.

number of variables and another set $\mathcal{E}$ of $|\mathcal{E}|$ non-directed edges equal to the number of co-occurring variables. A node $v \in \mathcal{V}$ corresponds to a variable from the time-series multivariate domain, and an edge $e \in \mathcal{E}$ is an ordered pair $\langle u, v \rangle \equiv \langle v, u \rangle$ of co-occurring variables $u, v \in \mathcal{V}$. The edges' weight f corresponds to the summation of the values of the variables $u, v \in \mathcal{V}$ whenever they co-occur in time, such that $f(u, v) = \sum_{i=0}^{s-1} \sum_{j=0}^{w-1} \mathbf{T}_{i,j,u} + \mathbf{T}_{i,j,v}$. We use summation as the graph-merging operator when building $\mathbf{A}$ from $\mathcal{G}$ because, in the face of a zero-one input, a multiplicative operator would provide values close to zero, division would make those values overgrow toward infinity, subtraction would turn some of them into negative, while summation provides consistently positive values upper bounded to $2 \times (s \times w)$, helping to sustain the magnitude of variables' values. This way, the whole graph is bounded to w, which is the number of timestamps existing in the training portion of the input tensor, and if a pair of variables never co-occur in the training data, no edge will be assigned to the graph, meaning that $\langle u, v \rangle \notin \mathcal{E}$, and $f(u, v) = 0$.

The GSE layer for an arbitrary graph is formulated as:

$$\mathbf{A}_\mu = \mathbf{W}_\mu \cdot \mathbf{A} + \mathbf{b}_\mu \tag{1.1}$$

$$\mathbf{A}_\eta = \mathbf{W}_\eta \circ cos_\theta\left(\mathbf{A}_\mu\right) + \mathbf{b}_\eta \tag{1.2}$$

$$\mathbf{Y}_\alpha = \mathbf{W}_\alpha \cdot \varphi\left(\mathbf{Y} \cdot \mathbf{A}_\eta\right) + \mathbf{b}_\alpha \tag{1.3}$$

where $\mathbf{W}_\alpha$, $\mathbf{W}_\eta$, $\mathbf{W}_\mu \in \mathbb{R}^{v \times v}$ are symmetrically-unconstrained weights and $\mathbf{b}_\alpha$, $\mathbf{b}_\eta$, $\mathbf{b}_\mu \in \mathbb{R}^v$ the bias. In Eq. 1.1, the layer starts by employing a linear transformation to the shared adjacency matrix $\mathbf{A}$, providing a per-variable embedding $\mathbf{A}_\mu$ after smoothing across neighbors. Subsequently, in Eq. 1.2, it uses the cosine similarity (see the Supplementary Material) on the output of Eq. 1.1, *i.e.*, $cos_\theta(\mathbf{A}_\mu)$, which is an intermediate activation function that provides the *Evolution Weights* a symmetric per-variable similarity adjacency-matrix. Under the same equation, the *Evolution Weights* goes through a point-wise operator with a symmetrically-unconstrained weight matrix that enables the network to filter meaningful similarities from spurious

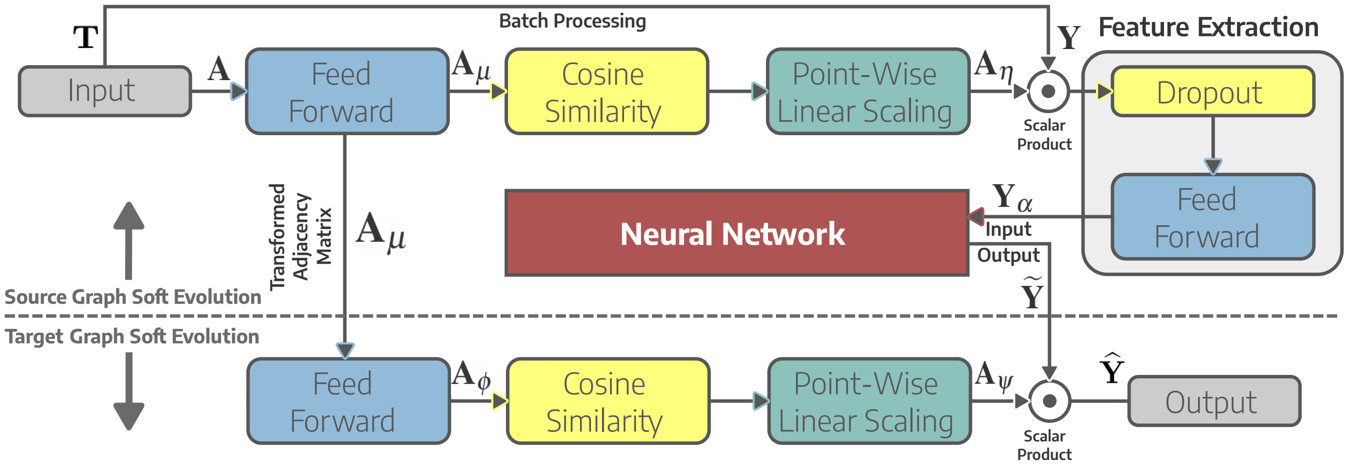


Fig. 3: Graph Soft Evolution layers assembled for evolution-based learning. In such a case, the first GSE layer's output (*i.e.*, source) will feed further layers of the neural network, whose result goes through the second GSE layer (*i.e.*, target). The GSE, as the last layer, does not use regularizers nor linear transformations before the output. Contrarily, it outputs the result from the scalar product between the learned representation and the data propagated throughout the network.

similarities by learning that two variables influence one another on different scales and resulting in $\mathbf{A}_\eta$. Next, in Eq. 1.3, it performs a batch-wise matrix-by-matrix multiplication between the adjacency matrix from Eq. 1.2 and the batched input tensor $\mathbf{Y}$ to combine the information from the graph, which generalizes samples and timestamps, with the time-series. The result will be followed by a dropout regularizer [44] and batch-wise matrix-by-matrix multiplication, where the final features from joining both tensors will be extracted and forwarded to the next layer of the network.

The evolution concept comes from the cooperation between two GSE layers, one at the beginning (*i.e.*, right after the input) and the other at the end (*i.e.*, right before the output) of a neural network, such as in the example shown in Fig. 3. As evolution arises from sharing hidden weights between a pair of non-sequential layers, we named this process after *Soft Evolution*. Accordingly, the first layer (*i.e.*, source) aims to learn the weights to scale the matrix and produce $\mathbf{A}_\mu$. Such a result is the input of the second GSE layer (*i.e.*, target), and it will be used for learning the evolved version of the adjacency matrix, referred to as $\mathbf{A}_\phi$ and produced as in Eq. 1.1. Notice that in Fig. 3, the source layer is different from the target one because we disregard the regularizer φ, trainable weights $\mathbf{W}_\alpha$, and bias $\mathbf{b}_\alpha$ from Eq. 1.3. They aim to enhance the feature-learning processes when multiple layers are stacked together and, as the last layer, GSE provides the output from already learned features through one last scalar product between the data propagated throughout the network, *i.e.*, $\widetilde{\mathbf{Y}}$, and the intermediate hidden adjacency-matrix, *i.e.*, $\mathbf{A}_\psi$.

One can see that the source GSE layer has two constant inputs: the graph and input tensor. The target GSE layer has two dynamic inputs, the shared graph from the source GSE layer and input propagated throughout the network. In this work scope, we use an auto-encoder between GSE layers to learn data codings from the source layer's output, which will be decoded into a representation closest to the expected output and later re-scaled by the target layer. In this sense, while the first layer learns a graph from the training data (*i.e.*, past data) working as a pre-encoding feature-extraction layer, the second one re-learns (*i.e.*, evolve) a graph at the end of the forecasting process based on future data, working as a post-decoding output-scaling layer. When joining the GSE layers with the auto-encoder, we assemble the Recurrent Graph Evolution Neural Network (REGENN).

2.3 Recurrent Graph Evolution Neural Network

REGENN is a graph-based time-aware auto-encoder with linear and non-linear components on parallel data-flows working together to provide future predictions based on past observations. The linear component is the autoregression implemented as a feed-forward layer, and the non-linear component is made of an encoder and a decoder module powered by a pair of GSE layers. Fig. 4 shows

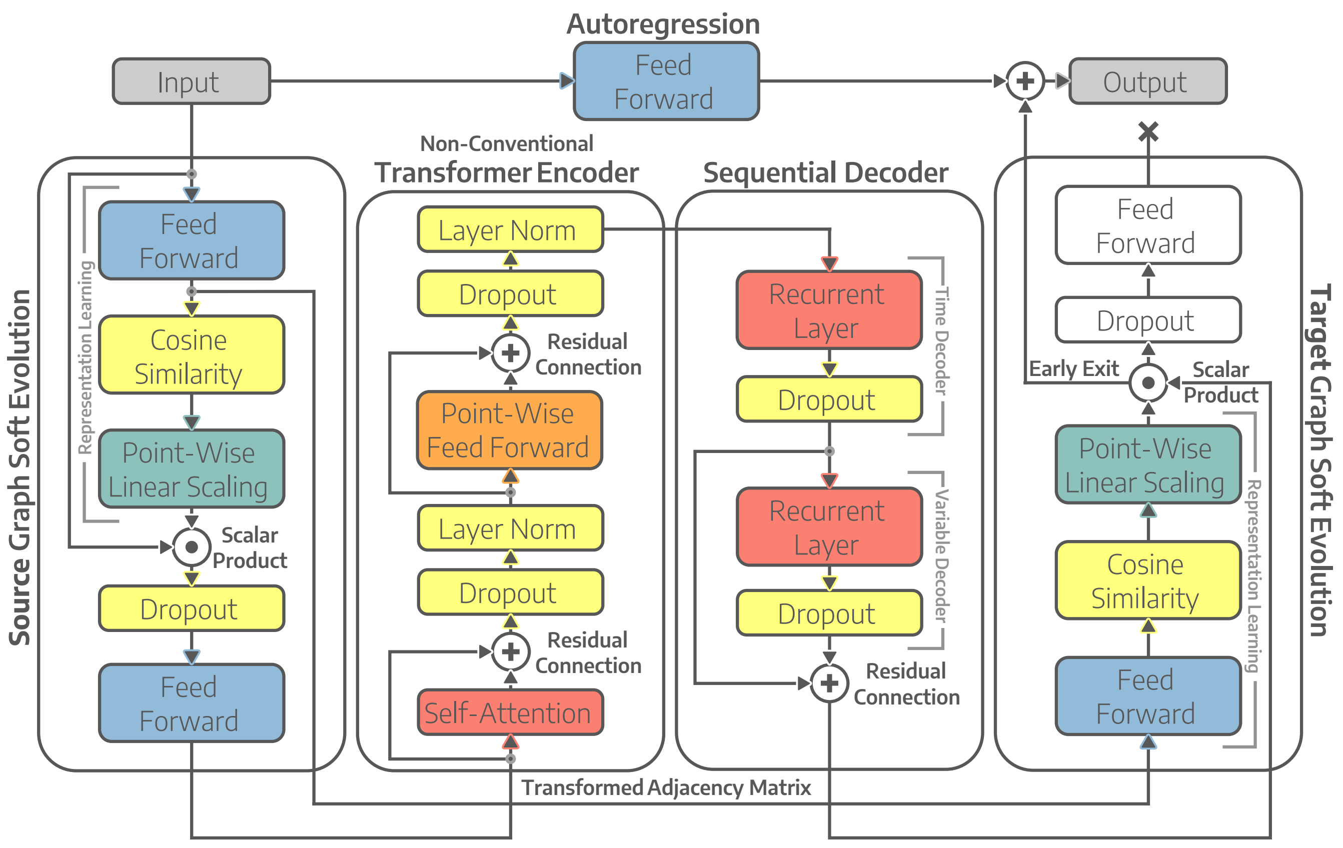


Fig. 4: Data diagram of the Recurrent Graph Evolution Neural Network (REGENN), which has a linear component parallel to a non-linear one. The linear component has a feed-forward layer, and the non-linear one has an auto-encoder and two GSE layers. Although equal to the first, the last GSE layer yields an early output as it is not stacked with another layer.

how these components communicate from the input to the output, and, in the following, we detail their operation.

Autoregressive Component

The non-periodical changes and constant progressions of the series across time usually decrease the performance of the network. That is because the output scale loses significance compared to the input, which comes from the complexity and non-linear nature of neural networks in time-series forecasting tasks [39]. Following a systematic strategy to deal with such a problem [45], [46], REGENN leverages from an Autoregressive (AR) layer working as a linear feed-forward shortcut between the input and output, which for a tridimensional input, is algebraically defined as:

$$\mathbf{Y}_\lambda = \mathbf{W} \cdot \mathbf{Y} + \mathbf{b} \equiv \left(\sum_{i=1}^{\omega} \mathbf{W}_{i,z} \times \mathbf{Y}_{s,i,v} \right) + \mathbf{b} \tag{2}$$

where $\mathbf{W} \in \mathbb{R}^{\omega \times z}$ are the weights and $\mathbf{b} \in \mathbb{R}^{z}$ the bias to be learned. The output of the linear component, *i.e.*, $\mathbf{Y}_\lambda \in \mathbb{R}^{s \times z \times v}$ as in Eq. 2, is element-wise added to the non-linear component's output, *i.e.*, $\mathbf{Y}_\psi \in \mathbb{R}^{s \times z \times v}$, so to produce the final predictions of the network $\widehat{\mathbf{Y}} \in \mathbb{R}^{s \times z \times v}$, formally given as $\widehat{\mathbf{Y}} = \mathbf{Y}_\lambda + \mathbf{Y}_\psi$. Subsequently, we describe the auto-encoder that produces the non-linear output of REGENN.

Encoder

We use a non-conventional Transformer Encoder [41] that employs self-attention to learn an encoding from the features forwarded by the GSE layer. It consists of multiple encoders joined through the scaled dot-product attention into a single set of encodings through multi-head attention. The number of expected features by the Transformer Encoder must be a multiple of the number of heads in the multi-head attention. Our encoder's non-conventionality comes from the fact that the first GSE layer's output goes through a single scaled dot-product attention on a single-head attention task. That is because the number of features produced by the encoder is equal to the length of the sliding window, and through single-head attention, the window can assume any length. The encoder module is defined as follows:

$$\mathbf{Y}_\varepsilon = \text{SELF-ATTENTION}\left(\mathrm{Q} : \mathbf{Y}_\alpha, \mathrm{K} : \mathbf{Y}_\alpha, \mathrm{V} : \mathbf{Y}_\alpha\right) \tag{3a}$$
$$\mathbf{Y}_\varepsilon = \text{LAYER-NORM}_{\gamma,\beta}\left(\mathbf{Y}_\varepsilon + \varphi\left(\mathbf{Y}_\varepsilon\right)\right) \tag{3b}$$
$$\mathbf{Y}_\varepsilon = \mathbf{W}_\varepsilon \cdot \varphi\left(\text{RELU}\left(\mathbf{W}_\iota \cdot \mathbf{Y}_\varepsilon + \mathbf{b}_\iota\right)\right) + \mathbf{b}_\varepsilon \tag{3c}$$
$$\mathbf{Y}_\varepsilon = \text{LAYER-NORM}_{\gamma,\beta}\left(\mathbf{Y}_\varepsilon + \varphi\left(\mathbf{Y}_\varepsilon\right)\right) \tag{3d}$$

where self-attention in Eq. 3a is a particular case of the multi-head attention, in which the input *query* Q, *key* K, and *value* V of the scaled dot-product attention, *i.e.*, $\text{SOFTMAX}\left(\mathrm{Q} \cdot \mathrm{K}^{\mathrm{T}} \div \sqrt{d_k}\right) \cdot \mathrm{V}$, are equal; and d_k is the dimension of the keys. The attention results are followed by a dropout regularization [44], a residual connection [47], and a layer normalization [48] as in Eq. 3b to ensure generalization. The first two layers work to avoid overfitting and gradient vanishing, while the last one normalizes the output such that the samples among the input have zero mean and unit variance $\gamma\left(\Delta\left(\mathbf{Y}_\varepsilon + \varphi\left(\mathbf{Y}_\varepsilon\right)\right)\right) + \beta$, where Δ is the normalization function, and γ and β are parameters to be learned. After, in Eq. 3c, the intermediate encoding goes through a double linear layer, a point-wise feed-forward layer, which, in this case, consists of two linear transformations in sequence with a ReLU activation in between, having the weights $\mathbf{W}_\varepsilon, \mathbf{W}_\iota$ and bias $\mathbf{b}_\varepsilon, \mathbf{b}_\iota$ as optimizable parameters. Finally, the transformed encoding goes through one last set of generalizing operations, as shown in Eq. 3d. The resulting encoding $\mathbf{Y}_\varepsilon \in \mathbb{R}^{s \times \omega \times v}$ is a tensor with the time-axis length matching the size of the sliding window ω.

Decoder

The previous encoding will be decoded by two sequence-to-sequence layers, which are Long Short Term Memory (LSTM) [43] units. The decoder operates in two of the tridimensional axes of the encoding, the time-axis and variable-axis, once at a time. Under the sample and time-axis, the time-axis decoder tracks temporal dependencies, looks for a set of weights that generalizes across variables, and translates the window-sized input into a stride-sized output:

$$\begin{aligned}
\mathbf{f}^{(t)}(v) &= \sigma_g\left(\mathbf{W}_{\mathbf{f}}^{(t)} \cdot \left[\mathbf{h}^{(t)}(v-1), \mathbf{Y}_\varepsilon^{(t)}(v)\right] + \mathbf{b}_{\mathbf{f}}^{(t)}\right) \\
\mathbf{i}^{(t)}(v) &= \sigma_g\left(\mathbf{W}_{\mathbf{i}}^{(t)} \cdot \left[\mathbf{h}^{(t)}(v-1), \mathbf{Y}_\varepsilon^{(t)}(v)\right] + \mathbf{b}_{\mathbf{i}}^{(t)}\right) \\
\mathbf{o}^{(t)}(v) &= \sigma_g\left(\mathbf{W}_{\mathbf{o}}^{(t)} \cdot \left[\mathbf{h}^{(t)}(v-1), \mathbf{Y}_\varepsilon^{(t)}(v)\right] + \mathbf{b}_{\mathbf{o}}^{(t)}\right) \\
\widetilde{\mathbf{C}}^{(t)}(v) &= \sigma_h\left(\mathbf{W}_{\mathbf{C}}^{(t)} \cdot \left[\mathbf{h}^{(t)}(v-1), \mathbf{Y}_\varepsilon^{(t)}(v)\right] + \mathbf{b}_{\mathbf{C}}^{(t)}\right) \\
\mathbf{C}^{(t)}(v) &= \mathbf{f}^{(t)}(v) \circ \mathbf{C}^{(t)}(v-1) + \mathbf{i}^{(t)}(v) \circ \widetilde{\mathbf{C}}^{(t)}(v-1) \\
\mathbf{h}^{(t)}(v) &= \varphi\left(\mathbf{o}^{(t)}(v) \circ \sigma_h\left(\mathbf{C}^{(t)}(v)\right)\right)
\end{aligned} \tag{4}$$

where v is the v-th variable of the t-th time-series group, and the weights $\mathbf{W}_{\mathbf{f}}^{(t)}, \mathbf{W}_{\mathbf{i}}^{(t)}, \mathbf{W}_{\mathbf{C}}^{(t)}, \mathbf{W}_{\mathbf{o}}^{(t)} \in \mathbb{R}^{\omega \times z}$ and bias $\mathbf{b}_{\mathbf{f}}^{(t)}, \mathbf{b}_{\mathbf{i}}^{(t)}, \mathbf{b}_{\mathbf{C}}^{(t)}, \mathbf{b}_{\mathbf{o}}^{(t)} \in \mathbb{R}^{z}$ are parameters to be learned. Along with Eq. 4, we refer to $\mathbf{f}$ as the forget gate's activation vector, $\mathbf{i}$ as the input and update gate's activation vector, $\mathbf{o}$ as the output gate's activation vector, $\widetilde{\mathbf{C}}$ as the cell input activation vector, $\mathbf{C}$ as the cell state vector, and $\mathbf{h}$ as the hidden state vector. The last hidden state vector goes through a dropout regularization φ before the next LSTM in the sequence.

Under the time and variable-axis, the next recurrent unit decodes the variable-axis from the partially-decoded encoding, searches for a set of weights that generalizes across time, and translates patterns arising from different variables on the same timestamp. The set of variables within the time-series does not necessarily imply a sequence, which does not interfere in the decoding process as long as the variables are always kept in the same order; a common approach used with boosting trees, such as the XGBoost [49] algorithm. The second LSTM, in which t is the t-th timestamp of the v-th variable group, is algebraically defined as follows:

$$\begin{aligned}
\mathbf{f}^{(v)}(t) &= \sigma_g\left(\mathbf{W}_{\mathbf{f}}^{(v)} \cdot \left[\mathbf{h}^{(v)}(t-1), \widetilde{\mathbf{Y}}_\varepsilon^{(v)}(t)\right] + \mathbf{b}_{\mathbf{f}}^{(v)}\right) \\
\mathbf{i}^{(v)}(t) &= \sigma_g\left(\mathbf{W}_{\mathbf{i}}^{(v)} \cdot \left[\mathbf{h}^{(v)}(t-1), \widetilde{\mathbf{Y}}_\varepsilon^{(v)}(t)\right] + \mathbf{b}_{\mathbf{i}}^{(v)}\right) \\
\mathbf{o}^{(v)}(t) &= \sigma_g\left(\mathbf{W}_{\mathbf{o}}^{(v)} \cdot \left[\mathbf{h}^{(v)}(t-1), \widetilde{\mathbf{Y}}_\varepsilon^{(v)}(t)\right] + \mathbf{b}_{\mathbf{o}}^{(v)}\right) \\
\widetilde{\mathbf{C}}^{(v)}(t) &= \sigma_h\left(\mathbf{W}_{\mathbf{C}}^{(v)} \cdot \left[\mathbf{h}^{(v)}(t-1), \widetilde{\mathbf{Y}}_\varepsilon^{(v)}(t)\right] + \mathbf{b}_{\mathbf{C}}^{(v)}\right) \\
\mathbf{C}^{(v)}(t) &= \mathbf{f}^{(v)}(t) \circ \mathbf{C}^{(v)}(t-1) + \mathbf{i}^{(v)}(t) \circ \widetilde{\mathbf{C}}^{(v)}(t-1) \\
\mathbf{h}^{(v)}(t) &= \widetilde{\mathbf{Y}}_\varepsilon + \varphi\left(\mathbf{o}^{(v)}(t) \circ \sigma_h\left(\mathbf{C}^{(v)}(t)\right)\right)
\end{aligned} \tag{5}$$

where $\widetilde{\mathbf{Y}}_\varepsilon$ is the partially-decoded encoding, and the weights $\mathbf{W}_{\mathbf{f}}^{(v)}$, $\mathbf{W}_{\mathbf{i}}^{(v)}$, $\mathbf{W}_{\mathbf{C}}^{(v)}$, $\mathbf{W}_{\mathbf{o}}^{(v)} \in \mathbb{R}^{z \times z}$ and bias $\mathbf{b}_{\mathbf{f}}^{(v)}$,

$\mathbf{b}_\mathbf{i}^{(v)}$, $\mathbf{b}_\mathbf{C}^{(v)}$, $\mathbf{b}_\mathbf{o}^{(v)} \in \mathbb{R}^z$ are internal parameters. The description of the notations within Eq. 4 holds for Eq. 5. Their difference is the residual connection with the partially-decoded encoding $\widetilde{\mathbf{Y}}_\varepsilon$ at the last hidden state vector after the dropout regularization φ. After the decoding is complete, the last recurrent unit output $\widetilde{\mathbf{Y}}$ goes through the second GSE layer, producing the non-linear output $\mathbf{Y}_\psi$ of REGENN.

2.4 Optimization Strategy

REGENN operates on a tridimensional space shared between samples, timestamps, and variables. In such a space, it carries out a time-based optimization strategy. The training process iterates over the time-axis of the dataset, showing to the network how the variables within a subset of time-series behave as time goes by and later repeating the process through subsets of different samples in distinct batches of preset window-size. However, the shared graph $\mathbf{A}$ is built during the first epoch of the training phase. Such a process happens in parallel to the optimization process, not impacting training stability nor convergence. After that point, $\mathbf{A}$ will remain as it is during the training and testing.

The network's weights are shared among the entire dataset and optimized towards best generalization simultaneously across samples, timestamps, and variables. We used Adam [50], a gradient descent-based algorithm, to optimize the model and, as the optimization criterion, the Mean Absolute Error (MAE), which is a generalization of the Support Vector Regression [51] with soft-margin criterion where Ω is the set of internal parameters of REGENN, $\widehat{\mathbf{Y}}$ is the network's output, and $\widehat{\mathbf{T}}$ the ground truth:

$$\underset{\Omega}{\textit{minimize}} \sum_{i=1}^{s} \sum_{j=1}^{w} \left| \widehat{\mathbf{Y}}_{i,j} - \widehat{\mathbf{T}}_{i,j} \right| \tag{6}$$

Due to the SARS-CoV-2 data behave as a streaming dataset, we adopted a transfer learning approach to train the network on that dataset. Transfer learning shares knowledge across different domains by using pre-trained weights of another neural network. The approach we adopted, although different, resembles Online Deep Learning [52]. The main idea is to train the network on incremental slices of the time-axis, such that the pre-trained weights of a previous slice are used to initialize the weights of the network in the next slice. This technique aims not only to achieve better forecasting performance but also to show that REGENN is superior to other algorithms throughout the pandemic.

3 EXPERIMENTS

3.1 Experimental Setup

Datasets

The results are based on three datasets, all of which are multi-sample, multivariate, and vary over time. The first dataset describes the Coronavirus Pandemic, referred to as SARS-CoV-2, made available by John Hopkins University [53]. It describes 3 variables through 120 days for 188 countries and varies from the first day of the pandemic to the day it completed four months of duration. The second one is the Brazilian Weather dataset collected from 253 sensors during 1,095 days regarding 4 variables. The third dataset is from the 2012 PhysioNet Computing in Cardiology Challenge [54], from which we are using 9 variables across 48 hours recorded from 11,988 ICU patients.

The variables within the datasets are:

- **SARS-CoV-2:** Number of Recovered, Number of Infected, and Number of Deaths;
- **Brazilian Weather:** Minimum Temperature, Maximum Temperature, Solar Radiation, and Rainfall; and,
- **PhysioNet:** Non-Invasive Diastolic Arterial Blood Pressure (mmHg), Non-Invasive Systolic Arterial Blood Pressure (mmHg), Invasive Diastolic Arterial Blood Pressure (mmHg), Non-Invasive Mean Arterial Blood Pressure (mmHg), Invasive Systolic Arterial Blood Pressure (mmHg), Invasive Mean Arterial Blood Pressure (mmHg), Urine Output (mL), Heart Rate (bpm), and Weight (Kg).

The number of datasets that can be used for multiple multivariate time-series forecasting is still limited. Although the ones we experimented with have a small number of variables, there is no theoretical upper bound for the number of samples, timestamps, and variables REGENN can handle. However, as REGENN deals with dense tridimensional tensors, increasing any of the axes $(samples \times time \times variables)$ in size will make the tensor grow at an almost-cubic rate, quickly overflowing the GPU Memory. Accordingly, REGENN ends up suffering from a trade-off between the dataset's size and the available hardware.

Data Pre-processing

The datasets were individually min-max normalized into a zero-one scale on the variable-axis to avoid gradient spikes and speed up training. We used the training data to define the min-max parameters required for normalization. Such parameters were applied for normalizing the test data as well. As a consequence of normalizing the entire dataset, the network's output will follow a similar scale. Therefore, the output was inversely transformed to what it was before the normalization for evaluating the results.

A simplistic yet effective approach to train time-series algorithms is through the Sliding Window technique [55], also referred to as Rolling Window. The window size is well known to be a highly sensitive hyperparameter [56], [57]. Consequently, we followed a non-tunable approach, in which we set the window size before the experiments, just taking into consideration the context and domain of the datasets. These values were used across all window-based experiments, including the baselines and ablation tests. It is noteworthy that most of the machine-learning algorithms are not meant to handle time-variant data, such that no sliding window was used in those cases. Conversely, we considered training timestamps as features and those reserved for testing as multi-task regression labels.

On the deep learning algorithms, we used a window size of 7 days for training and reserved 7 days for validation (between the training and test sets) to predict the last 14 days of the SARS-CoV-2 dataset. The 7-7-14 split idea comes from the disease incubation period, which is of 14 days. On the other hand, we used a window size of 84 days and reserved 28 days for validation to predict the last 56 days in the Brazilian Weather dataset. The 84-28-56 split is based on the seasonality of the weather data, such that we will look to the previous 3 months (a weather-season window) to predict the last 2 months of the upcoming season. Finally, we used a window size of 12 hours for training and 6 hours for validation to predict the last 6 hours of the PhysioNet dataset. The 12-6-6 split comes from the fact that patients in ICUs are in a critical state, such that predictions within 24 hours are more useful than long-term predictions.

Algorithms & Ensembles.

Many existing algorithms are limited because they neither support multi-task nor multi-output regression, making these algorithms even more limited to tasks when data is tridimensional. The most straightforward yet effective approach we followed to compare them to REGENN was to create a chain of ensembles[1]. In such a case, each estimator makes predictions on order specified by the chain using all of the available features provided to the model plus the predictions of earlier models in the chain. The number of estimators in each experiment varies according to the type of the ensemble and the type of the algorithms, and the final performance is the average of each estimator's performance. For simplicity sake, we grouped the algorithms as follows:

- ✹ Corresponds to tridimensional compliant algorithms of single estimators;
- ● Describes multivariate algorithms – s estimators, one estimator for each sample;
- ◎ Consists of multi-output and multi-task algorithms – v estimators, one estimator per variable;
- ⊙ Indicates single-target algorithms – $v \times z$ estimators, one estimator per variable and stride; and,
- ○ Represents univariate algorithms – $s \times v$ estimators, one estimator for each sample and variable.

Evaluation Metrics

As time-series forecasting works as a time-aware regression problem, our goal remains in predicting values that resemble the ground truth the most. As such, we used three evaluation metrics, the Mean Squared Logarithmic Error (MSLE), Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE), that, despite different, have a complementary meaning. The MSLE uses the logarithmic scale to assess how well the model can predict values close to zero. Contrarily, the MAE uses the absolute operator to explain how the model fared among the median values. In another perspective, the RMSE uses the square root to assess the model's ability to predict the larger values, which might be outliers. The metrics' formal definition is as follows:

$$MSLE = \frac{1}{n} \sum_{i=1}^{n} log \left(\frac{\widehat{\mathbf{Y}}_i + 1}{\widehat{\mathbf{T}}_i + 1} \right)^2 \tag{7}$$

$$MAE = \frac{1}{n} \sum_{i=1}^{n} \left| \widehat{\mathbf{Y}}_i - \widehat{\mathbf{T}}_i \right| \tag{8}$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\widehat{\mathbf{Y}}_i - \widehat{\mathbf{T}}_i \right)^2} \tag{9}$$

1. See *Regressor Chain* at https://bit.ly/3hBfxTA.

Hyperparameter Tuning

REGENN has two hyperparameters able to change the dimension of the weights' tensors, which are the window size (*i.e.*, input size) and the stride size (*i.e.*, output size). As already discussed, both were set before the experiments, and none of them were tuned towards any performance improvement. The trade-off of having fewer hyperparameters is to spend more energy on training the network towards a better performance. We are focusing on the network optimizer, gradient clipping, learning rate scheduler, and dropout regularization when we refer to tunable hyperparameters. Along these lines, we followed a controlled and limited exploratory approach similar to a random grid-search, starting with PYTORCH's defaults. The tuning process was on the validation set, intentionally reserved for measuring the network improvement. The tuning process follows by updating the hyperparameters whenever observing better results on the validation set, leading us to a set of optimized but no optimum hyperparameters. We used the set of optimized hyperparameters to evaluate REGENN on the test set and the default values for all the other algorithms [49], [58], [59] unless explicitly required for working with a particular dataset, as was the case of LSTNet [39], DSANet [40], and MLCNN [42]. The complete list of hyperparameters is in the Supplementary Material.

Computer Environment

The experiments related to machine-learning and time-series algorithms were carried out on a Linux-based system with 64 CPUs and 750 GB of RAM. The experiments related to deep-learning on the SARS-CoV-2 dataset were carried out on a Linux-based system with 56 CPUs, 256 GB of RAM, and 8 GPUs (Titan X – Pascal). The Brazilian Weather and PhysioNet datasets were tested on a different system with 32 CPUs, 512 GB of RAM, and 8 GPUs (Titan X – Maxwell). While CPU-based experiments are even across all CPU architectures, the same does not hold for GPUs, such that the GPU model and architecture must match to guarantee reproducibility. Aiming at complete reproducibility, we disclose not only the source code of REGENN on GitHub[2], but also the scripts, pre-processed data, and snapshots of all trained networks on a public folder at Google Drive[3].

3.2 Results

Subsequently, we go through the experimental results in each one of the benchmark datasets. Due to the fact that not all the algorithms perform evenly across them, we display the 34 most prominent ones out of the 49 tested algorithms; for the extended results, please refer to the Supplementary Material. We also discuss the ablation experiments, which were carried out with REGENN's hyperparameters; in the Supplementary Material, we provide two other rounds of this same experiment using as hyperparameters PYTORCH's defaults[4] and other settings recurrently employed in the literature. Additionally, we draw explanations about the *Evolution Weights*, *i.e.*, intermediate adjacency matrices from

2. Available at https://bit.ly/2YDBrOo.
3. Available at https://bit.ly/30csLiJ.
4. Available at https://bit.ly/2QuWRsD.

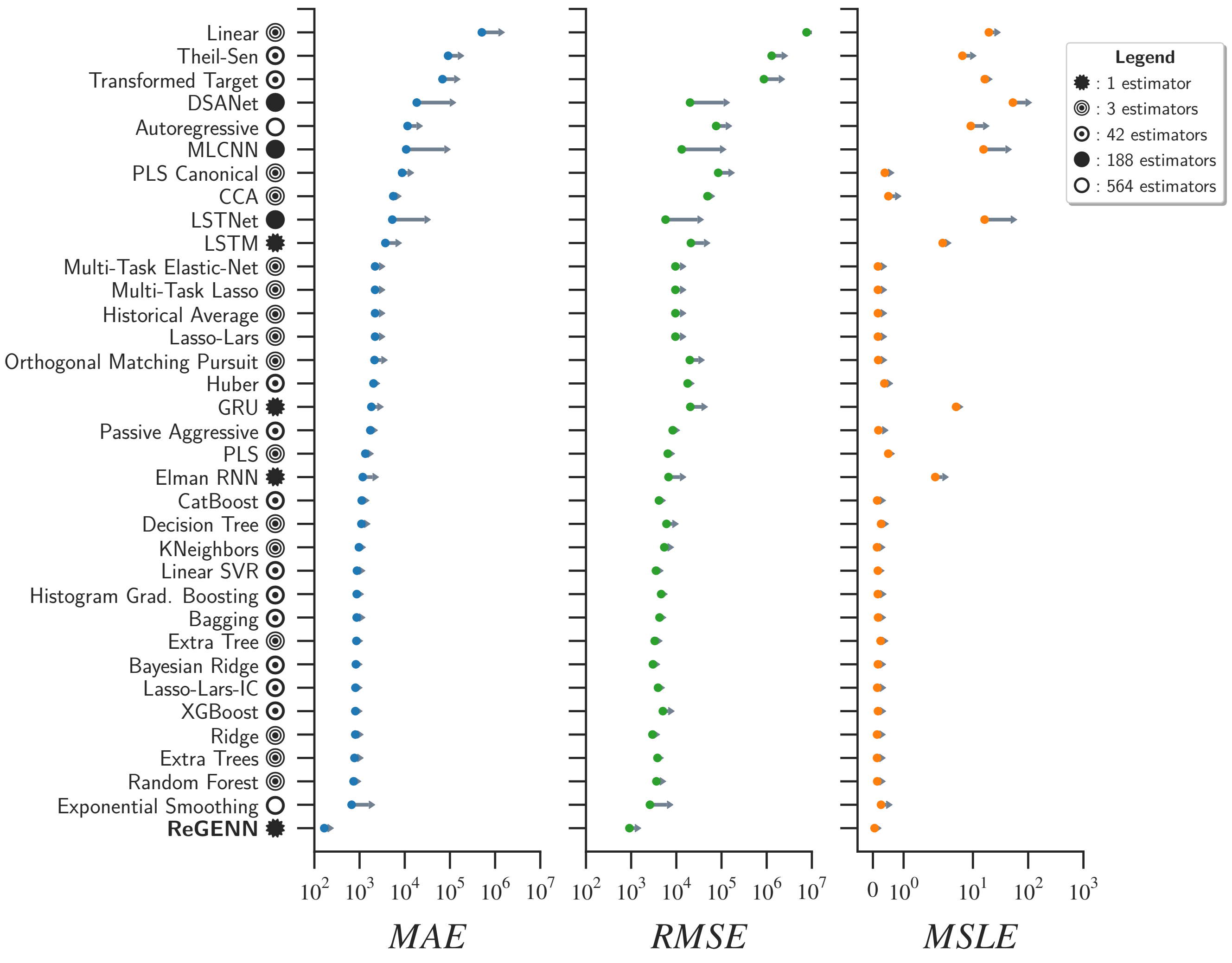


Fig. 5: Baseline results for the SARS-CoV-2 dataset over the Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Squared Logarithmic Error (MSLE). The results are presented in descending order of MAE (*i.e.*, worst performance at the top). The results confirmed ReGENN's superior performance as it is the algorithm with the lowest error and standard deviation – the improvement in the experiment was no lower than 64.87%. In the image, the algorithms are symbol-encoded based on their type and number of estimators; we use gray arrows to report the standard deviation of the results. The negative deviation, which is equal to the positive one, was suppressed for improved readability.

TABLE 2: Ablation results for the SARS-CoV-2 dataset using ReGENN's data-flow but no GSE layer. The experiment varies between Recurrent Unit (RU) and their directional flag, differing between Bidirectional (B) and Unidirectional (U). We use E to designate the non-conventional Transformer Encoder and AR the Autoregression component along with the table.

Recurrent Unit (RU)	**Elman RNN**			**GRU**			**LSTM**		
Network Architecture	MAE	RMSE	MSLE	MAE	RMSE	MSLE	MAE	RMSE	MSLE
${}_{\text{B}}\text{RU}$	424.32	2819.11	0.07	3130.47	28063.07	7.78	1800.23	20264.27	3.95
$\text{E} \rightarrow {}_{\text{B}}\text{RU}$	4161.35	28755.33	11.08	596.77	3446.67	0.17	704.80	4088.05	0.20
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	1624.44	17245.24	3.84	388.93	1888.70	0.09	1598.03	19056.28	3.89
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	**257.15**	**1438.72**	**0.09**	366.06	2158.32	0.10	4064.52	31196.10	11.30
$(\text{E} \rightarrow {}_{\text{B}}\text{RU}) + \text{AR}$	1471.49	17050.08	3.74	1502.20	16799.96	3.67	**211.93**	**1181.31**	**0.08**
$\text{E} \rightarrow {}_{\text{U}}\text{RU}$	1949.65	16925.69	3.84	968.83	5982.35	0.22	2778.55	12830.03	0.21
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	372.80	2155.36	0.11	**208.88**	**1141.56**	**0.08**	1380.99	16370.41	3.66
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	1472.05	16491.81	3.72	4052.71	31384.80	11.30	1713.72	18531.68	3.81
$(\text{E} \rightarrow {}_{\text{U}}\text{RU}) + \text{AR}$	1350.35	16673.11	3.69	2852.28	25869.18	7.60	260.66	1513.92	0.10
${}_{\text{U}}\text{RU}$	1175.99	6716.42	2.08	1825.66	20400.04	4.95	3723.88	21022.60	2.83
Recurrent Graph Evolution Neural Network (ReGENN)							**165.41**	**915.92**	**0.05**
ReGENN without Variable Decoder							197.51	1141.04	0.15
ReGENN without Autoregression							8208.53	50089.84	19.19
Autoregression Only							11529.79	76118.62	9.15
Overall Improvement							20.81%	19.77%	35.72%

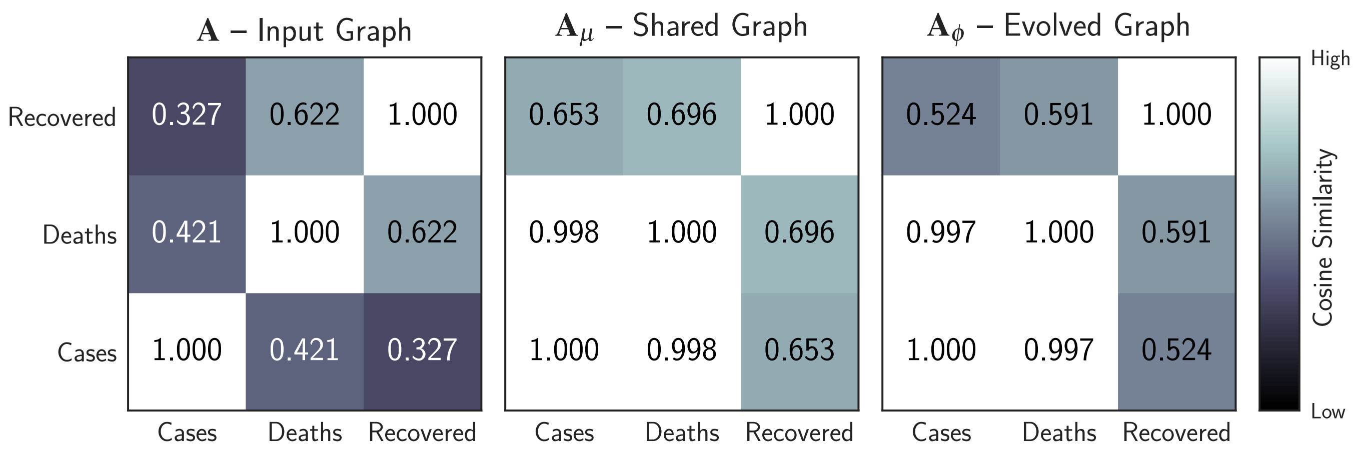


Fig. 6: Set of *Evolution Weights*, *i.e.*, cosine-similarity activated hidden weights extracted from REGENN at the end of the network training. The images compare the cosine similarity between the set of variables within the SARS-CoV-2 dataset.

the GSE layers, by using the cosine similarity on the adjacency matrices of co-occurring variables (see Section 2.2).

Experiment on SARS-CoV-2 dataset

The SARS-CoV-2 has being updated daily since the beginning of the Coronavirus Pandemic. We used a self-to-self transfer-learning approach to train the network in slices of time due to such a dataset's streaming nature. In short, the network was re-trained every 15 days with new incoming data, using as starting weights, the pre-trained weights from the network trained in the past 15 days so to evaluate REGENN's performance over the pandemic's progression.

In such a case, when the pandemic completed 45 days, the first time-slice in which REGENN was trained, it outperformed the second-placed algorithm, the Orthogonal Matching Pursuit, by 27.27% on the Mean Absolute Error (MAE), 16.50% on the Root Mean Square Error (RMSE), and 38.87% on the Mean Squared Logarithmic Error (MSLE). For all subsequent time-slices (*i.e.*, 60, 75, 90, 105, and 120 days), the second-placed algorithm was the Exponential Smoothing, which was outperformed with improvement no lower than 47.40% on the MAE, 17.19% on the RMSE, and 37.39% on the MSLE. We further detail the results on the time-slices mentioned above in the Supplementary Material. However, in Fig. 5, we detail the results on the complete dataset of 120 days, in which REGENN surpassed the Exponential Smoothing, the second-best algorithm, by 75.21% on the MAE, 64.87% on the RMSE, and 79.61% on the MSLE.

As a result of the analysis of the dataset in time-slices, we noticed that, as time goes by and more information is available on the SARS-CoV-2 dataset, the problem becomes more challenging to solve by looking individually at each country and more natural when looking at all of them together. Although countries have their particularities, which make the disease spread in different ways, the main goal is to decrease the spreading, such that similarities between the historical data of different countries provide for finer predictions. Furthermore, we observed that not all the estimators within an ensemble perform in the same way in the face of different countries. Due to the REGENN capability of observing inter-and intra-relationships between time-series, it performs better on highly uncertain cases like this one.

Subsequently, we present the ablation results, in which we utilized the same data-flow as REGENN but no GSE layer while systematically changing the decoder architecture. We provide results using different Recurrent Units (RU), including the Elman RNN [32], LSTM [43], and GRU [60]. We also varied the recurrent unit's directional flag between Unidirectional (U) and Bidirectional (B). That because a unidirectional recurrent unit tracks only forward dependencies while a bidirectional one tracks both forward and backward dependencies. A summarized tag describes each test's network architecture; for example, $(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$ means the model has a Transformer Encoder (E) as the encoder, a Unidirectional Recurrent Unit as the time-axis decoder, and a Bidirectional Recurrent Unit as the variable-axis decoder. Besides that, the output of the decoder is element-wise added to the Autoregression (AR) output. The table shows results with and without the encoder and AR component, besides using a single recurrent unit only for time-axis decoding.

According to the ablation results in Tab. 2, the improvement of REGENN is slightly smaller than previously reported. That is because its performance not only comes from the GSE layer but also from how the network handles the multiple multivariate time-series data. Consequently, the ablation experiments reveal that some models without GSE layers are enough to surpass all the competing algorithms. However, when using REGENN, we can improve them further and achieve 20.81% of additional reduction on the MAE, 19.77% on the RMSE, and 35.72% on the MSLE.

Fig. 6 shows the *Evolution Weights* originated from applying the cosine similarity on the hidden adjacency matrices of REGENN. When comparing the input and evolved graphs, the number of cases and deaths have a mild similarity. That might come from the fact that diagnosing infected people was already a broad concern at the beginning of the pandemic. The problem did not go away, but more infected people were discovered as more tests were made, and also because the disease spread worldwide. A similar scenario can be drawn from the number of recovered people and the number of cases, as infected people with mild or no symptoms were unaware of being infected. Contrarily, we can see that the similarity between recovered and deaths

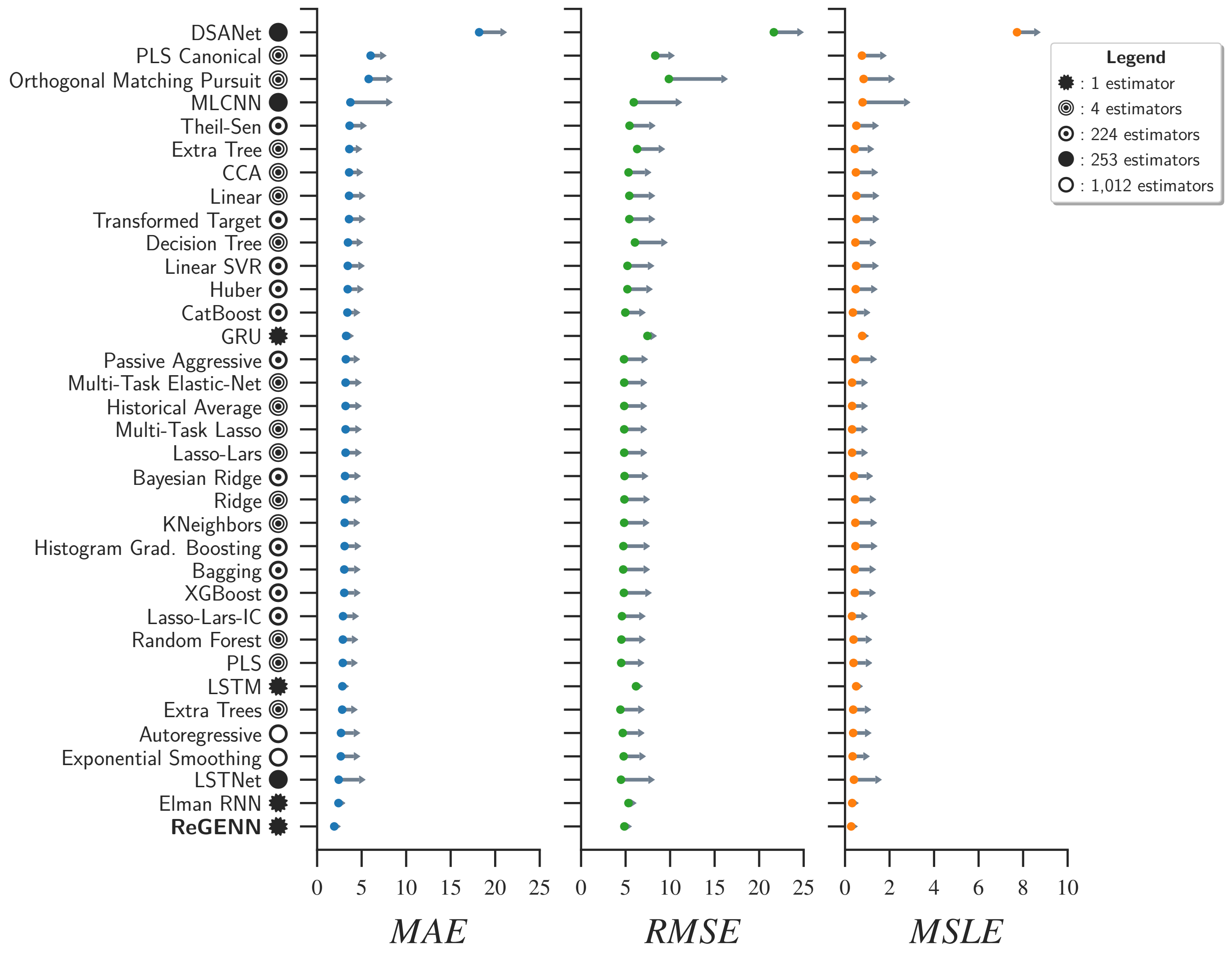


Fig. 7: Baseline results for the Brazilian Weather dataset presented in descending order of MAE. In this experiment, ReGENN once more outperformed all the competing algorithms, demonstrating versatility by performing well even on a highly-seasonal dataset with improvement no lower than 11.95%. In the face of seasonality, the Elman RNN surpassed the Exponential Smoothing, the previously second-best algorithm. In the image, the algorithms are symbol-encoded based on their type and number of estimators; we use gray arrows to report the results' standard deviation. The negative deviation, which is equal to the positive one, was suppressed for improved readability.

TABLE 3: Ablation results from experimenting over the Brazilian Weather dataset that was conducted by varying the Recurrent Unit (RU) and its directional flag between Bidirectional (B) and Unidirectional (U). We use E to designate the non-conventional Transformer Encoder and AR the feed-forward Autoregressive linear component.

Recurrent Unit (RU)	Elman RNN			GRU			LSTM		
Network Architecture	MAE	RMSE	MSLE	MAE	RMSE	MSLE	MAE	RMSE	MSLE
$_{\mathrm{B}}$RU	2.56	5.90	0.45	2.19	5.00	0.28	**2.20**	**5.04**	**0.28**
E $\rightarrow$ $_{\mathrm{B}}$RU	2.70	5.66	0.37	2.33	5.19	0.28	2.57	5.42	0.29
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{B}}$RU) + AR	**2.18**	**5.52**	**0.37**	2.38	5.72	0.41	2.44	5.75	0.41
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{U}}$RU) + AR	2.84	6.47	0.55	2.24	5.05	0.28	3.17	7.42	0.80
(E $\rightarrow$ $_{\mathrm{B}}$RU) + AR	3.58	8.13	0.98	3.31	7.57	0.82	3.61	8.19	0.96
E $\rightarrow$ $_{\mathrm{U}}$RU	3.12	5.86	0.32	2.81	5.69	0.34	2.96	5.54	0.30
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{B}}$RU) + AR	2.36	5.69	0.41	2.23	5.11	0.28	2.47	5.77	0.41
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{U}}$RU) + AR	2.52	5.83	0.42	**2.10**	**4.97**	**0.28**	4.88	10.24	1.61
(E $\rightarrow$ $_{\mathrm{U}}$RU) + AR	4.30	9.38	1.34	3.25	7.55	0.81	5.25	10.69	1.75
$_{\mathrm{U}}$RU	2.40	5.32	0.32	3.26	7.45	0.77	2.83	6.17	0.50
Recurrent Graph Evolution Neural Network (ReGENN)							**1.92**	**4.86**	**0.28**
ReGENN without Variable Decoder							1.95	4.85	0.27
ReGENN without Autoregression							2.00	4.95	0.27
Autoregression Only							2.69	4.68	0.37
Overall Improvement							8.42%	2.16%	0.00%

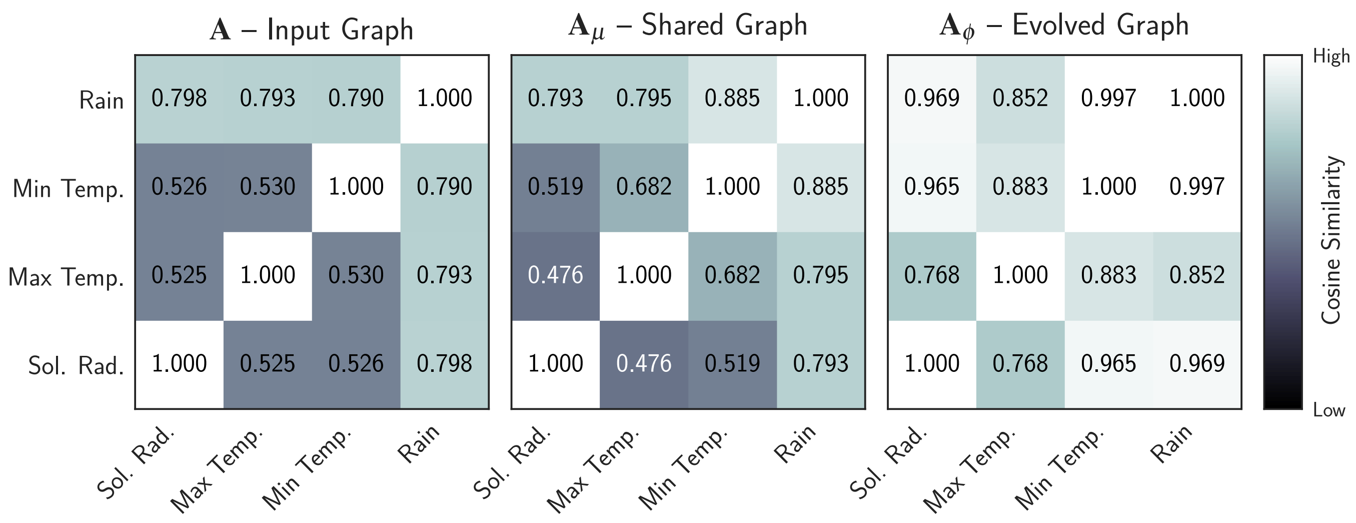


Fig. 8: The *Evolution Weights* from REGENN for the Brazilian Weather dataset, in which we use the cosine similarity activation function on the network's hidden weights to compare the relationship between pairs of variables. The image uses "Sol. Rad." as a shortening for Solar Radiation, "Temp" for Temperature, "Max" for Maximum, and "Min" for Minimum.

decreases over time, which comes from the fact that, as more tests are made, the mortality rate drops to a stable threshold due to the increased number of recovered people.

Experiment on Brazilian Weather dataset

The Brazilian Weather dataset is a highly seasonal dataset with a higher number of samples, variables, and timestamps than the previous one. For simplicity's sake, in this experiment, REGENN was trained on the whole training set at once. The results are in Fig. 7, in which REGENN was the first-placed algorithm, followed by the Elman RNN in second. REGENN overcame the Elman RNN by 11.95% on the MAE, 11.96% on the RMSE, and 25.84% on the MSLE.

We noticed that all the algorithms perform quite similarly for this dataset. The major downside for most algorithms comes from predicting small values close to zero, as noted by the MSLE results. In such a case, the ensembles showed a high variance when compared to REGENN. We believe this is why the Elman RNN shows performance closer to REGENN rather than to Exponential Smoothing, the third-placed algorithm, as REGENN has a single estimator, while the Exponential Smoothing is an ensemble of estimators. Another understanding of why some algorithms underperform on the MSLE is related to their difficulty to track temporal dependencies, such as weather seasonality.

The ablation results are in Tab. 3, in which we observed again that the network without the GSE layers already surpasses the baselines. When decommissioning the GSE layers of REGENN and using GRU instead of LSTM on the decoder, we observed a 3.86% improvement on the MAE, 10.02% on the RMSE, and 25.34% on the MSLE when compared to the Elman RNN results. Using REGENN instead, we achieve a further performance gain of 8.42% on the MAE and 2.16% on the RMSE over the ablation experiment.

Fig. 8 depicts the *Evolution Weights* for the current dataset, in which we can observe a consistent similarity between pairs of variables in the input graph, which does not repeat in the evolved graph, implying different relationships. We observe that the similarity between all pairs of variables increased on the evolved graph. The pairs *Solar Radiation* and *Rain*, *Maximum Temperature* and *Rain*, and *Solar Radiation* and *Minimum Temperature* stood out. Those pairs are mutually related, which comes from solar radiation interfering in maximum and minimum temperature and in the precipitation factors, where the opposite relation holds. What can be extracted from the *Evolution Weights*, in this case, is the notion of importance between pairs of variables so that the pairs that stood out are more relevant and provide better information during the forecasting process.

Experiment on PhysioNet dataset

The PhysioNet dataset presents a large number of samples and an increased number of variables, but little information on the time-axis, a setting in which ensembles still struggle to perform accurate predictions, as depicted in Fig. 9. Once again, REGENN keeps steady as the first-placed algorithm in performance, showing solid improvement over the Linear SVR, the second-placed algorithm. The improvement was 7.33% on the MAE and 35.13% on the MSLE, while the RMSE achieved by REGENN laid within the standard deviation of the Linear SVR, pointing out an equivalent performance between them. The Linear SVR is an ensemble of multiple estimators, while REGENN uses only one, making it better accurate for dealing with the current dataset.

As in Tab. 4, the ablation results reveal that a neural network architecture without the GSE layers can achieve a better performance than the baseline algorithms. In this specific case, we see that by using a bidirectional LSTM instead of unidirectional on the decoder module of the neural network, we can achieve a performance almost as good as REGENN, but not enough to surpass it, as REGENN still shows an improvement of 1.05% on the MAE and 0.98% on the RMSE over the experiment with bidirectional LSTM.

In this specific case, REGENN learns by observing multiple ICU patients. However, one cannot say that an ICU patient's state is somehow connected to another patient's state. Contrarily, the idea holds as in the first experiment,

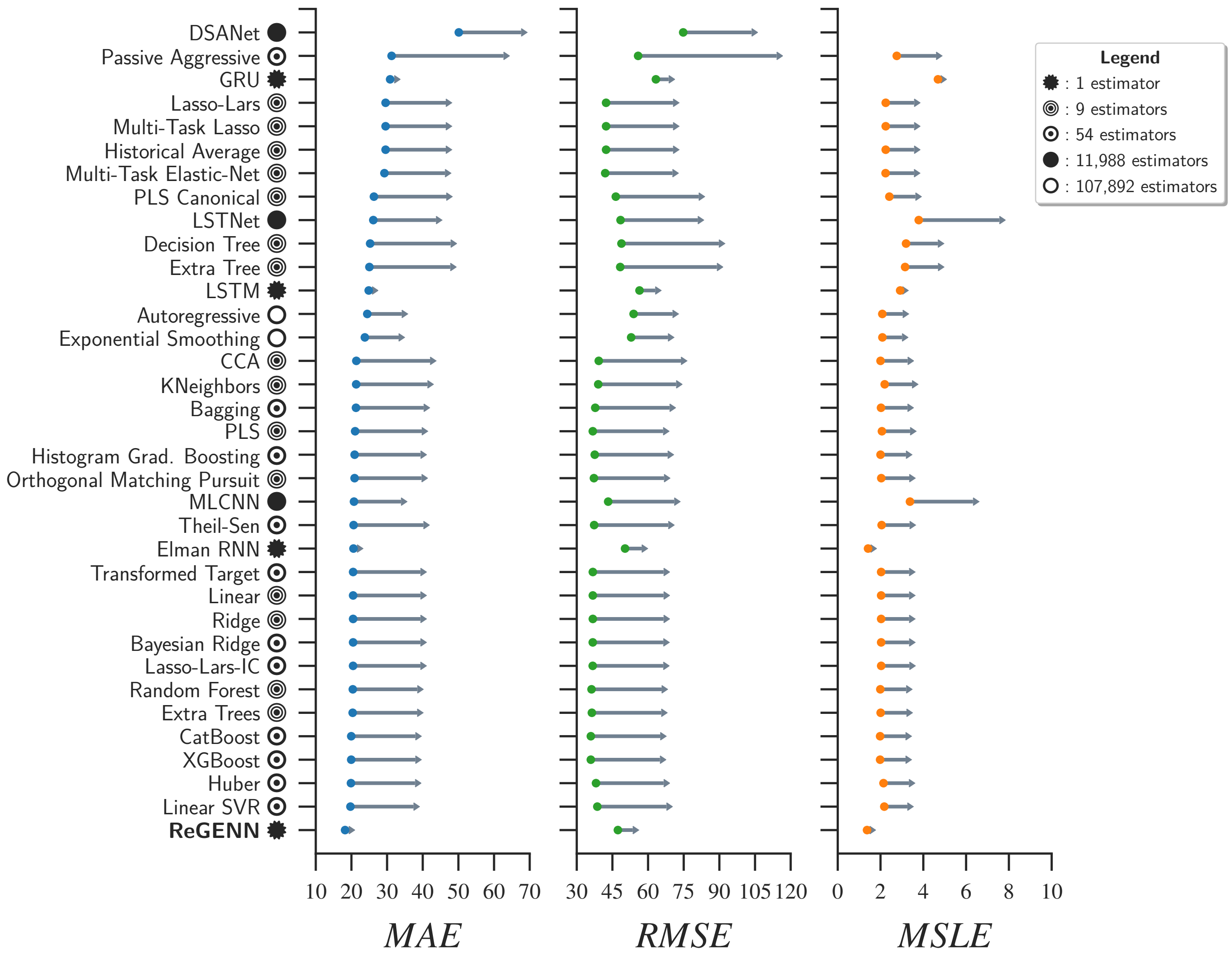


Fig. 9: Baseline results for the PhysioNet Computing in Cardiology Challenge 2012 dataset presented in descending order of MAE, in which ReGENN was the algorithm with the best performance followed by the Linear SVR. The comparative improvement was no lower than 7.33%, but, in this case, ReGENN yielded an RMSE compatible with the Linear SVR. In the image, the algorithms are symbol-encoded based on their type and number of estimators; we use gray arrows to report standard deviation. The negative deviation, which is equal to the positive one, was suppressed for improved readability.

TABLE 4: Ablation results over the PhysioNet Computing in Cardiology Challenge 2012 dataset, in which the Recurrent Unit (RU) is changed together with its directional flag, varying between Bidirectional (B) and Unidirectional (U). We use E as a shortening for the Transformer Encoder and AR for the Autoregressive linear component.

Recurrent Unit (RU)	**Elman RNN**			**GRU**			**LSTM**		
Network Architecture	MAE	RMSE	MSLE	MAE	RMSE	MSLE	MAE	RMSE	MSLE
$_{\text{B}}$RU	19.95	50.47	1.38	24.80	56.42	3.01	19.49	50.46	1.40
E $\rightarrow$ $_{\text{B}}$RU	19.11	49.16	1.41	24.17	54.81	3.00	18.88	48.55	1.40
(E $\rightarrow$ $_{\text{B}}$RU + $_{\text{B}}$RU) + AR	**18.72**	**48.60**	**1.37**	18.55	47.86	1.39	**18.42**	**47.78**	**1.37**
(E $\rightarrow$ $_{\text{B}}$RU + $_{\text{U}}$RU) + AR	18.97	49.59	1.38	**18.54**	**48.90**	**1.37**	18.46	48.33	1.36
(E $\rightarrow$ $_{\text{B}}$RU) + AR	18.81	50.20	1.38	18.67	48.59	1.39	18.54	47.35	1.38
E $\rightarrow$ $_{\text{U}}$RU	19.46	50.24	1.44	19.42	51.03	1.44	19.78	51.29	1.44
(E $\rightarrow$ $_{\text{U}}$RU + $_{\text{B}}$RU) + AR	18.76	48.81	1.39	18.65	48.64	1.38	18.55	48.28	1.38
(E $\rightarrow$ $_{\text{U}}$RU + $_{\text{U}}$RU) + AR	19.02	49.75	1.43	18.64	48.81	1.38	18.63	48.93	1.37
(E $\rightarrow$ $_{\text{U}}$RU) + AR	18.98	48.67	1.45	18.82	49.98	1.41	18.88	50.31	1.39
$_{\text{U}}$RU	20.58	50.32	1.42	30.85	63.30	4.68	24.85	56.41	2.92
Recurrent Graph Evolution Neural Network (ReGENN)							**18.22**	**47.31**	**1.37**
ReGENN without Variable Decoder							18.23	47.45	1.37
ReGENN without Autoregression							18.48	49.72	1.37
Autoregression Only							24.44	53.92	2.08
Overall Improvement							1.05%	0.98%	0.0%

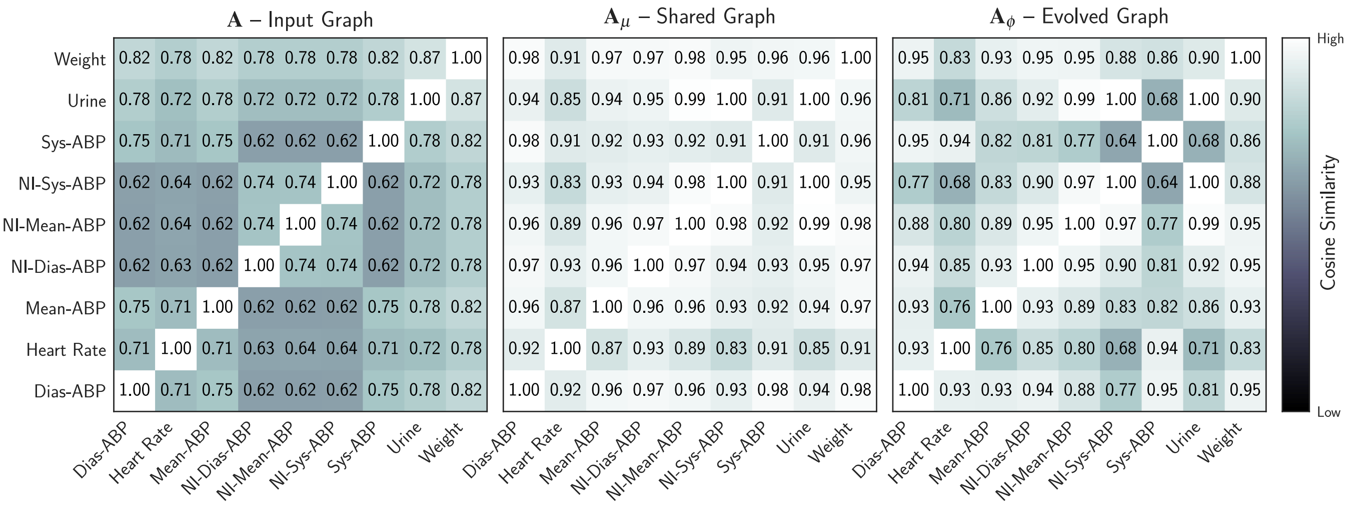


Fig. 10: *Evolution Weights* extracted from ReGENN after training on the PhysioNet Computing in Cardiology Challenge 2012 dataset, in which we use the cosine similarity to compare the relationship between pairs of variables. We use "ABP" as a shortening for *Arterial Blood Pressure*, "NI" as *Non-Invasive*, "Dias" as *Diastolic*, and "Sys" as *Systolic*.

where although the samples are different, they have the same domain, variables, and timestream, such that the information from one sample might help enhance future forecasting for another one. That means ReGENN learns both from the past of the patient and from the past of other patients. Nevertheless, we must be careful about drawing any understanding about these results, as the reason each patient is in the ICU is different, and while some explanations might be suited for a small set of patients, it tends not to generalize to a significant number of patients. When analyzing the *Evolution Weights* in Fig. 10 aided by a physician, we can say that there is a relationship between the amount of urine excreted by a patient and the arterial blood pressure, and also that there is a relation between the systolic and diastolic blood pressure. However, even aided by the *Evolution Weights*, we cannot further describe these relations once there are variables of the biological domain that are not being taken into consideration.

4 Overall Discussions

We refer to the *Evolution Weights* as the intermediate weights of the representation-learning process optimized throughout the network's training. Such weights are time-invariant and are a requirement for the feature-learning behind the GSE layer. Although time does not flow through the adjacency matrix, the network is optimized as a whole, such that every operation influences the gradients of the backward propagation process. That means the optimizer, influenced by the gradients of both time-variant and invariant data, will optimize the weights towards a better forecasting ability. Such a process depends not only on the network architecture but also on the optimization process's reliability.

That increases uncertainty, which is the downside of ReGENN, demanding more time to train the neural network and causing the improvement not to be strictly uprising. Consequently, training might take long sessions, even with consistently reduced learning rates on plateaus or simulated annealing techniques; this is influenced by the fact that the second GSE layer has two dynamic inputs, which arise from the graph-evolution process. However, we observed that throughout the epochs, the *Evolution Weights* reach a stable point with no major updates. As a result, the network demonstrates a remarkable improvement in its final iterations when the remaining weights intensely converge to a near-optimal configuration.

Even though ReGENN has a particular drawback, it demonstrates excellent versatility, which comes from its superior performance in the task of epidemiology modeling on the SARS-CoV-2 dataset, climate forecasting on the Brazilian Weather, and patient monitoring on intensive care units on the PhysioNet dataset. Consequently, we see ReGENN as a tool to be used in data-driven decision-making tasks, helping prevent, for instance, natural disasters or preparing for an upcoming pandemic. As a precursor in multiple multivariate time-series forecasting, there is still much to be improved. For example, reducing the uncertainty that harms ReGENN without decreasing its performance should be the first step, followed by extending the proposal to handle problems in the spatiotemporal field of great interest to traffic forecasting and environmental monitoring. Another possibility would be to remove the decoder's recurrent layers while tracking the temporal dependencies through multiple graphs, a new temporal-modeling perspective in which one could leverage from Graph Convolution Networks for extracting inter-variable relationships and using those as hidden-features during the time-series forecasting process, to which further hypothesis constraints may apply.

Notwithstanding, in some cases, where extensive generalization is not required, the analysis of singular multivariate time-series may be preferred to multiple multivariate time-series. That because, when focusing on a single series at a time, some but not all samples might yield a lower forecasting error, as the model will be driven to a single multivariate sample. However, both approaches for tackling time-series forecasting can coexist in the state-of-the-art, and, as a consequence, the decision to work on a higher or lower dimensionality must relate to which problem is being solved and how much data is available to solve it.

5 Conclusion

This paper tackles multiple multivariate time-series forecasting tasks by proposing the Recurrent Graph Evolution Neural Network (ReGENN), a graph-based time-aware auto-encoder powered by a pair of Graph Soft Evolution (GSE) layers, a further contribution of this study that stands for a graph-based learning-representation layer.

The literature handles multivariate time-series forecasting with outstanding performance, but up to this point, we lacked a technique with increased generalization over multiple multivariate time-series with sound performance. Previous research might have avoided tackling such a problem as a neural network, to that matter, is challenging to train and usually yields poor results. That because one aims to achieve good generalization on future observations for multivariate time-series that do not necessarily hold the same data distribution.

Because of that, ReGENN is a precursor in multiple multivariate time-series forecasting and, even though this is a challenging problem, ReGENN surpassed all the baselines and remained effective after three rounds of 30 ablation tests through distinct hyperparameters. The experiments were carried out over the SARS-CoV-2, Brazilian Weather, and PhysioNet datasets with improvements, respectively, of at least 64.87%, 11.96%, and 7.33%. As a consequence of the results, ReGENN shows a new range of possibilities in time-series forecasting, starting by demonstrating that ensembles poorly perform if compared to a single model able to learn the entanglement between different variables by looking at how they interact as time goes by and how multiple multivariate time-series synchronously evolve.

Authors contributions statement

GS conceived the experiments, GS prepared the data for experimentation, and GS conducted them. SH and BB validated GS's source-code implementation. SH and BB analyzed the preliminary results. SM, JR, and JS validated the final results. GS wrote the original draft. SH, BB, SM, JR, and JS iteratively polished the main ideas and the paper. All authors reviewed and approved the results and manuscript.

Acknowledgments

This work was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brazil (CAPES) – Finance Code 001; Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), through grants 2014/25337-0, 2016/17078-0, 2017/08376-0, 2018/17620-5, 2019/04461-9, and 2020/07200-9; Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) through grants 167967/2017-7, 305580/2017-5, and 406550/2018-2; National Science Foundation awards IIS-2014438, PPoSS-2028839, and IIS-1838042; and, the National Institute of Health awards NIH R01 1R01NS107291-01, and R56HL138415. We thank Jeffrey Valdez for his aid with Sunlab's computer infrastructure, Lucas Scabora, for his careful review of the paper, and Gustavo Merchan, M.D., for his input on the *Evolution Weights* of the PhysioNet dataset.

References

[1] S. L. Lavender, K. J. E. Walsh, L.-P. Caron, M. King, S. Monkiewicz, M. Guishard, Q. Zhang, and B. Hunt, "Estimation of the maximum annual number of North Atlantic tropical cyclones using climate models," *Science Advances*, vol. 4, no. 8, aug 2018.

[2] Y.-C. Kao, M. W. Rogers, D. B. Bunnell, I. G. Cowx, S. S. Qian, O. Anneville, T. D. Beard, A. Brinker, J. R. Britton, R. Chura-Cruz, N. J. Gownaris, J. R. Jackson, K. Kangur, J. Kolding, A. A. Lukin, A. J. Lynch, N. Mercado-Silva, R. Moncayo-Estrada, F. J. Njaya, I. Ostrovsky, L. G. Rudstam, A. L. E. Sandström, Y. Sato, H. Siguayro-Mamani, A. Thorpe, P. A. M. van Zwieten, P. Volta, Y. Wang, A. Weiperth, O. L. F. Weyl, and J. D. Young, "Effects of climate and land-use changes on fish catches across lakes at a global scale," *Nature Communications*, vol. 11, no. 1, pp. 25–26, dec 2020.

[3] S. C. Pryor, R. J. Barthelmie, M. S. Bukovsky, L. R. Leung, and K. Sakaguchi, "Climate change impacts on wind power generation," *Nature Reviews Earth & Environment*, vol. 1, no. 12, pp. 627–643, dec 2020.

[4] L. Laurent, J.-F. Buoncristiani, B. Pohl, H. Zekollari, D. Farinotti, M. Huss, J.-L. Mugnier, and J. Pergaud, "The impact of climate change and glacier mass loss on the hydrology in the Mont-Blanc massif," *Scientific Reports*, vol. 10, no. 1, dec 2020.

[5] A. Kelotra and P. Pandey, "Stock Market Prediction Using Optimized Deep-ConvLSTM Model," *Big Data*, vol. 8, no. 1, pp. 5–24, feb 2020.

[6] M. Ananthi and K. Vijayakumar, "Stock market analysis using candlestick regression and market trend prediction (CKRM)," *Journal of Ambient Intelligence and Humanized Computing*, apr 2020.

[7] S. Jiao, T. Shen, Z. Yu, and H. Ombao, "Change-point detection using spectral PCA for multivariate time series," jan 2021.

[8] D. Roy and L. Yan, "Robust Landsat-based crop time series modelling," *Remote Sensing of Environment*, vol. 238, mar 2020.

[9] Z. Zhu, J. Zhang, Z. Yang, A. H. Aljaddani, W. B. Cohen, S. Qiu, and C. Zhou, "Continuous monitoring of land disturbance based on Landsat time series," *Remote Sensing of Environment*, vol. 238, pp. 111–116, mar 2020.

[10] L. Yan and D. P. Roy, "Spatially and temporally complete Landsat reflectance time series modelling: The fill-and-fit approach," *Remote Sensing of Environment*, vol. 241, may 2020.

[11] N. Laptev, S. Amizadeh, and I. Flint, "Generic and Scalable Framework for Automated Time-series Anomaly Detection," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: ACM, aug 2015, pp. 1939–1947.

[12] J. Li, W. Pedrycz, and I. Jamal, "Multivariate time series anomaly detection: A framework of Hidden Markov Models," *Applied Soft Computing*, vol. 60, pp. 229–240, nov 2017.

[13] J. Hancock and T. M. Khoshgoftaar, "Performance of CatBoost and XGBoost in Medicare Fraud Detection," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, dec 2020, pp. 572–579.

[14] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 74, pp. 902–924, jul 2017.

[15] A. Orlov, J. Sillmann, and I. Vigo, "Better seasonal forecasts for the renewable energy industry," *Nature Energy*, vol. 5, no. 2, pp. 108–110, feb 2020.

[16] S. G. Baratsas, A. M. Niziolek, O. Onel, L. R. Matthews, C. A. Floudas, D. R. Hallermann, S. M. Sorescu, and E. N. Pistikopoulos, "A framework to predict the price of energy for the end-users with applications to monetary and energy policies," *Nature Communications*, vol. 12, no. 1, dec 2021.

[17] I. Perros, E. E. Papalexakis, R. Vuduc, E. Searles, and J. Sun, "Temporal phenotyping of medically complex children via PARAFAC2 tensor factorization," *Journal of Biomedical Informatics*, vol. 93, no. September 2018, pp. 103–125, may 2019.

[18] J. F. Rodrigues-Jr, G. Spadon, B. Brandoli, and S. Amer-Yahia, "Patient trajectory prediction in the Mimic-III dataset, challenges and pitfalls," *arXiv*, sep 2019.

[19] J. F. Rodrigues, G. Spadon, B. Brandoli, and S. Amer-Yahia, "Lig-doctor: Real-world clinical prognosis using a bi-directional neural network," in *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*, 2020, pp. 569–572.

[20] A. Afshar, I. Perros, H. Park, C. DeFilippi, X. Yan, W. Stewart, J. Ho, and J. Sun, "TASTE," in *Proceedings of the ACM Conference on Health, Inference, and Learning*, ser. CHIL '20. New York, NY, USA: ACM, apr 2020, pp. 193–203.

[21] L. Yan, H.-T. Zhang, J. Goncalves, Y. Xiao, M. Wang, Y. Guo, C. Sun, X. Tang, L. Jing, M. Zhang, X. Huang, Y. Xiao, H. Cao, Y. Chen, T. Ren, F. Wang, Y. Xiao, S. Huang, X. Tan, N. Huang, B. Jiao, C. Cheng, Y. Zhang, A. Luo, L. Mombaerts, J. Jin, Z. Cao, S. Li, H. Xu, and Y. Yuan, "An interpretable mortality prediction model for COVID-19 patients," *Nature Machine Intelligence*, vol. 2, no. 5, pp. 283–288, may 2020.

[22] J. F. Rodrigues-Jr, M. A. Gutierrez, G. Spadon, B. Brandoli, and S. Amer-Yahia, "LIG-Doctor: Efficient patient trajectory prediction using bidirectional minimal gated-recurrent networks," *Information Sciences*, vol. 545, pp. 813–827, feb 2021.

[23] M. Ienca and E. Vayena, "On the responsible use of digital data to tackle the COVID-19 pandemic," *Nature Medicine*, vol. 26, no. 4, pp. 463–464, apr 2020.

[24] T. P. Velavan and C. G. Meyer, "The COVID-19 epidemic," *Tropical Medicine & International Health*, vol. 25, no. 3, pp. 278–280, mar 2020.

[25] S. B. Omer, P. Malani, and C. del Rio, "The COVID-19 Pandemic in the US," *JAMA*, vol. 323, no. 18, pp. 1767–1768, apr 2020.

[26] A. Silvestrini and D. Veredas, "Temporal aggregation of univariate and multivariate time series models: A survey," *Journal of Economic Surveys*, vol. 22, no. 3, pp. 458–497, jul 2008.

[27] A. Jain, Jianchang Mao, and K. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, mar 1996.

[28] G. Zhang, B. Patuwo, and M. Y. Hu, "A simulation study of artificial neural networks for nonlinear time-series forecasting," *Computers & Operations Research*, vol. 28, no. 4, pp. 381–396, apr 2001.

[29] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," *CoRR*, vol. abs/2012.0, dec 2020.

[30] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds., vol. 11301 LNCS. Springer International Publishing, 2018, pp. 362–373.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, jun 1990.

[33] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, jul 2017.

[34] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, "GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs," in *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, mar 2018, pp. 339–349.

[35] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. California: International Joint Conferences on Artificial Intelligence Organization, jul 2018, pp. 3428–3434.

[36] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, sep 2017, pp. 3634–3640.

[37] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, sep 2020.

[38] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[39] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. New York, NY, USA: ACM, jun 2018, pp. 95–104.

[40] S. Huang, X. Wu, D. Wang, and A. Tang, "DSANet: Dual self-attention network for multivariate time series forecasting," in *International Conference on Information and Knowledge Management, Proceedings*, ser. CIKM'19. New York, NY, USA: ACM, nov 2019, pp. 2129–2132.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 2017-December, pp. 5999–6009, jun 2017.

[42] J. Cheng, K. Huang, and Z. Zheng, "Towards better forecasting by fusing near and distant future visions," *arXiv*, 2019.

[43] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[45] J. G. Zilly, R. K. Srivastava, J. Koutnik, and J. Schmidhuber, "Recurrent highway networks," *34th International Conference on Machine Learning, ICML 2017*, vol. 8, pp. 6346–6357, jul 2017.

[46] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway Networks," *arXiv*, 2015.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016, pp. 770–778.

[48] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv*, vol. abs/1607.0, jul 2016.

[49] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-August-2016, ACM. New York, NY, USA: ACM, aug 2016, pp. 785–794.

[50] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[51] V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 281–287.

[52] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, "Online Deep Learning: Learning Deep Neural Networks on the Fly," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. California: International Joint Conferences on Artificial Intelligence Organization, jul 2018, pp. 2660–2666.

[53] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track COVID-19 in real time," *The Lancet Infectious Diseases*, vol. 20, no. 5, pp. 533–534, may 2020.

[54] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, "Predicting in-hospital mortality of ICU patients: The PhysioNet/Computing in cardiology challenge 2012," in *Computing in Cardiology*, vol. 39, 2012, pp. 245–248.

[55] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: a Survey and Novel Approach," in *Data Mining in Time Series Databases*, ser. Series in Machine Perception and Artificial Intelligence. World Scientific, jun 2004, vol. Volume 57, pp. 1–21.

[56] R. Frank, N. Davey, and S. Hunt, "Input window size and neural network predictors," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 2. IEEE, 2000, pp. 237–242.

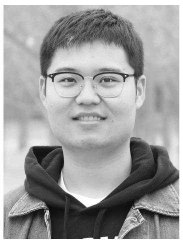
[57] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 31, no. 1-3, pp. 91–103, 2001.

[58] P. Fabian, Michel Vincent, G. Olivier, B. Mathieu, P. Peter, W. Ron, Vanderplas Jake, and D. Cournapeau, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[59] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., oct 2018, pp. 6638–6648.

[60] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv*, dec 2014.

Gabriel Spadon is currently working toward the PhD degree in Computer Science at the University of Sao Paulo, Brazil, part of which was carried out with the Georgia Institute of Technology, USA. He has worked intensively on network science and artificial intelligence during the last years. He has authored or co-authored several research articles on knowledge discovery through complex networks and data mining. His current research interests include neural-inspired models, graph-based learning, and complex networks.


Shenda Hong received the PhD degree from the School of Electronics Engineering and Computer Sciences, Peking University, in 2019. He is currently a (Boya) postdoctoral fellow at the National Institute of Health Data Science, Peking University. From 2019 to 2020, he was a postdoctoral fellow with the College of Computing, Georgia Institute of Technology, and a visiting researcher with Harvard Medical School. His research interests include machine learning methods for temporal sequences and time series, especially deep-learning methods for electronic health records and biomedical signals, including ECG and EEG.

Bruno Brandoli received the PhD degree (with honors) in computer science from the University of Sao Paulo, Brazil, in 2016. He is currently a faculty at the Computer Science Department and a researcher with the Institute for Big Data Analytics, Dalhousie University, Canada. Over the past years, he has authored or coauthored AI-related articles in relevant journals and conferences. His main research interests include deep-learning-based models, computer vision, and data-driven learning. He is currently leading projects focused on the state-of-the-art AI models dedicated to marine biology.

Stan Matwin is currently the director of the Institute for Big Data Analytics, Dalhousie University, Halifax, Nova Scotia, where he is a professor and Canada Research Chair (Tier 1) in Interpretability for Machine Learning. He is also a distinguished professor (Emeritus) at the University of Ottawa and a full professor with the Institute of Computer Science, Polish Academy of Sciences. His main research interests include big data, text mining, machine learning, and data privacy. He is a member of the Editorial Boards of *IEEE Transactions on Knowledge and Data Engineering* and *Journal of Intelligent Information Systems*. He was the recipient of the Lifetime Achievement Award of the Canadian AI Association (CAIAC).

Jose F. Rodrigues-Jr received the PhD from the University of Sao Paulo, Brazil, part of which was carried out from Carnegie Mellon University, USA, in 2007. He is currently an associate professor at the University of Sao Paulo, Brazil. He is a regular reviewer and author in his research field, which includes data science, machine learning, content-based data retrieval, visualization, and the application of such techniques in the medic, agriculture, and e-learning domains, contributing to publications in major journals and conferences.

Jimeng Sun in currently a professor at the Computer Science Department, University of Illinois, Urbana-Champaing. Previously, he was with the College of Computing, Georgia Institute of Technology. His research interest includes Artificial Intelligence for healthcare, deep learning for drug discovery, clinical trial optimization, computational phenotyping, clinical predictive modeling, treatment recommendation, and health monitoring.

CHAPTER 5 CONCLUSION

This thesis contributes to a thorough process for analyzing spatial and temporal data from the perspective of computer science applied to (A) Street Network Analysis, (B) Human Mobility Forecasting, and (C) Dynamic Processes Modeling in Time. The reported results emerge from an interchangeable underlying abstract formalism, the graph. Through such a structure, we contribute to the analysis of data on multiple scales, from a cities' street network at a time, going through the analysis of pairs of cities, and ending with the simultaneous analysis of whole countries expressed as time series. Accordingly, these contributions (as highlighted in Chapter 1) derived from the following general hypothesis:

General Hypothesis

The analytic processing by means of complex networks and graph metrics combined with artificial intelligence techniques from the computational intelligence realm can expand the comprehension and, consequently, the capacity of modeling and forecasting different human phenomena, providing us with information for acting on the network topology (i.e., street networks from maps), dynamics (i.e., pendular migration between cities), and inner processes (i.e., pandemics progression over time).

The results we achieved in the paper *Detecting multi-scale distance-based inconsistencies in cities through complex-networks* [Spadon *et al.* 2018] reinforce the potential of complex networks in expanding the comprehension and the capacity of intervention in cities through the topology and geometry of street meshes. Additionally, through the contribution described in *Reconstructing commuters network using machine learning and urban indicators* [Spadon *et al.* 2019], we show how to describe and predict the collective human behavior observed within pairs of cities using machine learning together with the explanation of the role of urban indicators in commuting patterns. Finally, in *Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning* [Spadon *et al.* 2021], we go beyond mobility and show how to shape dynamic processes observed among humans across space and time with a novel graph-inspired deep neural network architecture. We contend that these contributions mark strong evidence that advances have been made to understand challenging and relevant problems related to human phenomena.

5.1 Further Discussion

5.1.1 Street Network Analysis

Chapter 2 delineated analytical formulations that originated algorithms to inspect the design of street meshes aiming to provide redesign decisions for urban planners. Despite being our central motivation, there are significant challenges in redesigning a city. Changing the network topology will alter its centrality, which reshapes regions that attract vehicles and people. Nevertheless, our techniques are to be used in the initial design of a city and in precise alterations that are acceptable during the long-term existence of an urban center. Consequently, we contribute with the definition of a concept based on inherent problems to urban structures caused by the miss-allocation of points of interest, two algorithms to track and reduce inconsistent vertices, and two case studies that show how these results can aid planners in real-world problems from urban systems.

It is noteworthy that, during experimentation, we have assumed (i) transportation only through cities' streets and (ii) a city with uniform population distribution. Notwithstanding, our results hold for scenarios where these assumptions fail, as we can adapt edge weights following the type of displacement rather than using street distance. This is because our methods use a general concept of weight, and when providing additional information, such weight can assume any quantitative value. Nonetheless, whenever more variables are considered during the street network analysis, the set of inconsistencies within a city will require a specialist's validation rather than being a self-explanatory result.

5.1.2 Human Mobility Forecasting

Chapter 3 advanced with a model capable of forecasting and understanding the pendular migration pattern among Brazilian cities using governmental census data. We contributed to the prediction of fluxes using machine learning algorithms, which, through urban indicators, could predict the existence of the flux between two cities using classification and the number of people commuting between them through regression. Our model achieved state-of-the-art performance during experimentation, showing 90.4% of accuracy in the link prediction task and 77.6% of the R^2 Score in the weighted link prediction.

It is noteworthy that the related literature describes models primarily based on population indicators and the distance between cities. On the other hand, our results indicate that only these two variables are not enough to model human mobility patterns.

Contrarily, we verified that many other variables that describe the quality of life and work in cities are necessary to build a commuters network that resembles real-world ones. The analysis of the results enabled us to answer questions about the factors impacting commuters' fluxes between cities and determine potential reasons that lead people to live in cities other than those they work. Our results reveal that these reasons are primarily associated with the city of housing's quality of life and the city of work's economic growth.

5.1.3 Dynamic Processes Modeling in Time

Chapter 4 extended the Multivariate Time Series concept by delineating its Multiple Multivariate Time Series version, which refers to a stacked multivariate time series that share variables and timestream. That means we have the same variables observed during equal timestamps recorded synchronously for different samples. When looking at all the samples at once, the multivariate time-series forecasting problem yields an additional data dimension, and each time series becomes a multivariate sample of a higher-dimensional time-series forecasting problem. When dealing with higher-dimensional data, traditional artificial intelligence algorithms perform as ensembles by focusing on a single dimension of the problem, limiting the information shared about different yet related time series.

As a result of tackling the problem on a higher dimension, we created the Recurrent Graph Evolution Neural Network (ReGENN), a graph-inspired time-aware auto-encoder with linear and non-linear components working in parallel to provide forecasting capabilities based on observations from the past. ReGENN uses a pair of Graph Soft Evolution (GSE) layers, which stands for a graph-based learning-representation layer that improves the encoding and decoding processes by learning a shared graph over several time series and timestamps. ReGENN was confronted with numerous ensembles and traditional statistical methods, confirming improvements of up to 64.87% over the competing algorithms.

ReGENN operates on a three-dimensional space from the input to the output due to being designed to work with higher-dimensional time-series data. Consequently, it can operate not only across time but also over space when spatiotemporal data is provided. That means, ReGENN learns by looking at observations from the past, which can be related to one or many time series of different samples recorded in different places across the globe. However, a set of time series can be used with ReGENN if, and only if, they are related to some extent. For example, the sound wave of two birds singing and sharing the same habitat is feasible for ReGENN's input, but not birds singing at different habitats.

5.2 Practical Implications

Science has a stringent methodological fashion that aims for numerical results presentation and technical discussions. In such a way, the broad understanding about the relevance of the scientific research does not achieve a more comprehensive range of people, limiting the general perception about the usefulness of science. With that in mind, this section presents a brief discussion about the practical implication of this thesis using a less-academic way of writing to ease the understanding of the results to the general public.

5.2.1 Street Network Analysis

In Chapter 2, we present the *Urban Inconsistencies*, our first contribution to the analysis of cities using computational methods. In a city, there are urban planning imperfections that affect nearby citizens' locomotion, which can be found on the streets blueprint. The methods we devised are able to detect such imperfections by means of mathematical formulations. Along with that chapter, we show how traffic engineers and urban planners can use this new concept in practice. The idea is to guide the improvement of a city's urban mobility both for pedestrians and for automobile vehicles. The output of our methods, despite objective, must be analyzed by specialists, such as traffic engineers and urban planners, who will ponder other factors related to the big culture but that did reach the algorithms' input. Specialists will make the best decisions for the local community based on their knowledge, providing unique designs for each city.

Urban Inconsistencies have great potential for use in more general scenarios, but it also depends on the experts' knowledge. For example, we can use this concept to better distribute goods across distribution centers around a country, thus minimizing costs related to buying and selling merchants. It can also help to plan the location of vaccination centers, providing information to the government on distributing vaccines, which are scarce in many places worldwide. In general, our results can be used with spatial data related to locomotion through urban streets, roads, highways, railways, bicycle paths, and other street-like structures to analyze potential mobility issues. Regardless of what is being analyzed, the ultimate goal will always be the same, providing information for specialists to make more accurate decisions. In the long run, these decisions tend to generate lower costs for constructing and maintaining street-like structures. Cost reduction, in this case, can indicate a better use of taxes or the reduction of expenses of a private company.

5.2.2 Human Mobility Forecasting

In Chapter 3, we show how one can use machine learning to solve a decades-old migration forecasting problem. Migration is an activity present in human history and something that we continually experience at different intensity levels. For example, when moving between cities, states, and countries. Through this research, we found what leads someone to migrate at the same time that we provide a way to forecast the intensity of such migration. Such reasons may seem obvious in some cases, but the question is broad and related to a population subgroup. Thus, in that chapter, we addressed the specific case of commuting workers between cities, describing people working in one city but residing in another. Throughout the chapter, we review several theoretical approaches that carry out similar problems to this one, pointing out their pros and cons compared to our proposal, which focuses on machine learning. As a result, we show how to predict the number of workers migrating between Brazilian cities in 2010 through governmental census data. Furthermore, we show that the reasons that lead Brazilians to work in one city and reside in another are directly related to the job opportunity and the cost of living. As a result, people tend to work in cities with more job opportunities, live in others with a lower living cost, and daily commute to work. Therefore, comprehending migratory phenomena have a significant impact on the understanding of populations worldwide. By demonstrating the potential of artificial intelligence in solving this problem, we open doors for revisiting problems as old as this one bringing more knowledge to our society.

The understanding of migration has a comprehensive economic impact on regional planning. For example, it allows a better understanding of the concentration and distribution of wealth among the population. Besides, it potentially explains other phenomena such as crime and unemployment that may be connected with the lack of jobs in particular cities. More broadly, we can extrapolate the proposed solution to a range of other problems, such as buying and selling goods, international trade between countries, and the spread of epidemics between places. As is the case of migration between cities, we can find the relationship between two entities in the human brain, nature's food chains, interpersonal and professional relationships, among others. Modeling these relations through artificial intelligence expands our knowledge about society and ourselves, implying more information for decision-making, whether focused on public or private policies.

5.2.3 Dynamic Processes Modeling in Time

In Chapter 4, we introduce a new neural network designed to predict time series and spatiotemporal series, which are time-varying and space-time-varying data. A neural network is a computational technique derived from machine learning[1] that mimics a brain neuron's functioning, aiming to predict future not-yet-seen data by observing past data. As a result of our study, we created a computational technique from a theoretical formulation that provides a different way of looking at time-series forecasting problems. This novel perspective describes that one needs to observe multiple series of events that happen mutually and synchronously to predict the future more accurately. For example, when monitoring an endangered animal, one requires understanding the animal's habitat while considering direct and indirect predators. Thus, we transferred such a perception of time to a scenario where computers, through a neural network, can analyze the relationships between different series-like data. This way, we achieved highly accurate predictions as compared with real-world circumstances. Specifically, we contributed with a novel neural

[1] Also referred to as *deep learning.*

network technique that has shown great potential in predicting the evolution of pandemics worldwide, forecasting climate at a country level, and monitoring the health status of ICU-admitted patients. However, because time is a basic premise for human existence, the applications of a neural network in forecasting events related to time series are manifold.

We can use data of several natures through multiple variables varying over time. This is the case of predicting the evolution of migratory patterns between governmental censuses or monitoring the human mobility improvement over cities between different municipal administrations. Regardless of the application, one can expect our model to highly accurately describe the future based on the past; at least, more accurately than the 48 algorithms against which we compared our method. Because forecasting future events is one of the basic premises for decision-making related to urban planning, consumer behavior forecasting, market trend analysis, among other applications, our neural network can be thoroughly valuable by both the private and public sectors. Regardless of the purpose of use, knowing glances of the future allows one to make clear decisions, indirectly optimizing decision-making time and better allocating the available resources.

5.3 Open Problems

Regarding the analysis of street meshes using complex networks, there are still open problems related to identifying the backbone of a city, which is linked to the study of Scellato *et al.* 2006. The backbone consists of a group of streets that are of central importance to a city's locomotion. Due to its synthesizer nature, such a backbone allows a hierarchical delineation of a city's traffic, helping identify streets more susceptible to problems. A first metric that can be explored to solve that task is the Spanning Tree Centrality [Mavroforakis *et al.* 2015, Teixeira, Santos and Francisco 2016], which computes all the spanning trees based on shortest paths to quantify the importance of the edges of a graph according to how many times each edge appears in a different spanning tree. This approach can be performed iteratively, defining the level of importance of the edges in:

- ◇ *Primary streets:* the main streets of the city with the highest traffic of people and vehicles, characterizing streets that might be more sensitive to locomotion issues;
- ◇ *Secondary streets:* consist of capillary streets that are connected to the arterial (primary) streets, which are used to avoid and distribute the traffic of vehicles; and,
- ◇ *Ternary streets:* these are streets responsible for routing inside the neighborhoods.

Through the backbone's hierarchic delineation, it is possible to produce short summaries of network traffic, enabling drivers to promptly understand their overall organization, find routes more quickly, or even explore an unknown city more efficiently. Such a problem is still open as the related algorithms are computationally expensive, requiring too much time to provide a result and making it difficult to experiment on medium and large cities.

Regarding dynamic processes like human mobility, which impact several layers of the society, from infrastructural planning and economic decisions to the spread of epidemics, one open problem is to infer urban indicators through graph metrics. A model capable of inferring urban indicators can predict indicators in regions where they are unknown, which would reduce costs related to obtaining such information and expand our understanding of cities worldwide. In this experiment, one should focus on predicting urban indicators related to Gross Domestic Product, Unemployment Rate, and Traffic Accidents, all of which are broadly collected by a range of countries, aiding in the model validation. To this end, one can use traditional regression techniques and classification algorithms adapted to regression tasks, such as Nearest Neighbors [Aha, Kibler and Albert 1991], Decision Trees [Breiman 2017], and Adaptive Boosting [Freund and Schapire 1995]; an approach that can also leverage deep learning techniques [LeCun, Bengio and Hinton 2015], especially graph-inspired neural networks architectures [Zhang, Cui and Zhu 2020].

Accordingly, it becomes feasible to use these indicators to infer global human mobility. The greatest challenge of this approach is related to the discrepant characteristics of cities, such that using only Brazilian data will probably not be enough to estimate fluxes among cities around the world. However, cluster analysis can help because international cities with similar topology to the Brazilian ones might share similarities related to their urban indicators. This way, it is possible to devise models focused on predicting urban indicators and mobility fluxes among clusters of similar cities, explaining how urban indicators and graph metrics act upon global human-movement phenomena. Contributions in this area have the potential to evaluate the quality of census data (*i.e.*, contrasting the manually collected with the predicted data when the model is accurate enough), predict the fluxes of commuters in years between governmental censuses, and predict fluxes where such information is unknown or private to the general public.

Concerning the analysis of street networks, the validation of the result can be through analytic methods, mainly statistical ones. In this case, one should evaluate the significance of the findings, for instance, through null models [Roxburgh and Matsuki 1999] and confidence intervals [Efron 1987]. Examples in these areas refer to the generation of random networks [Villas Boas, Rodrigues and da Fontoura Costa 2009] that enable comparing the number of times a pattern appears in the real network and several random networks. Another technique to be used is bootstrapping [Efron 1992], a random sampling evaluation with replacement that defines confidence intervals of a given statistical evaluation metric. To analyze the correlation between graph metrics and urban indicators, we highlight the correlation of Pearson, Spearman, and Kendall [Chiang 2003]. It is noteworthy that they can be evaluated according to the behavior of the probability distribution fitting (*e.g.*, Poisson, Gaussian, and Pareto) of both urban indicators and graph metrics.

Concerning machine and deep learning, classifiers and regressors must be verified in the last step of the machine or statistical learning processes. This verification unfolds into a set of metrics that translates into performance estimates. The quality of the classifiers' predictions can be apprised through the accuracy score, confusion matrix, and F-Measure [Rijsbergen 1979]. Similarly, regressors' prediction can be analyzed through the R^2 Score, mean squared error, and mean absolute error [James *et al.* 2013]. Together, these metrics provide a panorama of accuracy, miss-classification, deviation from the expected output, and behavior against the dataset size. They guide the hyperparameters tuning process towards better outputs and provide a verdict on success or failure.

5.4 Scientific Production

This section provides a list of Spadon's publications from 2017 to 2021, which were divided into first-authored publications (main research line) and co-authored publications (collaborations with the research group). More information about these publications and their full texts is available in his Google Scholar, ReserachGate, and ORCID profiles.

5.4.1 First-Authored Publications

Journal Articles

1. **Spadon G.**, Hong S., Brandoli B., Matwin S., Rodrigues-Jr J. F., and Sun J., *Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning.* Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2021;
2. **Spadon G.**, Carvalho A. C. P. L. F., Rodrigues-Jr J. F., and Alves L. G. A., *Reconstructing Commuters Network using Machine Learning and Urban Indicators.* Scientific Reports. Springer Nature, 2019; and,
3. **Spadon G.**, Brandoli B., Eler D. M., and Rodrigues-Jr J. F., *Detecting Multi-Scale Distance-Based Inconsistencies in Cities through Complex-Networks.* Journal of Computational Science. Elsevier, 2018.

Book Chapters

4. **Spadon G.**, Gimenes G., and Rodrigues-Jr J. F., *Topological Street-Network Characterization through Feature-Vector and Cluster Analysis.* International Conference on Computational Science. Springer, 2018;
5. **Spadon G.**, Brandoli B., Eler D. M., and Rodrigues-Jr J. F., *A Distance-Based toolset to Track Inconsistent Urban Structures Through Complex-Networks*[2]. Inter-

[2] Awarded paper – Invited for publication at the *Journal of Computational Science.*

national Conference on Computational Science. Springer, 2018; and,

6. **Spadon G.**, Scabora L. C., Araujo M. V., Oliveira P. H., Brandoli B., Sousa E. P. M., Traina-Jr C., and Rodrigues-Jr J. F., *Complex-Network Tools to Understand the Behavior of Criminality in Urban Areas*[3]. Information Technology-New Generations. Springer, 2018.

Conference Papers

7. **Spadon G.** and Rodrigues-Jr J. F., *Computer-Assisted City Touring for Explorers*. Workshop on Big Social Data and Urban Computing (BiDU) co-located with the 44th International Conference on Very Large Data Bases (VLDB). CEUR-WS, 2018;
8. **Spadon G.**, Gimenes G., and Rodrigues-Jr J. F., *Identifying Urban Inconsistencies via Street Networks*[3]. Procedia Computer Science 108. Elsevier, 2017; and,
9. **Spadon G.**, Scabora L. C., Oliveira P. H., Araujo M.V., Brandoli B., Sousa E. P. M., Traina-Jr C., and Rodrigues-Jr J. F., *Behavioral Characterization of Criminality Spread in Cities*[3]. Procedia Computer Science 108. Elsevier, 2017.

Public Communication

10. *Redes Complexas Ajudam a Entender Movimento Diário da População Entre Cidades*. Jornal da USP, 2019. Available online at <https://bit.ly/3f1T8jh>;
11. *Using Computational Models to Improve Street Planning*. Science Trends, 2019. Available online at <https://bit.ly/2S9pujp>;
12. *Reconstructing Commuter Networks using Machine Learning and Urban Indicators*. Science Trends, 2019. Available online at <https://bit.ly/3u2ofzt>; and,
13. *Redes de Ruas Produzidas em Computador Ajudam a Planejar Trânsito*. Jornal da USP, 2018. Available online at <https://bit.ly/3wz5yp5>.

5.4.2 Co-Authored Publications

Journal Articles

14. Brandoli B., Geus A. R., Souza J. R., **Spadon G.**, Soares-Jr A., Rodrigues-Jr J. F., Komorowski. J, Matwin S., *Aircraft Fuselage Corrosion Detection using Artificial Intelligence*. Sensors. MDPI, 2021;
15. Rodrigues-Jr J. F., Gutierrez M., **Spadon G.**, Brandoli B., Amer-Yahia S., *LIG-Doctor: Efficient Patient Trajectory Prediction using Bidirectional Minimal Gated-Recurrent Networks*. Information Sciences. Elsevier, 2021;

[3] Results from the Masters' dissertation.

16. Brandoli B., **Spadon G.**, Esau T., Hennessy P., Carvalho A. C. P. L. F., Rodrigues-Jr J. F., Amer-Yahia S., *DropLeaf: A Precision Farming Smartphone Application for Measuring Pesticide Spraying Methods.* Journal of Computers and Electronics in Agriculture. Elsevier, 2020;

17. Rodrigues-Jr J. F., **Spadon G.**, Brandoli B., Amer-Yahia S., *Patient Trajectory Prediction in the Mimic-III Dataset, Challenges and Pitfalls.* arXiv, 2019;

18. Correia R. C. M., **Spadon G.**, Gomes P. H. A., Eler D. M., Garcia R. E., and Olivete-Jr C., *Hadoop Cluster Deployment: A Methodological Approach.* Information. MDPI, 2018; and,

19. Scabora L. C., Oliveira P. H., **Spadon G.**, Kaster D., Rodrigues-Jr J. F., Traina A., Traina-Jr C., *Cutting-Edge Relational Graph Data Management with Edge-k: From One to Multiple Edges in the Same Row.* Journal of Information and Data Management. SBC, 2018.

Book Chapters

20. Rocha J. E., Olivete-Jr C., Gomes P. H. A., Garcia R. E., Correia R. C. M., **Spadon G.**, and Eler D. M., *Internet-Based Education: A New Milestone for Formal Language and Automata Courses.* Information Technology-New Generations. Springer, 2018; and,

21. Correia R. C. M., **Spadon G.**, Eler D. M., Olivete-Jr C., and Garcia R. E., *Teaching Distributed Systems using Hadoop.* Information Technology-New Generations. Springer, 2018.

Conference Papers

22. Scabora L. C., **Spadon G.**, Cazzolato M. T., Kaster D. S., Traina A. J., Rodrigues-Jr J. F., Traina-Jr C. *SHARq: Sharing Recursive Queries in Relational Databases.* Proceedings of the 36th Annual ACM Symposium on Applied Computing. ACM, 2021;

23. Scabora L. C., **Spadon G.**, Oliveira P. H., Rodrigues-Jr J. F., Traina-Jr C., *Enhancing Recursive Graph Querying on RDBMS with data Clustering Approaches.* Proceedings of the 35th Annual ACM Symposium on Applied Computing. ACM, 2020;

24. Rodrigues-Jr J. F., **Spadon G.**, Brandoli B., Amer-Yahia S., *Lig-Doctor: Real-World Clinical Prognosis using a Bi-Directional Neural Network.* 33rd International Symposium on Computer-Based Medical Systems (CBMS). IEEE, 2020;

25. Brandoli B., **Spadon G.**, Arruda M. D. S., Gonçalves W. N., Carvalho A. C. P. L. F., and Rodrigues-Jr J. F., *A Smartphone Application to Measure the Quality of Pest Control Spraying Machines via Image Analysis.* Proceedings of the 33rd Annual ACM Symposium on Applied Computing. ACM, 2018;
26. Arruda M. D. S., **Spadon G.**, Rodrigues-Jr J. F., Gonçalves W. N., and Brandoli B., *Recognition of Endangered Pantanal Animal Species using Deep Learning methods.* International Joint Conference on Neural Networks (IJCNN). IEEE, 2018;
27. Nesso-Jr M. R., Cazzolato M. T., Scabora L. C., Oliveira P. H., **Spadon G.**, Souza J. A., Oliveira W. D., Chino D. Y. T., Rodrigues-Jr J. F., Traina A. J. M., and Traina-Jr C., *RAFIKI: Retrieval-Based Application for Imaging and Knowledge Investigation.* 31st International Symposium on Computer-Based Medical Systems (CBMS). IEEE, 2018;
28. Ferreira L. D., **Spadon G.**, Carvalho A. C. P. L. F., and Rodrigues-Jr J. F., *A Comparative Analysis of the Automatic Modeling of Learning Styles Through Machine Learning Techniques.* Frontiers in Education Conference (FIE). IEEE, 2018;
29. Marques L. F., Correia R. C. M., **Spadon G.**, Eler D. M., Olivete-Jr C., and Garcia R. E., *Databases Available through APIs using Restify: Characteristics, Programming Models, and Benchmarks.* 12th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2017;
30. Caldeira D. C., Correia R. C. M., **Spadon G.**, Eler D. M., Olivete-Jr C., and Garcia R. E., *Data Mining on Linkedin Data to Define Professional Profile via MineraSkill Methodology.* 12th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2017;
31. Almeida C. E. M., Correia R. C. M., Eler D. M., Olivete-Jr C., and Garcia R. E., Scabora L. C., and **Spadon G.**, *Prediction of Winners in MOBA Games.* 12th Iberian Conference on Information Systems and Technologies. IEEE, 2017; and,
32. Gomes P. H. A., Garcia R. E., **Spadon G.**, Eler D. M., Olivete-Jr C., and Correia R. C. M., *Teaching Software Quality via Source Code Inspection Tool.* Frontiers in Education Conference (FIE). IEEE, 2017.

Technical Reports

33. Oishi, C. M., Amaral, F. V. G., França, H. L., Nakata, W. H., Aguiar, D. A., Santos, G. F. O., Medeiros, D. O., **Spadon, G.**, Rodrigues-Jr, J. F., Martínez, J. M., Santos, L. T., and Soares Filho, D. M., *Neural networks for seismic data inversion.* Mathematics in Industry Reports. Cambridge Open Engage, 2021.

REFERENCES

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-based learning algorithms. **Machine Learning**, Springer, v. 6, n. 1, p. 37–66, jan 1991. ISSN 0885-6125. Citation on page 85.

BALCAN, D.; COLIZZA, V.; GONÇALVES, B.; HU, H.; RAMASCO, J. J.; VESPIGNANI, A. Multiscale mobility networks and the spatial spreading of infectious diseases. **Proceedings of the National Academy of Sciences**, National Academy Sciences, v. 106, n. 51, p. 21484–21489, dec 2009. ISSN 0027-8424. Citation on page 20.

BARABÁSI, A.-L. **Network Science**. [S.l.]: CUP, 2016. Citation on page 17.

BARABÁSI, A.-L.; ALBERT, R. Emergence of Scaling in Random Networks. **Science**, American Association for the Advancement of Science (AAAS), v. 286, n. 5439, p. 509–512, oct 1999. ISSN 0036-8075. Citation on page 17.

BARBOSA, H.; BARTHELEMY, M.; GHOSHAL, G.; JAMES, C. R.; LENORMAND, M.; LOUAIL, T.; MENEZES, R.; RAMASCO, J. J.; SIMINI, F.; TOMASINI, M. Human mobility: Models and applications. **Physics Reports**, Elsevier, v. 734, p. 1–74, mar 2018. ISSN 03701573. Citation on page 20.

BARTHELEMY, M. From paths to blocks: New measures for street patterns. **Environment and Planning B: Urban Analytics and City Science**, v. 44, n. 2, p. 256–271, mar 2017. ISSN 2399-8083. Citation on page 19.

BARTHÉLEMY, M.; FLAMMINI, A. Modeling Urban Street Patterns. **Physical Review Letters**, American Physical Society (APS), v. 100, n. 13, apr 2008. ISSN 0031-9007. Citation on page 19.

BATTAGLIA, P. W.; PASCANU, R.; LAI, M.; REZENDE, D. J.; KAVUKCUOGLU, K. Interaction networks for learning about objects, relations and physics. In: LEE, D. D.; SUGIYAMA, M.; LUXBURG, U. von; GUYON, I.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**. [S.l.: s.n.], 2016. p. 4502–4510. Citation on page 20.

BATTY, M. The Size, Scale, and Shape of Cities. **Science**, American Association for the Advancement of Science, v. 319, n. 5864, p. 769–771, feb 2008. ISSN 0036-8075. Citation on page 20.

BATTY, M.; LONGLEY, P. A. **Fractal cities: a geometry of form and function**. [S.l.]: Academic press, 1994. Citation on page 20.

BENENSON, I.; TORRENS, P. M. **Geosimulation: Automata-based Modeling of Urban Phenomena**. [S.l.]: John Wiley & Sons, 2004. 1–287 p. ISBN 9780470843499. Citation on page 20.

BETTENCOURT, L. M.; LOBO, J.; HELBING, D.; KÜHNERT, C.; WEST, G. B. Growth, innovation, scaling, and the pace of life in cities. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 104, n. 17, p. 7301–7306, apr 2007. ISSN 0027-8424. Citation on page 20.

BOCCALETTI, S.; LATORA, V.; MORENO, Y.; CHAVEZ, M.; HWANG, D. Complex networks: Structure and dynamics. **Physics Reports**, Elsevier BV, v. 424, n. 4-5, p. 175–308, feb 2006. ISSN 03701573. Citation on page 17.

BOEING, G. **Methods and Measures for Analyzing Complex Street Networks and Urban Form**. 1–236 p. Phd Thesis (PhD Thesis) — University of California, Berkeley, 2017. Citation on page 19.

BOEING, G. A multi-scale analysis of 27,000 urban street networks: Every US city, town, urbanized area, and Zillow neighborhood. **Environment and Planning B: Urban Analytics and City Science**, v. 47, n. 4, p. 590–608, may 2018. ISSN 2399-8083. Citation on page 20.

______. The Morphology and Circuity of Walkable and Drivable Street Networks. In: **Modeling and Simulation in Science, Engineering and Technology**. [S.l.: s.n.], 2019. p. 271–287. Citation on page 19.

BREIMAN, L. **Classification and regression trees**. [S.l.]: Routledge, 2017. Citation on page 85.

CARDILLO, A.; SCELLATO, S.; LATORA, V.; PORTA, S. Structural properties of planar graphs of urban street patterns. **Physical Review E**, American Physical Society (APS), v. 73, n. 6, jun 2006. ISSN 1539-3755. Citation on page 19.

CARTER, M.; HOWARD, M. P.; OWENS, N. D.; REGISTER, D.; KENNEDY, J.; PECHEUX, K. K.; NEWTON, A. *et al.* Effects of catastrophic events on transportation system management and operations: Howard street tunnel fire baltimore city, maryland - july 18, 2001. United States. Joint Program Office for Intelligent Transportation Systems, 2002. Citation on page 20.

CHENG, Z.; YANG, Y.; WANG, W.; HU, W.; ZHUANG, Y.; SONG, G. Time2Graph: Revisiting Time Series Modeling with Dynamic Shapelets. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2020. v. 34, n. 04, p. 3617–3624. ISSN 2374-3468. Citation on page 18.

CHIANG, C. L. **Statistical Methods of Analysis**. [S.l.]: World Scientific, 2003. ISBN 978-981-238-309-9. Citation on page 85.

COLIZZA, V.; BARRAT, A.; BARTHÉLEMY, M.; VESPIGNANI, A. The role of the airline transportation network in the prediction and predictability of global epidemics. **Proceedings of the National Academy of Sciences**, National Academy Sciences, v. 103, n. 7, p. 2015–2020, feb 2006. ISSN 0027-8424. Citation on page 20.

CORCORAN, P.; JILANI, M.; MOONEY, P.; BERTOLOTTO, M. Inferring semantics from geometry. In: **Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems**. New York, NY, USA: Association for Computing Machinery (ACM), 2015. p. 1–10. ISBN 9781450339674. Citation on page 19.

COSTA, L. F.; TRAVENÇOLO, B. A. N.; VIANA, M. P.; STRANO, E. On the efficiency of transportation systems in large cities. **EPL (Europhysics Letters)**, IOP Publishing, v. 91, n. 1, jul 2010. ISSN 0295-5075. Citation on page 19.

CRUCITTI, P.; LATORA, V.; PORTA, S. Centrality measures in spatial networks of urban streets. **Physical Review E**, American Physical Society (APS), v. 73, n. 3, mar 2006. ISSN 1539-3755. Citation on page 19.

CUI, C.; HAN, Z. Spatial patterns of retail stores using POIs data in Zhengzhou, China. In: **2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM)**. [S.l.]: Institute of Electrical and Electronics Engineers (IEEE), 2015. p. 88–92. ISBN 978-1-4799-7748-2. Citation on page 18.

DAVIES, T.; JOHNSON, S. D. Examining the Relationship Between Road Structure and Burglary Risk Via Quantitative Network Analysis. **Journal of Quantitative Criminology**, Springer Nature, v. 31, n. 3, p. 481–507, sep 2015. ISSN 0748-4518. Citation on page 19.

DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. In: LEE, D. D.; SUGIYAMA, M.; LUXBURG, U. von; GUYON, I.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**. [S.l.: s.n.], 2016. p. 3837–3845. Citation on page 20.

DIBBLE, J.; PRELORENDJOS, A.; ROMICE, O.; ZANELLA, M.; STRANO, E.; PAGEL, M.; PORTA, S. On the origin of spaces: Morphometric foundations of urban form evolution. **Environment and Planning B: Urban Analytics and City Science**, v. 46, n. 4, p. 707–730, may 2017. ISSN 2399-8083. Citation on page 19.

DUVENAUD, D.; MACLAURIN, D.; AGUILERA-IPARRAGUIRRE, J.; GÓMEZ-BOMBARELLI, R.; HIRZEL, T.; ASPURU-GUZIK, A.; ADAMS, R. P. Convolutional networks on graphs for learning molecular fingerprints. In: CORTES, C.; LAWRENCE, N. D.; LEE, D. D.; SUGIYAMA, M.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada**. [S.l.: s.n.], 2015. p. 2224–2232. Citation on page 20.

EFRON, B. Better Bootstrap Confidence Intervals. **Journal of the American Statistical Association**, Taylor & Francis Group, v. 82, n. 397, p. 171–185, mar 1987. ISSN 0162-1459. Citation on page 85.

______. Bootstrap methods: another look at the jackknife. In: **Breakthroughs in statistics**. [S.l.]: Springer, 1992. p. 569–593. Citation on page 85.

EUBANK, S.; GUCLU, H.; Anil Kumar, V. S.; MARATHE, M. V.; SRINIVASAN, A.; TOROCZKAI, Z.; WANG, N. Modelling disease outbreaks in realistic urban social networks. **Nature**, Nature Publishing Group, v. 429, n. 6988, p. 180–184, may 2004. ISSN 0028-0836. Citation on page 20.

EULER, L. Solutio Problematis ad Geometriam Situs Pertinentis. **Commentarii academiae scientiarum Petropolitanae**, v. 8, p. 128–140, 1741. Citation on page 17.

FREUND, Y.; SCHAPIRE, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, Elsevier, v. 904, n. 1, p. 23–37, 1995. ISSN 16113349. Citation on page 85.

GALBRUN, E.; PELECHRINIS, K.; TERZI, E. Urban navigation beyond shortest route: The case of safe paths. **Information Systems**, Elsevier BV, v. 57, p. 160–171, apr 2016. ISSN 03064379. Citation on page 19.

GIL, J. Street network analysis “edge effects”: Examining the sensitivity of centrality measures to boundary conditions. **Environment and Planning B: Urban Analytics and City Science**, SAGE Publications, v. 44, n. 5, p. 819–836, sep 2016. ISSN 2399-8083. Citation on page 19.

GILMER, J.; SCHOENHOLZ, S. S.; RILEY, P. F.; VINYALS, O.; DAHL, G. E. Neural message passing for quantum chemistry. In: TEH, Y. W. (Ed.). **Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017**. [S.l.]: PMLR, 2017. (Proceedings of Machine Learning Research, v. 70), p. 1263–1272. Citation on page 20.

GOH, S.; CHOI, M. Y.; LEE, K.; KIM, K.-m. How complexity emerges in urban systems: Theory of urban morphology. **Physical Review E**, American Physical Society (APS), v. 93, n. 5, may 2016. ISSN 2470-0045. Citations on pages 18 and 19.

GONZÁLEZ, M. C.; HIDALGO, C. A.; BARABÁSI, A.-L. Understanding individual human mobility patterns. **Nature**, Nature publishing group, v. 458, n. 7235, p. 238–238, mar 2009. ISSN 0028-0836. Citation on page 17.

HAMILTON, W. L.; YING, R.; LESKOVEC, J. Representation learning on graphs: Methods and applications. **IEEE Data Eng. Bull.**, Institute of Electrical and Electronics Engineers (IEEE), v. 40, n. 3, p. 52–74, 2017. Citation on page 20.

HAMILTON, W. L.; YING, Z.; LESKOVEC, J. Inductive representation learning on large graphs. In: GUYON, I.; LUXBURG, U. von; BENGIO, S.; WALLACH, H. M.; FERGUS, R.; VISHWANATHAN, S. V. N.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA**. [S.l.: s.n.], 2017. p. 1024–1034. Citation on page 20.

HILLIER, B. **Space is the Machine: A Configurational Theory of Architecture**. [S.l.]: Space Syntax, 2007. ISBN 9780955622403. Citation on page 19.

HOSHEN, Y. VAIN: attentional multi-agent predictive modeling. In: GUYON, I.; LUXBURG, U. von; BENGIO, S.; WALLACH, H. M.; FERGUS, R.; VISHWANATHAN, S. V. N.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA**. [S.l.: s.n.], 2017. p. 2701–2711. Citation on page 20.

HU, W.; YANG, Y.; CHENG, Z.; YANG, C.; REN, X. Time-Series Event Prediction with Evolutionary State Graph. In: **Proceedings of the 14th ACM International Conference on Web Search and Data Mining**. New York, NY, USA: Association for

Computing Machinery (ACM), 2021. p. 580–588. ISBN 9781450382977. Citation on page 18.

IACUS, S. M.; SANTAMARIA, C.; SERMI, F.; SPYRATOS, S.; TARCHI, D.; VESPE, M. Human mobility and COVID-19 initial dynamics. **Nonlinear Dynamics**, v. 101, n. 3, p. 1901–1919, aug 2020. ISSN 0924-090X. Citation on page 18.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An introduction to statistical learning**. [S.l.]: Springer, 2013. Citation on page 86.

KANG, C.; SOBOLEVSKY, S.; LIU, Y.; RATTI, C. Exploring human movements in Singapore: A comparative analysis based on mobile phone and taxicab usages. In: **Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing - UrbComp'13**. New York, New York, USA: Association for Computing Machinery (ACM), 2013. ISBN 9781450323314. Citation on page 18.

KEARNES, S. M.; MCCLOSKEY, K.; BERNDL, M.; PANDE, V. S.; RILEY, P. Molecular graph convolutions: moving beyond fingerprints. **J. Comput. Aided Mol. Des.**, v. 30, n. 8, p. 595–608, 2016. Citation on page 20.

KIM, J.; WILHELM, T. What is a complex graph? **Physica A: Statistical Mechanics and its Applications**, Elsevier BV, v. 387, n. 11, p. 2637–2652, apr 2008. ISSN 03784371. Citation on page 17.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. In: **5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings**. [S.l.]: OpenReview.net, 2017. Citation on page 20.

KRAEMER, M. U. G.; YANG, C.-H.; GUTIERREZ, B.; WU, C.-H.; KLEIN, B.; PIGOTT, D. M.; PLESSIS, L. du; FARIA, N. R.; LI, R.; HANAGE, W. P.; BROWNSTEIN, J. S.; LAYAN, M.; VESPIGNANI, A.; TIAN, H.; DYE, C.; PYBUS, O. G.; SCARPINO, S. V. The effect of human mobility and control measures on the COVID-19 epidemic in China. **Science**, American Association for the Advancement of Science, v. 368, n. 6490, p. 493–497, may 2020. ISSN 0036-8075. Citation on page 18.

KRUECKEBERG, D. A.; SILVERS, A. L. **Urban planning analysis: methods and models**. [S.l.]: John Wiley & Sons, 1974. Citation on page 20.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, may 2015. ISSN 0028-0836. Citations on pages 18 and 85.

LI, X.; PARROTT, L. An improved Genetic Algorithm for spatial optimization of multi-objective and multi-site land use allocation. **Computers, Environment and Urban Systems**, v. 59, p. 184–194, sep 2016. ISSN 01989715. Citation on page 19.

LI, Y.; TARLOW, D.; BROCKSCHMIDT, M.; ZEMEL, R. S. Gated graph sequence neural networks. In: BENGIO, Y.; LECUN, Y. (Ed.). **4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings**. [S.l.]: OpenReview.net, 2016. Citation on page 20.

LING, X.; WU, L.; WANG, S.; MA, T.; XU, F.; LIU, A. X.; WU, C.; JI, S. Multi-Level Graph Matching Networks for Deep Graph Similarity Learning. **arXiv preprint: 2007.04395**, jul 2020. Citation on page 18.

MARUHASHI, K.; SHIGEZUMI, J.; YUGAMI, N.; FALOUTSOS, C. EigenSP: A More Accurate Shortest Path Distance Estimation on Large-Scale Networks. In: **2012 IEEE 12th International Conference on Data Mining Workshops**. [S.l.]: Institute of Electrical and Electronics Engineers (IEEE), 2012. p. 234–241. ISBN 978-1-4673-5164-5. Citation on page 19.

MAVROFORAKIS, C.; GARCIA-LEBRON, R.; KOUTIS, I.; TERZI, E. Spanning Edge Centrality. In: **Proceedings of the 24th International Conference on World Wide Web**. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015. p. 732–742. ISBN 9781450334693. Citation on page 84.

MYRONENKO, S.; WENGER, A.; ATMAZHOV, S. Capacity analysis of the street and road network of modern regional center. **Odes'kyi Politechnichnyi Universytet. Pratsi**, Odessa National Polytechnic University, n. 3, p. 21–25, dec 2016. ISSN 20762429. Citation on page 19.

NEWMAN, M. E. J. Assortative Mixing in Networks. **Physical Review Letters**, American Physical Society (APS), v. 89, n. 20, oct 2002. ISSN 0031-9007. Citation on page 17.

______. Modularity and community structure in networks. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 103, n. 23, p. 8577–8582, jun 2006. ISSN 0027-8424. Citation on page 17.

OISHI, C. M.; AMARAL, F. V. G.; FRANÇA, H. L.; NAKATA, W. H.; AGUIAR, D. A.; SANTOS, G. F. de O.; MEDEIROS, D. de O.; SPADON, G.; RODRIGUES-JR, J. F.; MARTÍNEZ, J. M.; SANTOS, L. T.; SOARES-FILHO, D. M. Neural networks for seismic data inversion. **Mathematics in Industry Reports**, Cambridge Open Engage, 2021. Citation on page 18.

PAN, G.; QI, G.; ZHANG, W.; LI, S.; WU, Z.; YANG, L. Trace analysis and mining for smart cities: issues, methods, and applications. **IEEE Communications Magazine**, Institute of Electrical and Electronics Engineers (IEEE), v. 51, n. 6, p. 120–126, jun 2013. ISSN 0163-6804. Citation on page 18.

PORTA, S.; CRUCITTI, P.; LATORA, V. The network analysis of urban streets: A dual approach. **Physica A: Statistical Mechanics and its Applications**, Elsevier BV, v. 369, n. 2, p. 853–866, sep 2006. ISSN 03784371. Citation on page 19.

______. The Network Analysis of Urban Streets: A Primal Approach. **Environment and Planning B: Planning and Design**, SAGE Publications, v. 33, n. 5, p. 705–725, oct 2006. ISSN 0265-8135. Citation on page 19.

PORTA, S.; LATORA, V.; WANG, F.; RUEDA, S.; STRANO, E.; SCELLATO, S.; CARDILLO, A.; BELLI, E.; CÀRDENAS, F.; CORMENZANA, B.; LATORA, L. Street Centrality and the Location of Economic Activities in Barcelona. **Urban Studies**, SAGE Publications, v. 49, n. 7, p. 1471–1488, may 2011. ISSN 0042-0980. Citation on page 19.

PORTA, S.; STRANO, E.; IACOVIELLO, V.; MESSORA, R.; LATORA, V.; CARDILLO, A.; WANG, F.; SCELLATO, S. Street Centrality and Densities of Retail and Services in Bologna, Italy. **Environment and Planning B: Planning and Design**, SAGE Publications, v. 36, n. 3, p. 450–465, jun 2009. ISSN 0265-8135. Citation on page 19.

QIU, L.; XIAO, Y.; QU, Y.; ZHOU, H.; LI, L.; ZHANG, W.; YU, Y. Dynamically Fused Graph Network for Multi-hop Reasoning. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2019. p. 6140–6150. ISBN 9781950737482. Citation on page 18.

RAVENSTEIN, E. G. The Laws of Migration. **Journal of the Royal Statistical Society**, JSTOR, v. 52, n. 2, jun 1889. ISSN 09528385. Citation on page 20.

RIJSBERGEN, C. J. V. **Information Retrieval**. 2nd. ed. Newton, MA, USA: Butterworth-Heinemann, 1979. ISBN 0408709294. Citation on page 86.

ROXBURGH, S. H.; MATSUKI, M. The Statistical Validation of Null Models Used in Spatial Association Analyses. **Oikos**, JSTOR, v. 85, n. 1, p. 68–78, apr 1999. ISSN 00301299. Citation on page 85.

SANTORO, A.; RAPOSO, D.; BARRETT, D. G. T.; MALINOWSKI, M.; PASCANU, R.; BATTAGLIA, P. W.; LILLICRAP, T. A simple neural network module for relational reasoning. In: GUYON, I.; LUXBURG, U. von; BENGIO, S.; WALLACH, H. M.; FERGUS, R.; VISHWANATHAN, S. V. N.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA**. [S.l.: s.n.], 2017. p. 4967–4976. Citation on page 20.

SARKAR, S. Spectral (Re)construction of Urban Street Networks: Generative Design Using Global Information from Structure. In: **Design Computing and Cognition'14**. Cham: Springer International Publishing, 2015. p. 41–55. Citation on page 19.

SCARSELLI, F.; GORI, M.; TSOI, A. C.; HAGENBUCHNER, M.; MONFARDINI, G. Computational capabilities of graph neural networks. **IEEE Trans. Neural Networks**, Institute of Electrical and Electronics Engineers (IEEE), v. 20, n. 1, p. 81–102, 2009. Citation on page 20.

SCELLATO, S.; CARDILLO, A.; LATORA, V.; PORTA, S. The backbone of a city. **The European Physical Journal B**, Springer Nature, v. 50, n. 1-2, p. 221–225, mar 2006. ISSN 1434-6028. Citation on page 84.

SCELLATO, S.; FORTUNA, L.; FRASCA, M.; GÓMEZ-GARDEÑES, J.; LATORA, V. Traffic optimization in transport networks based on local routing. **The European Physical Journal B**, Springer Nature, v. 73, n. 2, p. 303–308, jan 2010. ISSN 1434-6028. Citation on page 18.

SCRIPPS, J.; NUSSBAUM, R.; TAN, P.-N.; ESFAHANIAN, A.-H. Link-Based Network Mining. In: **Structural Analysis of Complex Networks**. Boston: Birkhäuser Boston, 2010. p. 403–419. ISBN 9780817647889. Citation on page 19.

SHIODE, S.; SHIODE, N. Network-based space-time search-window technique for hotspot detection of street-level crime incidents. **International Journal of Geographical Information Science**, Informa UK Limited, v. 27, n. 5, p. 866–882, may 2013. ISSN 1365-8816. Citation on page 19.

SONG, C.; QU, Z.; BLUMM, N.; BARABÁSI, A.-L. Limits of Predictability in Human Mobility. **Science**, American Association for the Advancement of Science, v. 327, n. 5968, p. 1018–1021, feb 2010. ISSN 0036-8075. Citation on page 18.

SONG, Y.; MERLIN, L.; RODRIGUEZ, D. Comparing measures of urban land use mix. **Computers, Environment and Urban Systems**, v. 42, p. 1–13, nov 2013. ISSN 01989715. Citation on page 19.

SOPAN, A.; REY, P. J.; SHNEIDERMAN, B. The Dynamics of Web-Based Community Safety Groups: Lessons Learned from the Nation of Neighbors. **IEEE Signal Processing Magazine**, Institute of Electrical and Electronics Engineers (IEEE), v. 30, n. 6, p. 157–162, nov 2013. Citation on page 19.

SOUZA, G. Spadon de. **Characterization of mobility patterns and collective behavior through the analytical processing of real-world complex networks**. Master's Thesis (Master's Thesis) — Universidade de São Paulo, 2017. Citation on page 18.

SPADON, G.; BRANDOLI, B.; ELER, D. M.; RODRIGUES-JR, J. F. Detecting multi-scale distance-based inconsistencies in cities through complex-networks. **Journal of Computational Science**, 2018. ISSN 1877-7503. Citations on pages 19, 27, 28, and 79.

SPADON, G.; CARVALHO, A. C. P. L. F. de; RODRIGUES-JR, J. F.; ALVES, L. G. A. Reconstructing commuters network using machine learning and urban indicators. **Scientific Reports**, v. 9, n. 1, dec 2019. ISSN 2045-2322. Citations on pages 43, 79, and 139.

SPADON, G.; GIMENES, G.; RODRIGUES, J. F. Topological street-network characterization through feature-vector and cluster analysis. In: SPRINGER. **International Conference on Computational Science**. [S.l.], 2018. p. 274–287. Citations on pages 20, 27, and 103.

SPADON, G.; GIMENES, G.; RODRIGUES-JR, J. F. Identifying urban inconsistencies via street networks. **Procedia Computer Science**, v. 108, p. 18–27, 2017. ISSN 1877-0509. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland. Citation on page 28.

SPADON, G.; HONG, S.; BRANDOLI, B.; MATWIN, S.; RODRIGUES-JR, J. F.; SUN, J. Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Institute of Electrical and Electronics Engineers (IEEE), p. 1–17, 2021. ISSN 0162-8828. Citations on pages 18, 59, 79, and 139.

SPADON, G.; MACHADO, B. B.; ELER, D. M.; RODRIGUES, J. F. A distance-based tool-set to track inconsistent urban structures through complex-networks. In: SPRINGER. **International Conference on Computational Science**. [S.l.], 2018. p. 288–301. Citations on pages 27 and 103.

SPADON, G.; RODRIGUES-JR, J. F. Computer-assisted city touring for explorers. In: **Proceedings of the Workshop on Big Social Data and Urban Computing (BiDU) of the 44th International Conference on Very Large Data Bases (VLDB)**. [S.l.]: CEUR-WS.org, 2018. Citations on pages 19 and 27.

SPADON, G.; SCABORA, L. C.; ARAUJO, M. V. S.; OLIVEIRA, P. H.; SOUSA, E. P. M.; TRAINA-JR, C.; MACHADO, B. B.; RODRIGUES-JR, J. F. Behavioral Characterization of Criminality Spread in Cities. In: **International Conference on Computational Science**. [S.l.]: Elsevier BV, 2017. v. 108, p. 2537–2541. ISSN 1877-0509. International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland. Citation on page 18.

SPADON, G.; SCABORA, L. C.; ARAUJO, M. V. S.; OLIVEIR, P. H.; MACHADO, B. B.; SOUSA, E. P. M.; TRAINA, C.; RODRIGUES, J. F. Complex-Network Tools to Understand the Behavior of Criminality in Urban Areas. In: LATIFI, S. (Ed.). **Information Technology - New Generations**. [S.l.]: Springer International Publishing, 2018. p. 493–500. ISBN 978-3-319-54978-1. Citation on page 18.

STOUFFER, S. A. Intervening Opportunities: A Theory Relating Mobility and Distance. **American Sociological Review**, [American Sociological Association, SAGE Publications, Inc.], v. 5, n. 6, p. 845–867, dec 1940. ISSN 00031224. Citation on page 20.

STRANO, E.; NICOSIA, V.; LATORA, V.; PORTA, S.; BARTHÉLEMY, M. Elementary processes governing the evolution of road networks. **Scientific Reports**, Springer Nature, v. 2, n. 1, dec 2012. ISSN 2045-2322. Citation on page 19.

STRANO, E.; VIANA, M.; da Fontoura Costa, L.; CARDILLO, A.; PORTA, S.; LATORA, V. Urban Street Networks, a Comparative Analysis of Ten European Cities. **Environment and Planning B: Planning and Design**, SAGE Publications, v. 40, n. 6, p. 1071–1086, dec 2013. ISSN 0265-8135. Citation on page 20.

SUKHBAATAR, S.; SZLAM, A.; FERGUS, R. Learning multiagent communication with backpropagation. In: LEE, D. D.; SUGIYAMA, M.; LUXBURG, U. von; GUYON, I.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain**. [S.l.: s.n.], 2016. p. 2244–2252. Citation on page 20.

TEIXEIRA, A. S.; SANTOS, F. C.; FRANCISCO, A. P. Spanning Edge Betweenness in Practice. In: **Studies in Computational Intelligence**. [S.l.]: Springer Nature, 2016. v. 644, p. 3–10. ISBN 9783319305684. Citation on page 84.

TRAVENÇOLO, B.; da F. Costa, L. Accessibility in complex networks. **Physics Letters A**, Elsevier BV, v. 373, n. 1, p. 89–95, dec 2008. ISSN 03759601. Citation on page 19.

VELICKOVIC, P.; CUCURULL, G.; CASANOVA, A.; ROMERO, A.; LIÒ, P.; BENGIO, Y. Graph attention networks. **arXiv preprint: abs/1710.10903**, 2017. Citation on page 20.

VERMA, S.; ZHANG, Z. Graph capsule convolutional neural networks. **arXiv preprint: abs/1805.08090**, 2018. Citation on page 20.

VIANA, M. P.; da Fontoura Costa, L. Fast long-range connections in transportation networks. **Physics Letters A**, Elsevier BV, v. 375, n. 15, p. 1626–1629, apr 2011. ISSN 03759601. Citation on page 19.

Villas Boas, P. R.; RODRIGUES, F. A.; da Fontoura Costa, L. Modeling Highway Networks with Path-Geographical Transformations. In: FORTUNATO, S.; MANGIONI, G.; MENEZES, R.; NICOSIA, V. (Ed.). **Studies in Computational Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. v. 207, p. 115–126. ISBN 9783642012051. Citation on page 85.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of "small-world" networks. **Nature**, Springer Nature, v. 393, n. 6684, p. 440–442, jun 1998. ISSN 0028-0836. Citation on page 17.

XIAO, Y.; WEBSTER, C.; ORFORD, S. Identifying house price effects of changes in urban street configuration: An empirical study in Nanjing, China. **Urban Studies**, SAGE Publications, v. 53, n. 1, p. 112–131, jan 2014. ISSN 0042-0980. Citation on page 19.

XU, K.; HU, W.; LESKOVEC, J.; JEGELKA, S. How powerful are graph neural networks? In: **7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, Conference Track Proceedings**. [S.l.]: OpenReview.net, 2019. Citation on page 20.

XU, K.; LI, C.; TIAN, Y.; SONOBE, T.; KAWARABAYASHI, K.; JEGELKA, S. Representation learning on graphs with jumping knowledge networks. In: DY, J. G.; KRAUSE, A. (Ed.). **Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018**. [S.l.]: PMLR, 2018. (Proceedings of Machine Learning Research, v. 80), p. 5449–5458. Citation on page 20.

YING, Z.; YOU, J.; MORRIS, C.; REN, X.; HAMILTON, W. L.; LESKOVEC, J. Hierarchical graph representation learning with differentiable pooling. In: BENGIO, S.; WALLACH, H. M.; LAROCHELLE, H.; GRAUMAN, K.; CESA-BIANCHI, N.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, December 3-8, 2018, Montréal, Canada**. [S.l.: s.n.], 2018. p. 4805–4815. Citation on page 20.

ZAMITH, M.; LEAL-TOLEDO, R. C. P.; CLUA, E.; TOLEDO, E. M.; MAGALHÃES, G. V. de. A new stochastic cellular automata model for traffic flow simulation with drivers' behavior prediction. **Journal of Computational Science**, v. 9, p. 51–56, jul 2015. ISSN 18777503. Citation on page 19.

ZHANG, M.; CUI, Z.; NEUMANN, M.; CHEN, Y. An end-to-end deep learning architecture for graph classification. In: MCILRAITH, S. A.; WEINBERGER, K. Q. (Ed.). **Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018**. [S.l.]: AAAI Press, 2018. p. 4438–4445. Citation on page 20.

ZHANG, W.; CAO, K.; LIU, S.; HUANG, B. A multi-objective optimization approach for health-care facility location-allocation problems in highly developed cities such as Hong Kong. **Computers, Environment and Urban Systems**, Elsevier BV, v. 59, p. 220–230, sep 2016. ISSN 01989715. Citation on page 18.

ZHANG, Z.; CUI, P.; ZHU, W. Deep Learning on Graphs: A Survey. **IEEE Transactions on Knowledge and Data Engineering**, Institute of Electrical and Electronics Engineers (IEEE), 2020. ISSN 1041-4347. Citations on pages 18 and 85.

ZHONG, C.; ARISONA, S. M.; HUANG, X.; BATTY, M.; SCHMITT, G. Detecting the dynamics of urban structure through spatial network analysis. **International Journal of Geographical Information Science**, Informa UK Limited, v. 28, n. 11, p. 2178–2199, nov 2014. ISSN 1365-8816. Citation on page 19.

APPENDIX

A

ADDITIONAL PUBLICATIONS

In the following pages, we reproduce the subsequent thesis-related papers:

1. **Spadon G.**, Gimenes G., and Rodrigues-Jr J. F., *Topological Street-Network Characterization through Feature-Vector and Cluster Analysis*. International Conference on Computational Science. Springer, 2018.

⋆ Reprinted by permission from Springer Customer Service Centre GmbH (Order Number: 5044431447079): Springer Nature, Topological street-network characterization through feature-vector and cluster analysis [Spadon, Gimenes and Rodrigues 2018], Springer International Publishing AG.

2. **Spadon G.**, Brandoli B., Eler D. M., and Rodrigues-Jr J. F., *A Distance-Based Tool-Set to Track Inconsistent Urban Structures Through Complex-Networks*[1]. International Conference on Computational Science. Springer, 2018.

⋆ Reprinted by permission from Springer Customer Service Centre GmbH (Order Number: 5044431149899): Springer Nature, A distance-based tool-set to track inconsistent urban structures through complex-networks [Spadon *et al.* 2018], Springer International Publishing AG.

3. **Spadon G.** and Rodrigues-Jr J. F., *Computer-Assisted City Touring for Explorers*. Workshop on Big Social Data and Urban Computing (BiDU) co-located with the 44th International Conference on Very Large Data Bases (VLDB). CEUR-WS, 2018.

⋆ Paper under CC0 1.0 Universal (CC0 1.0)[2] Public Domain Dedication license, with copyright for the individual papers by the papers' authors, permitted for private and academic purposes without consent.

[1] Awarded paper – Invited for publication at the *Journal of Computational Science*.
[2] **Available at:** <https://creativecommons.org/publicdomain/zero/1.0/>

A Distance-Based Tool-Set to Track Inconsistent Urban Structures Through Complex-Networks

Gabriel Spadon[1](✉), Bruno B. Machado[2], Danilo M. Eler[3], and Jose F. Rodrigues Jr.[1]

[1] University of Sao Paulo, Sao Carlos, SP, Brazil
spadon@usp.br, junio@icmc.usp.br

[2] Federal University of Mato Grosso do Sul, Ponta Pora, MS, Brazil
bruno.brandoli@ufms.br

[3] Sao Paulo State University, Presidente Prudente, SP, Brazil
daniloeler@fct.unesp.br

Abstract. Complex networks can be used for modeling street meshes and urban agglomerates. With such a model, many aspects of a city can be investigated to promote a better quality of life to its citizens. Along these lines, this paper proposes a set of distance-based pattern-discovery algorithmic instruments to improve urban structures modeled as complex networks, detecting nodes that lack access *from*/*to* points of interest in a given city. Furthermore, we introduce a greedy algorithm that is able to recommend improvements to the structure of a city by suggesting where points of interest are to be placed. We contribute to a thorough process to deal with complex networks, including mathematical modeling and algorithmic innovation. The set of our contributions introduces a systematic manner to treat a recurrent problem of broad interest in cities.

Keywords: Complex network · Network analysis · Urban structure

1 Introduction and Related Works

The synergy of real-world systems can be described as complex networks that exchange information through their entities' relationships. Such networks can model complex systems from neuronal networks to subway systems [1] and also, they can shape cities when linking the network topology with georeferenced data.

By analyzing the complex network of a city, it is possible to extract features that can describe urban problems, which are meaningful indicators for city planners [2]. Such features can reveal, for instance, sites where social activities are more intense, regions where facilities should be placed, and neighborhoods that lack street access. Particularly, these networks can expose distance-based inconsistencies, which is how we refer to nodes that lack efficient street access *from*/*to* others in the same network, possibly resulting in structural bottlenecks.

Y. Shi et al. (Eds.): ICCS 2018, LNCS 10860, pp. 288–301, 2018.
https://doi.org/10.1007/978-3-319-93698-7_22

Along these lines, we identified a lack of methods to analyze and improve the structure, mobility, and street access of cities. Consequently, in this paper, we contribute with a mathematical tool-set and algorithms to track distance-based inconsistencies by analyzing the complex-network topology of a city. Our results have implications for the street access, supporting a finer street planning by enhancing mobility indicators and providing better city's structural assessment.

The core assumption of this study is that the network is supposed to provide streets that render the shortest distance between places. In this regard, our tool-set uses two distance-functions to track nodes that do not provide shortest distance routes between them and other nodes that are of some interest. Nodes that fail in providing minimum-length routes are considered to be inconsistent nodes, which are evidence of problems in the city structure. Accordingly, in the face of an inconsistency, we raise two hypotheses: **(A)** the network lacks a more appropriate mesh; or, instead, **(B)** the city lacks its points of interest placed in better locations. The first one indicates the need for new points of interest to distribute their load. Contrarily, the second one indicates the need for relocating points of interest because the topology of the terrain cannot afford new streets.

A vast number of studies have been conducted to analyze inherent properties and behaviors found in cities. For instance, multiple metrics have been adopted to explain their structural conditions [3], their intense traffic of vehicles [4], and the emergence of collective behavior [5]. In other studies, the authors centered on the geometrical perspective of the network [6], and on the elements positioning [7,8]. Furthermore, there are those who reviewed the role of the city elements [9,10], that addressed the support to the urban planning and design [11,12], and that improved the facility-location analysis and planning of street meshes [13]. In addition to the ones that inspected the effectiveness of the underground systems [10] and the improvement of long-range connections [14], besides those who defined the concept of accessibility through complex networks and cities [15] and that tested the centrality of cities considering their space syntax [16–18].

In this paper, we contribute with a tool-set that improve the analysis of cities by tracking inconsistent urban structures through complex networks. This proposal follows by showing that the distance between two nodes can reveal ill-located points of interest and that such information can be used to make a city better distance-efficiently to their citizens. To present our contributions, this paper is organized as follows: Sect. 2 discusses our mathematical formulation and related algorithms; Sect. 3 discusses the results about the applicability of the proposed tool-set; and, Sect. 4 presents the conclusions and final remarks.

2 Mathematical Formulation and Algorithms

2.1 Preliminaries

Along the text, we refer to a complex network as a distance-weighted directed graph $\mathtt{G} = \{\mathtt{V}, \mathtt{E}\}$, which is composed by a set $\mathtt{V}$ of $|\mathtt{V}|$ nodes and a set $\mathtt{E}$ of $|\mathtt{E}|$

edges. To model a city as a complex network, we considered streets as the edges and their crossings as the nodes, aiming to preserve their element's geometry. An edge $\mathtt{e} \in \mathtt{E}$ is an ordered pair $\langle \mathtt{i}, \mathtt{j} \rangle$, in which $\mathtt{i} \in \mathtt{V}$ is named *source* and $\mathtt{j} \in \mathtt{V}$ is named *target*, $\mathtt{i} \neq \mathtt{j}$. Each node $\mathtt{i} \in \mathtt{V}$ has two properties $\{\mathcal{L}^{\mathtt{i}}_{\mathtt{at}}, \mathcal{L}^{\mathtt{i}}_{\mathtt{on}}\}$ that correspond to their coordinates—$\mathcal{L}^{\mathtt{i}}_{\mathtt{at}}$ is the latitude and $\mathcal{L}^{\mathtt{i}}_{\mathtt{on}}$ the longitude. Based on such coordinates, we conferred to the edges in $\mathtt{E}$ a floating-point weight that refers to the great-circle (or *inline*) distance between their *source* and *target*.

Our mathematical tool-set tracks inconsistencies identified through distance functions to detect which element does not follow a pattern. The pattern that we consider refers to the real-world distance between the nodes of the network, which in turn can provide insights about the locomotion through the city streets. In this regard, we begin by tracking a set $\mathtt{P} \subset \mathtt{V}$ of points of interest; the idea, then, is to determine two sets of nodes that surrounds a point of interest $\mathtt{p} \in \mathtt{P}$, which can reveal the city's inconsistencies through applying algebraic operations. We introduce these two sets as the *perimeter set of p* and the *network set of p*.

2.2 Grouping Nodes in the Surroundings of Points of Interest

The first set matches the closest nodes to a point of interest $\mathtt{p}$ according to the great-circle (or *inline*) distance, which is referred as the *perimeter set of p*:

$$\mathtt{V}^{\mathtt{E}}_{\mathtt{p}} = \{\mathtt{v} \in \mathtt{V} | \mathtt{d}^{\mathtt{E}}_{\mathtt{vp}} < \mathtt{d}^{\mathtt{E}}_{\mathtt{v\bar{p}}}, \forall \mathtt{p} \in \mathtt{P}, \forall \bar{\mathtt{p}} \in \mathtt{P}, \mathtt{p} \neq \bar{\mathtt{p}}\} \tag{1}$$

where $\mathtt{d}^{\mathtt{E}}_{\mathtt{ij}}$ is the great-circle distance between $\mathtt{i}$ and $\mathtt{j}$ in the surface of Earth:

$$\mathtt{d}^{\mathtt{E}}_{\mathtt{ij}} = \mathcal{R} \times \mathtt{arcos}\left(\mathtt{sin}(\mathcal{L}^{\mathtt{i}}_{\mathtt{at}})\mathtt{sin}(\mathcal{L}^{\mathtt{j}}_{\mathtt{at}}) + \mathtt{cos}(\mathcal{L}^{\mathtt{i}}_{\mathtt{at}})\mathtt{cos}(\mathcal{L}^{\mathtt{j}}_{\mathtt{at}})\mathtt{cos}(\triangle_{\mathcal{L}^{\mathtt{ij}}_{\mathtt{on}}})\right) \tag{2}$$

where $\mathcal{L}^{\mathtt{i}}_{\mathtt{at}}$ and $\mathcal{L}^{\mathtt{j}}_{\mathtt{at}}$ are the latitudes, $\triangle_{\mathcal{L}^{\mathtt{ij}}_{\mathtt{on}}}$ is the difference between the longitudes $\mathcal{L}^{\mathtt{i}}_{\mathtt{on}}$ and $\mathcal{L}^{\mathtt{j}}_{\mathtt{on}}$, both of nodes $\mathtt{i}$ and $\mathtt{j}$. Also, $\mathcal{R}$ is the radius of Earth (6,378 km), and all values are represented in radians. Given a graph $\mathtt{G} = \{\mathtt{V}, \mathtt{E}\}$ and a set of points of interest $\mathtt{P}$, a node $\mathtt{v} \in \mathtt{V}$ pertains to the perimeter set of only one $\mathtt{p} \in \mathtt{P}$.

$$\therefore \mathtt{V}^{\mathtt{E}}_{\mathtt{p}} \text{ and } \mathtt{V}^{\mathtt{E}}_{\bar{\mathtt{p}}} \text{ — } \forall \mathtt{p} \in \mathtt{P}, \forall \bar{\mathtt{p}} \in \mathtt{P}, \mathtt{p} \neq \bar{\mathtt{p}} \text{ — are mutually disjoint.}$$

The second set corresponds to the *network set of p*, which is made of nodes closest to a point of interest $\mathtt{p}$ according to the length of their shortest paths:

$$\mathtt{V}^{\mathtt{N}}_{\mathtt{p}} = \{\mathtt{v} \in \mathtt{V} | \mathtt{d}^{\mathtt{N}}_{\mathtt{vp}} < \mathtt{d}^{\mathtt{N}}_{\mathtt{v\bar{p}}}, \forall \mathtt{p} \in \mathtt{P}, \forall \bar{\mathtt{p}} \in \mathtt{P}, \mathtt{p} \neq \bar{\mathtt{p}}\} \tag{3}$$

where $\mathtt{d}^{\mathtt{N}}_{\mathtt{ij}}$ is the length of the shortest directed path ($\mathtt{spath}_{\mathtt{ij}}$) between $\mathtt{i}$ and $\mathtt{j}$—*i.e.* the sum of weights of all the edges in a minimum-length path, as follows:

$$\mathtt{minimum\_length}(\mathtt{spath}_{\mathtt{ij}}) = \sum_{\mathtt{e} \in \mathtt{spath}_{\mathtt{ij}}} \mathtt{weight}(\mathtt{e}) \tag{4}$$

Recall that, the edge weight is given by the straight-line distance between their nodes using the great-circle distance (see Eq. 2). We refer to the shortest path length as *network distance*, in the sense that one must necessarily (in the best case) move across this path to go from the source node to the target node. Notice that any node $\mathtt{v} \in \mathtt{V}$ is network-closest to one and only one $\mathtt{p} \in \mathtt{P}$.

$$\therefore\ \mathtt{V}^{\mathtt{N}}_{\mathtt{p}} \text{ and } \mathtt{V}^{\mathtt{N}}_{\bar{\mathtt{p}}} \text{ — } \forall \mathtt{p} \in \mathtt{P}, \forall \bar{\mathtt{p}} \in \mathtt{P}, \mathtt{p} \neq \bar{\mathtt{p}} \text{ — are mutually disjoint.}$$

In cases where the complex network is directed, the *network-distance TO a point of interest* is not necessarily the same as the *network-distance FROM a point of interest*, which may result in different network sets for the same $\mathtt{p}$. This detail is addressed in the following section, where we define the network set from a point of interest to the nodes in $\mathtt{V}$ by mean of the *reversed network-set of p*:

$$\bar{\mathtt{V}}^{\mathtt{N}}_{\mathtt{p}} = \{\mathtt{v} \in \mathtt{V} | \mathtt{d}^{\mathtt{N}}_{\mathtt{pv}} < \mathtt{d}^{\mathtt{N}}_{\bar{\mathtt{p}}\mathtt{v}}, \forall \mathtt{p} \in \mathtt{P}, \forall \bar{\mathtt{p}} \in \mathtt{P}, \mathtt{p} \neq \bar{\mathtt{p}}\} \tag{5}$$

2.3 Compartmentalizing Inconsistencies for Directed Networks

Consider different public services of a city as points of interest; such services may have different ways to assist the population, but all of them must require locomotion as a condition for assistance. For example, in the case of doctors' clinics, it is desired that patients get there efficiently. In turn, police stations require that their police officers efficiently reach the house of the citizens. In the case of schools, the daily routine demands an efficient back-and-forth transit to students. Along with other services that can be fitted with this assumption. Notice that, we are referring to efficient paths as the ones with minimum length.

In the first example, there is an implicit displacement from a node $\mathtt{v}$ to a node $\mathtt{p}$; in the second one, the displacement is from the node $\mathtt{p}$ to the node $\mathtt{v}$; and, in the third case, there is a bi-directional displacement between $\mathtt{v}$ and $\mathtt{p}$, in which $\mathtt{v}$ is an ordinary node and $\mathtt{p}$ is a specific point-of-interest. Based on the network direction, those three cases led to the following definitions:

1. **Inward Inconsistency:** nodes that are inline-closest to a point of interest, but network-closest (from $\mathtt{v}$ to $\mathtt{p}$, as given by Eq. 3) to a different one:

$$\Psi^{\mathtt{I}}_{\mathtt{p}} = \mathtt{V}^{\mathtt{E}}_{\mathtt{p}} - \mathtt{V}^{\mathtt{N}}_{\mathtt{p}} \tag{6}$$

2. **Outward Inconsistency:** the same as the previous category, but in the opposite direction (from $\mathtt{p}$ to $\mathtt{v}$, as given by Eq. 5), resulting in the set:

$$\Psi^{\mathtt{O}}_{\mathtt{p}} = \mathtt{V}^{\mathtt{E}}_{\mathtt{p}} - \bar{\mathtt{V}}^{\mathtt{N}}_{\mathtt{p}} \tag{7}$$

3. **Absolute Inconsistency:** nodes that are, simultaneously, considered to be inward and outward inconsistencies—*i.e.* nodes in the sets' intersection:

$$\Psi^{\mathtt{A}}_{\mathtt{p}} = \Psi^{\mathtt{I}}_{\mathtt{p}} \cap \Psi^{\mathtt{O}}_{\mathtt{p}} \tag{8}$$

As mentioned, these categories rely on the direction of the network. In cases where there is no direction, there will be no minimum-length divergence between paths of a round trip, but yet the inconsistencies can be tracked by calculating the difference between the perimeter set $\mathtt{V}^{\mathtt{E}}_{\mathtt{p}}$ and the network set $\mathtt{V}^{\mathtt{N}}_{\mathtt{p}}$ of $\mathtt{p}$. To provide further discussion, hereinafter we are considering just directed networks.

2.4 Tracking Distance-Based Inconsistencies

In this section, we discuss Algorithm 1 that joins the concepts that we previously introduced. The aim of such algorithm is to track distance-based inconsistencies in distance-weighted directed networks by using a set $\mathtt{P}$ of $|\mathtt{P}|$ of points of interest. Notice that, despite the definition of inconsistency is segmented into three types (see Sect. 2.3), the algorithm considers a single inconsistency type at a time.

The algorithm starts by filling a set of empty sets, each one reserved to store the inconsistencies of a single point of interest (see lines 1 to 2). Subsequently, we use $\mathtt{p}^{\mathtt{E}}$ and $\mathtt{p}^{\mathtt{N}}$ to store, respectively, the inline-closest and network-closest points of interest to a node $\mathtt{v} \in \mathtt{V}$ (see lines 5 and 6). We used the external functions `inline_closest` and `network_closest` (see lines 8 and 9) to extract the closest point of interest to the node $\mathtt{v}$; they implement, respectively, Eq. 2 and 4. Following, we perform a test to check whether a node is an inconsistency or not; thus, if the inline-closest point $\mathtt{p}^{\mathtt{E}}$ and the network-closest point $\mathtt{p}^{\mathtt{N}}$ are not the same (see line 11) then $\mathtt{v}$ is an inconsistency of $\mathtt{p}^{\mathtt{E}}$ (see line 12). Finally, a set of the inconsistencies of $|\mathtt{P}|$ points of interest is returned as the result (see line 13).

```
    Data: $G = \{V, E\}$, $P \subset V$, and $c \in \{I, O, A\}$ — c is used to indicate the direction
    Result: $\{\Psi^c_p, \forall p \in P\}$ — a set of inconsistencies for all points of interest $p \in P$
1   $\Psi^c \leftarrow \emptyset$
2   for each $p \in P$ do
3       $\Psi^c_p \leftarrow \emptyset$ // notice that $\Psi^c_p \subset \Psi^c, \forall p \in P$, therefore $|\Psi^c| = |P|$
4   for each $v \in V$ do
5       $p^E \leftarrow \emptyset$
6       $p^N \leftarrow \emptyset$
7       for each $\bar{p} \in P$ do
8           $p^E \leftarrow$ inline_closest($v, \langle p^E, \bar{p} \rangle$)
9           $p^N \leftarrow$ network_closest($v, \langle p^N, \bar{p} \rangle, c$)
10      if $p^E \neq \emptyset$ and $p^N \neq \emptyset$ then
11          if $\{p^E\} - \{p^N\} \neq \emptyset$ then
12              $\Psi^c_{p^E} \leftarrow \Psi^c_{p^E} \cup \{v\}$ // v should be closer to $p^E$ than to $p^N$
13  return $\Psi^c$
```

Algorithm 1: An algorithm to track distance-based inconsistencies in cities modeled as networks. We use $\mathtt{p}^{\mathtt{E}}$ and $\mathtt{p}^{\mathtt{N}}$ to refer to the closest points of interest to a node $\mathtt{v}$ considering, respectively, the inline and the network distances; other methods are related to the ones of Sect. 2.2.

Given a graph $\mathtt{G} = \{\mathtt{V}, \mathtt{E}\}$, a set $\mathtt{P}$ of $|\mathtt{P}|$ points of interest, and an inconsistent node $\mathtt{i}$; such node is known to be an inconsistency to one and only one $\mathtt{p} \in \mathtt{P}$.

$\therefore$ $\Psi^{\mathtt{c}}_{\mathtt{p}}$ and $\Psi^{\mathtt{c}}_{\bar{\mathtt{p}}}$ — $\forall \mathtt{p} \in \mathtt{P}, \forall \bar{\mathtt{p}} \in \mathtt{P}, \mathtt{p} \neq \bar{\mathtt{p}}$, $\mathtt{c} \in \{\mathtt{I}, \mathtt{O}, \mathtt{A}\}$ — are mutually disjoint.

Consequently, it is possible to derive two other sets from a point of interest $\mathtt{p}$: **(i)** the inconsistency set $\Psi^{\mathtt{c}}_{\mathtt{p}}$; and **(ii)** the set of consistent nodes $\bar{\Psi}^{\mathtt{c}}_{\mathtt{p}} = \mathtt{V}^{\mathtt{E}}_{\mathtt{p}} - \Psi^{\mathtt{c}}_{\mathtt{p}}$, such that $\bar{\Psi}^{\mathtt{c}}_{\mathtt{p}} \cap \Psi^{\mathtt{c}}_{\mathtt{p}} = \emptyset$. The consistent nodes are fundamental to the process of suggesting locations to points of interest because they provide a smaller average distance to the nodes in their perimeter, different than an inconsistent node.

2.5 Reducing Distance-Based Inconsistencies

In this section, we introduce Algorithm 2, which was designed to suggest changes in the location of points of interest to improve their access through the streets of a city. The task of finding a perfect location for a point of interest might demand the test of all possibilities through an exhaustive search. Consequently, our algorithm has a greedy approach that uses information about centrality metrics to guide the placement of a point of interest. Centrality is not only an adequate technique to quantify the importance of a node but also it is capable to indicate central locations that are equally accessible to all nodes of a network.

Along these lines, we decided to adopt ***Straightness Centrality*** [9] as the centrality metric of Algorithm 2 because it analyzes the nodes of a network by joining both inline and network distances. It is noteworthy that any distance-based centrality metric could be employed, as well as multiple metrics together; however, different metrics tend to provide dubious or bad choices for a relocation.

Our algorithm starts by initializing auxiliary variables (see line 1) and by tracking the inconsistent nodes in the original version of the network (see line 2). In line 4, it starts looping until all points of interest have been replaced or until there are no more inconsistencies to be reduced from the original network. After that, it tries to change one point of interest at a time (see line 7). The candidates to host a point of interest pertains to the induced subgraph $\mathtt{G}^{\mathtt{E}}_{\mathtt{p}}$ of consistent nodes (see line 8). By using the induced subgraph the algorithm searches for the node that has the highest centrality value among all the other ones (see line 9).

The algorithm continues by testing the highest central node as the new location to the point of interest; such that, it temporally replaces the node (see line 10) and then it collects information about the inconsistencies of this network configuration (see line 11). Following, it tests whether the new configuration causes fewer inconsistencies then the previous one (see line 12) before marking the node for relocation (see lines 13 to 15). In a greedy fashion, it first selects the point of interest that by being replaced will lead to the highest elimination of inconsistencies. After choosing the one to be replaced, we perform integrity tests, we mark the node as relocated, and then we remove it (see lines 16 to 19).

The algorithm ends when there are no more profitable changes (see line 21). It is noteworthy to mention that each point of interest can be moved only once; this is due to the greedy nature of the algorithm. Otherwise, it would run until there are no more inconsistencies in the network at a prohibitive computational cost. The output of the algorithm is a set $\mathtt{R}$ of new locations (see line 22); each element $\mathtt{r} \in \mathtt{R}$ is an ordered pair $\mathtt{r} = \langle \mathtt{old}_{\mathtt{p}}, \mathtt{new}_{\mathtt{p}} \rangle$ that denotes the current ($\mathtt{old}_{\mathtt{p}}$) node where a point of interest is and a better node ($\mathtt{new}_{\mathtt{p}}$) for placing it.

Algorithm 2 runs in $\mathtt{O}(|\mathtt{V}||\mathtt{P}|^3)$ in the average case, where $|\mathtt{P}|$ is the number of points of interest and $|\mathtt{V}|$ is the number of nodes, $|\mathtt{P}| \ll |\mathtt{V}|$. Besides that, the algorithm was designed to be straightly parallelized; and, moreover, in our tests, it took less than a minute to compute a whole city with 200,000 inhabitants.

Correctness of the algorithm formulation

In this section, we demonstrate that Algorithm 2 is finite and it never increases the number of inconsistencies of a city, as required by the problem formulation.

Theorem 1. *We hypothesize that Algorithm 2 provides a set of central and consistent nodes that can replace specific points of interest in a city because replacing them will* ***never increase the total number of inconsistencies****.*

Proof. Hereinafter, aiming to prove Theorem 1 by reduction to absurdity, we are supposing that the use of Algorithm 2 can increase the number of inconsistencies.

```
Data: $G = \{V, E\}$, $P \subset V$, and $c \in \{I, O, A\}$ — c is used to indicate the direction
Result: R — set of suggested positions for points of interest
1  $R \leftarrow \emptyset$ $\bar{P} \leftarrow \emptyset$ $\tilde{P} \leftarrow \emptyset$
2  $\Psi^c \leftarrow$ algorithm_1$(G, P, c)$
3  $\Phi^c \leftarrow \Psi^c$ // copy of the original set
4  while $|P| - |\bar{P}| > 0$ and $\left(\sum_{i=1}^{|P|} |\Psi_i^c| \geq \sum_{i=1}^{|P|} |\Phi_i^c|\right)$ do
5      $old_p \leftarrow \emptyset$
6      $new_p \leftarrow \emptyset$
7      for each $p \in (P - \bar{P})$ do
8          $G_p^E \leftarrow G\left(V_p^E - \Psi_p^c\right)$ // induced subgraph of consistent nodes
9          $\bar{p} \leftarrow$ extract_central$(G_p^E)$
10         $\mathbb{P} \leftarrow \left(\left((P - \bar{P}) \cup \tilde{P}\right) - \{p\}\right) \cup \{\bar{p}\}$
11         $\Omega^c \leftarrow$ algorithm_1 $(G, \mathbb{P}, c)$
12         if $\left(\sum_{i=1}^{|P|} |\Phi_i^c| > \sum_{i=1}^{|P|} |\Omega_i^c|\right)$ then
13             $\Phi^c \leftarrow \Omega^c$ // new lowest number of inconsistencies
14             $old_p \leftarrow p$
15             $new_p \leftarrow \bar{p}$
16     if $old_p \neq \emptyset$ and $new_p \neq \emptyset$ then
17         $R \leftarrow R \cup \{old_p, new_p\}$ // $old_p$ was moved to $new_p$
18         $\bar{P} \leftarrow \bar{P} \cup \{old_p\}$ // old location
19         $\tilde{P} \leftarrow \tilde{P} \cup \{new_p\}$ // new location
20     else
21         break // there are no more enhancements to be made
22 return R
```

Algorithm 2: An algorithm that uses the contributions of Algorithm 1 to reduce distance-based inconsistencies of cities shaped as networks.

Bearing in mind that the type of the inconsistency has no effect on such proof, we will follow by proving the algorithm using Inward Inconsistency (see Sect. 2.3).

Consider the existence of a city mapped as a complex network $\mathtt{G} = \{\mathtt{V}, \mathtt{E}\}$ and a set $\mathtt{P}$ of $|\mathtt{P}|$ points of interest located in this same city. We start by finding the *perimeter set of p* ($\mathtt{V}_\mathtt{p}^\mathtt{E}$) for each $\mathtt{p} \in \mathtt{P}$, which is given by Eq. 1. Subsequently, we proceed with gathering the *network set of p* ($\mathtt{V}_\mathtt{p}^\mathtt{N}$) that is defined by Eq. 3.

Following, we detect a consistent node $\bar{\mathtt{p}}$ that is the most central by an arbitrary centrality metric. The most central node is the one that has the highest centrality when compared to the other nodes, potentially being a better place for positioning a point of interest in a city. We follow by replacing $\mathtt{p}$ by the most central node $\bar{\mathtt{p}}$ in its perimeter. Then, we calculate the updated perimeter ($\mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{E}$) and network ($\mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{N}$) sets, both of $\bar{\mathtt{p}}$. Notice that $\mathtt{p} \neq \bar{\mathtt{p}}$, thus $\mathtt{V}_\mathtt{p}^\mathtt{E} \neq \mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{E}$ and $\mathtt{V}_\mathtt{p}^\mathtt{N} \neq \mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{N}$.

At this point, there are two pairs of answers, one pair for $\mathtt{p}$ and the other one for $\bar{\mathtt{p}}$, as follows: $\langle \mathtt{V}_\mathtt{p}^\mathtt{E}, \mathtt{V}_\mathtt{p}^\mathtt{N} \rangle$ and $\langle \mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{E}, \mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{N} \rangle$. The algorithm we proposed will replace $\mathtt{p}$ by $\bar{\mathtt{p}}$ following Eq. 9, which corresponds to a clause saying that the sets computed from $\bar{\mathtt{p}}$ will be used just if they provide fewer inconsistencies than the original set; otherwise, it will keep the original one without making any changes.

$$\Psi_\mathtt{p}^\mathtt{I} = \begin{cases} \mathtt{V}_\mathtt{p}^\mathtt{E} - \mathtt{V}_\mathtt{p}^\mathtt{N}, & \left|\mathtt{V}_\mathtt{p}^\mathtt{E} - \mathtt{V}_\mathtt{p}^\mathtt{N}\right| \leq \left|\mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{E} - \mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{N}\right| \\ \mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{E} - \mathtt{V}_{\bar{\mathtt{p}}}^\mathtt{N}, & \mathtt{otherwise} \end{cases} \tag{9}$$

The algorithm ceases when all the points of interest are changed at least once or when no change will result in inconsistency elimination (see Sect. 2.5); as such, the algorithm is guaranteed to be finite. Therefore, by reduction to absurdity, it is an *absurdum* to suppose that the number of inconsistencies increases due to the use of Algorithm 2 because the algorithm provides a set with less or equal inconsistencies than the original set—as defined by Eq. 10.

$$\therefore \left|\Psi_\mathtt{p}^\mathtt{I}\right| \leq \left|\mathtt{V}_\mathtt{p}^\mathtt{E} - \mathtt{V}_\mathtt{p}^\mathtt{N}\right| \tag{10}$$

3 Results and Discussions

The tool-set we proposed was validated over the Brazilian city of Sao Carlos. Such city was instantiated as a complex network through a digital map from *OpenStreetMap*[1]. We considered streets as edges and their crossings as nodes; this way, we preserved the georeferenced attributes of the city that are necessary to the distance computation of our tool-set. The resulting network is planar and it can be represented in two dimensions, in which edges intersect only at nodes.

3.1 Assessing Inconsistency Recovery

In this section, we analyze the inconsistent nodes found in the city of Sao Carlos regarding the location of hospitals, police stations, and public schools, which are our points of interest; such public services are known to be affected respectively

[1] www.openstreetmap.org.

by inward, outward, and absolute inconsistencies as described in Sect. 2.3. It is noteworthy that each set of points of interest are independent, as such, the inconsistencies of one set of points have no relationship with the ones of others.

The inconsistent nodes we tracked are in Table 1, which suggest that their occurrence is connected to the number of points of interest. In fact, they appear whenever different perimeters meet; as a consequence, there is no way to eradicate them without altering the network topology by changing the streets' direction or creating new streets. In addition, more points of interest mean more boundaries, what tends to increase their number. Hence, the challenge is to find locations to points of interest that reduce, rather than eradicate, inconsistencies.

We used Algorithm 2 so to reduce the inconsistencies from Sao Carlos (see Table 1). The algorithm suggested relocating 6 hospitals, 2 police stations, and 9 public schools; such configuration, was able to reduce 160 inconsistencies from the hospitals (from 559 to 399), 123 inconsistencies from the police stations (from 342 to 219), and 179 inconsistencies from the public schools (from 663 to 484). Notice that the inconsistencies of some points of interest raised in number from the original to the enhanced city, which is a setback of our approach. However, as we have already proved, the total number of inconsistencies is always smaller.

Table 1. Analysis of the inconsistencies of the city of Sao Carlos, in which we considered police stations, hospitals, and public schools as points of interest; we use # to refer to the total number of inconsistencies and % to their percentage.

n^{th} POI	Original City						Enhanced City						n^{th} POI
	Hospitals		*Police Stations*		*Schools*		*Hospitals*		*Police Stations*		*Schools*		
	#	%	#	%	#	%	#	%	#	%	#	%	
01	013	02.3%	032	09.3%	015	02.2%	14	03.5%	30	13.7%	19	03.9%	01
02	002	00.3%	004	01.1%	077	11.6%	02	00.5%	48	21.9%	13	02.6%	02
03	012	02.1%	086	25.1%	043	06.4%	18	04.5%	96	43.8%	37	07.6%	03
04	019	03.4%	029	08.4%	071	10.7%	04	01.0%	32	14.6%	58	11.9%	04
05	030	05.3%	191	55.8%	114	17.1%	87	21.8%	13	05.9%	57	11.7%	05
06	049	08.7%	—	—	003	00.4%	51	12.7%	—	—	01	00.2%	06
07	145	25.9%	—	—	008	01.2%	26	06.5%	—	—	01	00.2%	07
08	039	06.9%	—	—	015	02.2%	22	05.5%	—	—	18	03.7%	08
09	012	02.1%	—	—	078	11.7%	31	07.7%	—	—	77	15.9%	09
10	043	07.6%	—	—	051	07.6%	63	15.7%	—	—	48	09.9%	10
11	072	12.8%	—	—	038	05.7%	45	11.2%	—	—	41	08.4%	11
12	095	16.9%	—	—	015	02.2%	17	04.2%	—	—	11	02.2%	12
13	028	05.0%	—	—	056	08.4%	19	04.7%	—	—	10	02.0%	13
14	—	—	—	—	008	01.2%	—	—	—	—	16	03.3%	14
15	—	—	—	—	060	09.0%	—	—	—	—	51	10.5%	15
16	—	—	—	—	011	01.6%	—	—	—	—	26	05.3%	16
Total	559	100%	342	100%	663	100%	399	100%	219	100%	484	100%	Total

3.2 Supporting the Designing of Urban Structures

Our tool-set is not only to be used in the automatic recovery of inconsistencies, but also to assist human-made urban-planning decisions. This is the case, for instance, when a specialist designs a city by having knowledge of the citizens' needs. In this case, Algorithms 1 and 2 can aid the process by analyzing and recommending distance-efficiently locations that are feasible to points of interest.

This section introduces two hypothetical case studies that depict our tool-set in practice. Both of them were conducted considering a subset of hospitals and public schools of the city of Sao Carlos (see Sect. 3.1). Nonetheless, our tool-set is extendable to any point of interest since it is equivalent to all of them.

Both case studies follow as in Fig. 1, in which we start by finding a point of interest, next we try to solve the problem by ourselves, and then we use the algorithms to improve our results; all steps are guided under the light of the nodes' ***straightness centrality***. Furthermore, all case studies are represented by the induced subgraph of the point of interest being analyzed and, although we have illustrated the inconsistencies in Fig. 1, in the case studies they are not visible because they do not provide visual information to the other images.

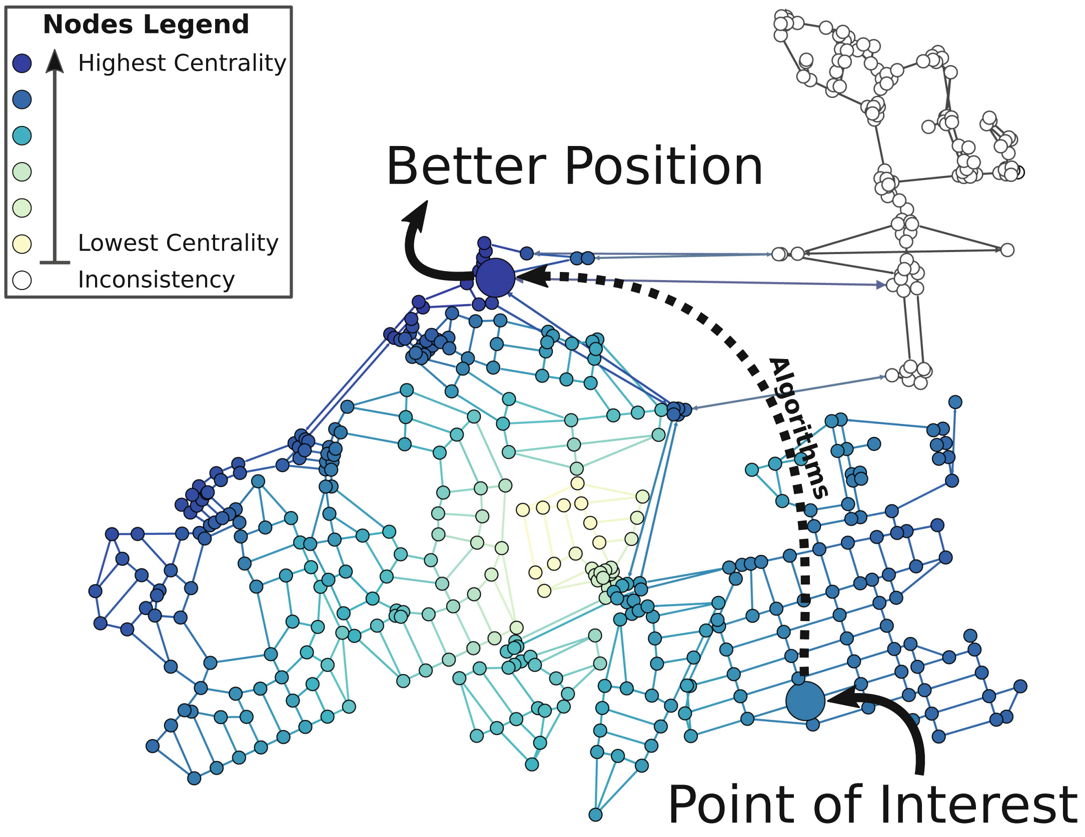


Fig. 1. Illustration of the process of designing urban structures under the light of centrality metrics. This process starts by identifying nodes that are of interest, then it follows by tracking their inconsistencies, and it ends by suggesting new locations—that reduce the number of inconsistencies—to place these nodes.

Case Study 1: Creating a new hospital to reduce demand

From the set of hospitals of the city of Sao Carlos, we identified one that, when compared to another hospital in the city, has excessive nodes in its perimeter (see Fig. 2a). There is no specific explanation of the hospital's location and, for instance, we can think that the city may have grown after the hospital has been built or the planners did not take the surroundings of the hospital into account. One thing is for sure, an extensive area with an ill-positioned point of interest will deprive the street access of the nodes; in this case, when points of interest are healthcare facilities, time-critical activities, as the transportation of patients in a critical state, can be jeopardized by lack of street access. Hence, the problem becomes where to build a hospital and how to avoid inconsistencies.

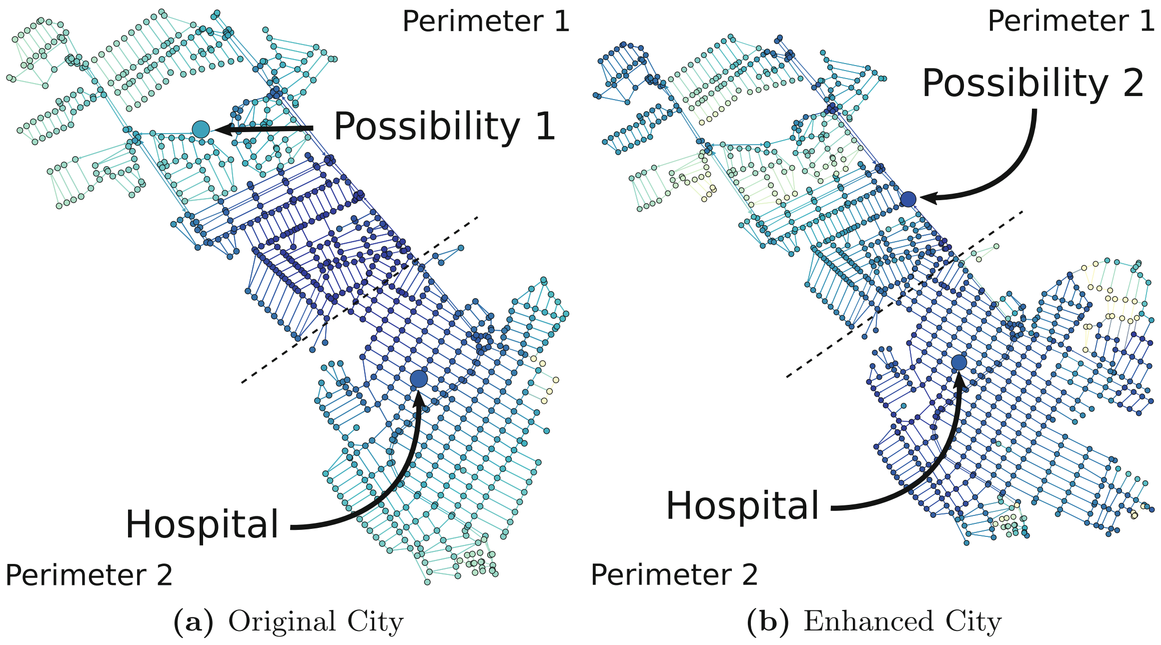


Fig. 2. Illustration of the assisted urban planning task from the first case study, in which the point of interest is a hospital and the color of the nodes denotes their centrality—the darker, the higher. Figure 2a shows a hospital's perimeter that is too large causing lack of access. We placed a new hospital in an eye-based central location in that same area to solve this issue. Afterwards, we used the algorithm to reduce inconsistencies, which suggested relocating the new hospital to a more central location that reduces the hospital's inconsistencies; as in Fig. 2b.

First, we tried to solve the problem manually by an eye-based analysis of a location that could provide equal nodes to the perimeters of both hospitals. Figure 2a shows a possible place to the new hospital as well as the resulting perimeter of both of them, which are defined by a line that cuts the image in half. After that, we inserted the proposed location in the set of hospitals and we used Algorithm 1 to track the inconsistencies of the resulting configuration. Such configuration lead us to 615 inconsistencies, which is a bigger value than the original city. Thus, we succeeded in building a hospital that splits the perimeter into two, but we failed in providing efficient access to both old and new hospital.

In a second approach, we analyzed the nodes' centrality together with a supporting visualization. We colored the nodes by their centrality, what allowed us to notice that the selected location for the new hospital is a node with low centrality. Then, we used Algorithm 2 to suggest a better place for the new hospital while keeping the location of the old one. Doing so, the city inconsistencies were reduced from 615 to 352 (see Fig. 2b), which positively reflected in the mobility of this area by distributing the demand between both hospitals. Thus, creating a new hospital in a specific location was able to reduce almost half of the inconsistencies of the city without relocating the existing ones.

Case Study 2: Merging schools to centralize public resources

In a similar fashion, we identified two public schools that are adjacent and support a short set of nodes. In this case, the proximity of the schools (see Fig. 3a) is a problem since none of them is used up to its capacity implying a waste of public resources. In a first approach, by using Algorithm 2 to relocate them, the number of inconsistencies was reduced from 663 to 635.

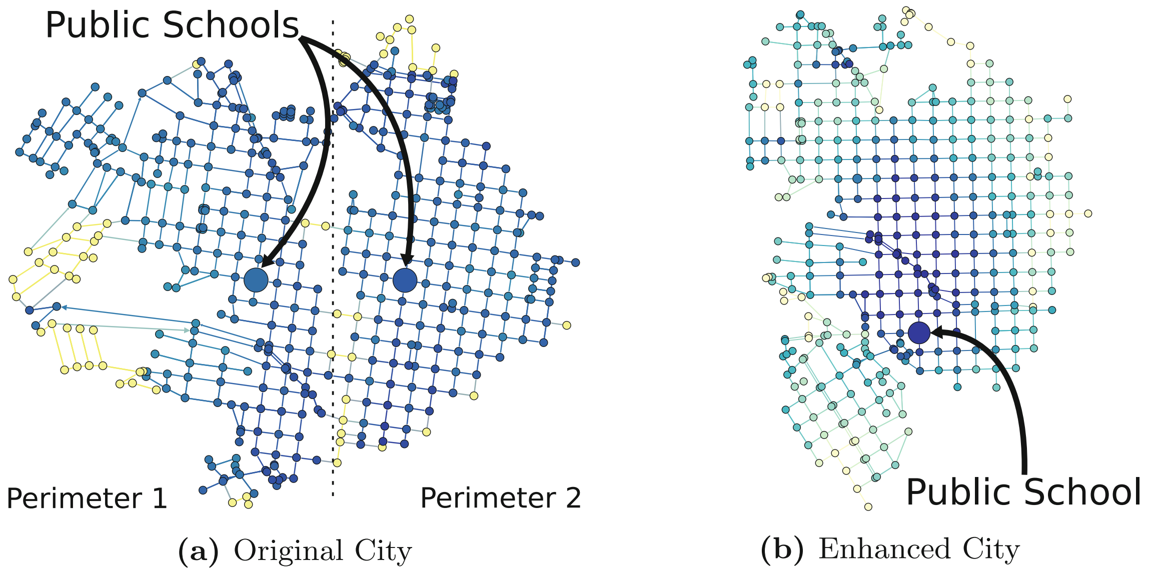


Fig. 3. Illustration of the assisted urban planning task from the second case study, in which the points of interest are public schools and the color of the nodes denotes their centrality—the darker, the higher. In this case study, we treated a problem related to the waste of resources that was caused by having two schools near each other; Fig. 3a shows the problematic area, which is small, increasing the drawbacks related to access. By replacing both schools with a single one we achieved a better coverage of nodes, as depicted in Fig. 3b.

Considering the size of the perimeter of both schools, we decided to remove one school to improve the utility of the one that remained. By centralizing the schools in a single node, we can reduce inconsistencies because there will be fewer perimeters bordering each other; hence, the inconsistencies, located whenever two of them meet, will be naturally decreased. To further enhance this process, we used the color-coded centrality metric to choose a candidate to be the new sole school. Afterward, we used Algorithm 2 to provide a better location (see Fig. 3b), which reduced the total number of inconsistencies from 635 to 445.

3.3 Discussions on Results Generalization

For a concise results presentation, we have assumed: **(i)** that any displacement is through cities' streets; and, **(ii)** a city with a uniform population distribution. However, our tool-set holds for scenarios where these assumptions are not true.

We can use weights in accordance with the type of the displacement rather than using streets distance. This is because our tool-set uses a general concept of weight and when providing additional information such weight can assume any quantitative value—*i.e.* travel time, edge capacity, route cost, and so on.

About the population distribution, it is possible, for instance, to use a normal distribution peaked at the center of the city, multimodal distributions, or census data. This information can aid in the analysis of urban agglomerations if it is used to assign values to sets of nodes corresponding to the population density of the area that they belong to. Nevertheless, the set of inconsistencies would depend on the analysis of a specialist rather than being a self-explanatory result.

Also, despite being central to our problem formulation, the viability of redesigning a city is not suited for most cases. Furthermore, changing the topology of the network will alter the centrality of its elements, which will modify regions that attract vehicles and people. Our tool-set is not only to be used in redesigning a city but also on the initial design when all possibilities are open.

Finally, our proposal has open problems that support further studies: **(1)** the tool-set to track inconsistencies is categorical, then further algebra can aid in identifying the severity of a network inconsistency in a continuous, rather than binary, manner; **(2)** for simplicity's sake, we assumed the origin and destination of all paths as nodes of the network; such nodes are street intersections, which might not be real-world points of interest, requiring the addition of new nodes.

4 Conclusion

This paper was instantiated as a set of mathematical formalisms and algorithms to track and reduce distance-based inconsistencies improving access *to/from* points of interest in a city. Beyond the mathematical formulation, we provided a proof of concept and case studies, all of which indicate that our tool-set is able to suggest better placements for points of interest at the same time that it improves the access to the majority of the nodes of a city by reducing its inconsistencies.

More specifically, our contributions are in the definition of a concept based on intrinsic problems to urban structures that are caused by the misallocation of points of interest in cities; also, in two algorithms that were devised to track and reduce inconsistent nodes in complex networks; and, finally, in a case study, in which we show how our tool-set and algorithms can aid planners and designers.

In summary, our methods were proved empirically and formally, granting potential for prompt contribution and for opening new research questions. In addition, as a future work, we shall embrace link prediction methods for suggesting relocations in the network topology, *i.e.* proposing variations in the flow's direction, in the task of looking for a better topological setting for a city.

Acknowledgment. We would like to thank the Brazilian agencies CNPq (167967/2017-7), FAPESP (2016/17078-0 and 2017/08376-0) and CAPES that fully supported this research.

References

1. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.: Complex networks: structure and dynamics. Phys. Rep. **424**(4–5), 175–308 (2006)
2. Porta, S., Latora, V., Wang, F., Strano, E., Cardillo, A., Scellato, S., Iacoviello, V., Messora, R.: Street centrality and densities of retail and services in Bologna, Italy. Environ. Plan. B: Plan. Des. **36**(3), 450–465 (2009)
3. Masucci, A.P., Stanilov, K., Batty, M.: Limited urban growth: London's street network dynamics since the 18th century. PLoS One **8**(8), e69469 (2013)
4. Kaczynski, A.T., Koohsari, M.J., Stanis, S.A.W., Bergstrom, R., Sugiyama, T.: Association of street connectivity and road traffic speed with park usage and park-based physical activity. Am. J. Health Promot. **28**(3), 197–203 (2014)
5. Sopan, A., Rey, P.J., Shneiderman, B.: The dynamics of web-based community safety groups: lessons learned from the nation of neighbors. IEEE Sig. Process. Mag. **30**(6), 157–162 (2013)
6. Corcoran, P., Jilani, M., Mooney, P., Bertolotto, M: Inferring semantics from geometry. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2015. ACM (2015)
7. Barthélemy, M., Flammini, A.: Modeling urban street patterns. Phys. Rev. Lett. **100**(13), 138702 (2008)
8. Zhong, C., Arisona, S.M., Huang, X., Batty, M., Schmitt, G.: Detecting the dynamics of urban structure through spatial network analysis. Int. J. Geogr. Inf. Sci. **28**(11), 2178–2199 (2014)
9. Crucitti, P., Latora, V., Porta, S.: Centrality measures in spatial networks of urban streets. Phys. Rev. E **73**(3), 036125 (2006)
10. Costa, L.F., Travençolo, B.A.N., Viana, M.P., Strano, E.: On the efficiency of transportation systems in large cities. EPL (Europhys. Lett.) **91**(1), 18003 (2010)
11. Strano, E., Nicosia, V., Latora, V., Porta, S., Barthélemy, M.: Elementary processes governing the evolution of road networks. Sci. Rep. **2**, 296 (2012)
12. Spadon, G., Gimenes, G., Rodrigues Jr, J.F.: Identifying urban inconsistencies via street networks. In: International Conference on Computational Science, ICCS 2017, 12–14 June 2017, Zurich, Switzerland, vol. 108, pp. 18–27. Elsevier BV (2017)
13. Li, X., Parrott, L.: An improved genetic algorithm for spatial optimization of multi-objective and multi-site land use allocation. Comput. Environ. Urban Syst. **59**, 184–194 (2016)
14. Viana, M.P., Costa, L.F.: Fast long-range connections in transportation networks. Phys. Lett. A **375**(15), 1626–1629 (2011)
15. Travençolo, B., Costa, L.F.: Accessibility in complex networks. Phys. Lett. Sect. A: General, Atomic Solid State Phys. **373**(1), 89–95 (2008)
16. Crucitti, P., Latora, V., Porta, S.: Centrality in networks of urban streets. Chaos: an interdisciplinary. J. Nonlinear Sci. **16**(1), 015113 (2006)
17. Porta, S., Crucitti, P., Latora, V.: The network analysis of urban streets: a dual approach. Phys. A **369**(2), 853–866 (2006)
18. Cardillo, A., Scellato, S., Latora, V., Porta, S.: Structural properties of planar graphs of urban street patterns. Phys. Rev. E **73**(6), 066107 (2006)

Topological Street-Network Characterization Through Feature-Vector and Cluster Analysis

Gabriel Spadon(✉), Gabriel Gimenes, and Jose F. Rodrigues Jr.

University of Sao Paulo, Sao Carlos, SP, Brazil
spadon@usp.br, {ggimenes,junio}@icmc.usp.br

Abstract. Complex networks provide a means to describe cities through their street mesh, expressing characteristics that refer to the structure and organization of an urban zone. Although other studies have used complex networks to model street meshes, we observed a lack of methods to characterize the relationship between cities by using their topological features. Accordingly, this paper aims to describe interactions between cities by using vectors of topological features extracted from their street meshes represented as complex networks. The methodology of this study is based on the use of digital maps. Over the computational representation of such maps, we extract global complex-network features that embody the characteristics of the cities. These vectors allow for the use of multidimensional projection and clustering techniques, enabling a similarity-based comparison of the street meshes. We experiment with 645 cities from the Brazilian state of Sao Paulo. Our results show how the joint of global features describes urban indicators that are deep-rooted in the network's topology and how they reveal characteristics and similarities among sets of cities that are separated from each other.

Keywords: Network topology · Feature vector · Cluster analysis

1 Introduction and Related Works

Complex networks are used to shape real-world systems, *e.g.* networks of protein interaction, street meshes, and subway lines. These networks, as mathematical models, stand out due to their algebraic properties and computing potential, with analytical applicability to support cognitive processes of decision-making [1]. Through metrics and methods based on topology and/or geometry, it is possible to identify characteristics of interest that are not obvious for human inspections based on reading; this is because the networks may be wide (high number of vertices), intricate (high number of edges), or may hold non-trivial patterns and attributes whose observation depends on the application of algorithms.

In the specific case of the representation of street networks, complex networks describe factors related to the displacement of individuals, allocation of services,

Y. Shi et al. (Eds.): ICCS 2018, LNCS 10860, pp. 274–287, 2018.
https://doi.org/10.1007/978-3-319-93698-7_21

the improvement of tasks related to transport, and even to the study of factors from collective behavior, when the network is weighted by the associated data. In this regard, we observed a lack of methods to characterize groups of cities by means of the features that can be extracted from their topology, which is the aim of this research. This methodology has potential to enhance the understanding of an urban space and to explain the reason why cities share properties of interest.

To this end, we developed a methodology composed of *Data Acquisition and Preparation*, *Feature Extraction and Selection*, and *Feature Vector Analysis*. We analyzed 645 cities from the state of Sao Paulo, aiming to provide comprehension of peculiarities from different cities by interpreting global network-characteristics. These cities are representations of street meshes that were extracted from digital maps, such that they were gathered and analyzed by using machine-learning methods of feature extraction, multidimensional projection, and cluster analysis. In order to demonstrate our methodology, we investigate the following hypotheses: **(A)** *the network topology is a tool-set that can reveal groups of cities with similar characteristics, potentially revealing disparities*; **(B)** *although cities may share administrative boundaries with others, they cluster with cities with no apparent geographical similarity*; and, **(C)** *there might be interesting correlations between urban and/or territorial indicators and the features extracted from the street-network topology of a given set of cities*. The answering of such assumptions allows us to render better analysis of urban agglomerations by helping in the understanding of cities by comprehending how they are arranged within the geographical extent of their territorial boundaries.

Aiming to solve questions related to the urban scenario, a vast number of studies have been conducted to explain cities considering their intense flow of vehicles [2] and collective behaviors [3], while others analyzed the accidents density in street networks [4] and the discrepancies between cities driven by their urban indicators [5]. Furthermore, some authors investigated metrical and analytical methods applied to cities [6,7], others approached the assistance to the urban planning and design [8–10], and there are those who advanced with facility-location analysis and planning in street meshes [11]. However, although cluster analysis has been less focused [12,13], it is still an important toolset [14].

Two state-of-the-art works used clustering techniques to analyze groups of cities, but both of them left open questions to be explored. The first one had the intention to measure the similarity among ten European cities [12], while the second one performed an eye-based cluster evaluation considering the proximity and overlap of 1,150 cities, mainly from the Anglo-Saxon America [13]. Their lack of proficiency is mainly because they do not employ clustering algorithms in the same fashion that we do, including validation metrics and analytical indicators.

In this paper, we contribute with a methodology that advances the analysis of cities modeled as complex networks. To present our contributions, this paper is organized as follows: Sect. 2 displays our methodology while explaining the validation of its results; Sect. 3 discusses the results about the applicability of the proposed methods; and, Sect. 4 presents the conclusions and final remarks.

2 Methodology

Our methodology is based on the intersection of methods of data **A**cquisition, **M**odeling, and **C**omputation, and it follows a process flow depicted in Fig. 1.

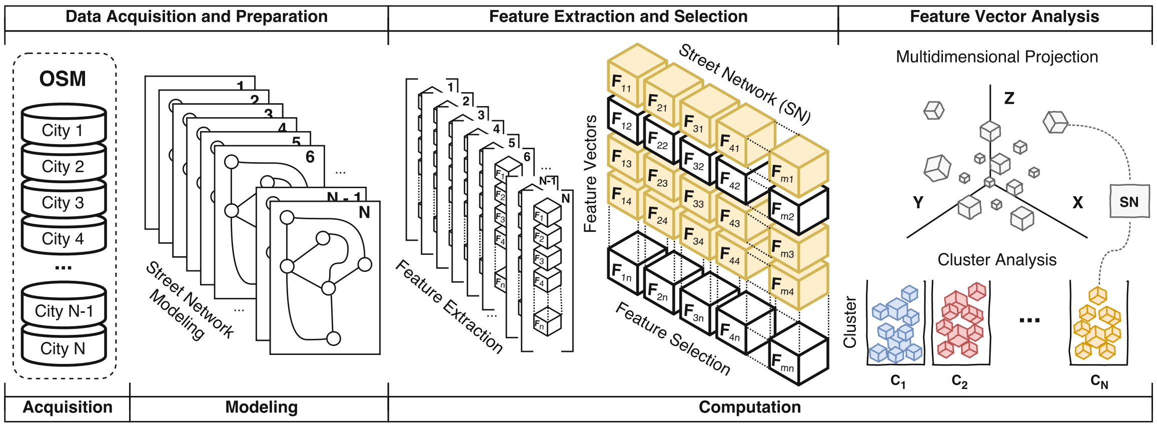


Fig. 1. Methodology for street-network characterization through feature-vector and cluster analysis based on data **A**cquisition, **M**odeling, and **C**omputation. The methodology starts by acquiring digital maps of cities from the OpenStreetMap (OSM), such maps are used for the modeling of complex networks. The resulting networks are used in the processes of extraction and selection of topological-features. These features are analyzed according to data-mining methods of multidimensional projection and cluster detection.

2.1 Preliminaries

Hereinafter, we represent complex networks as distance-weighted directed graphs. Notice that, despite different, complex networks and graphs are considered to be equivalent. A graph $\mathbf{G} = \{V, E\}$ is composed of a set of $|V|$ nodes and a set of $|E|$ edges. Furthermore, each edge $e \in E$ is known to be an ordered pair $\langle o, d \rangle$, in which $o \in V$ is named *origin* and $d \in V$ is named *destination*, $o \neq d$. We provided to the edges a double-precision floating-point weight d_{od}, which refers to the *great-circle distance* between node o and node d. The *great-circle distance* refers to the Euclidean distance between two points on the surface of a sphere; which in our case, the sphere is a projection of the Earth.

2.2 Data Acquisition and Preparation

For each one of the 645 cities from the Brazilian state of Sao Paulo, we got their administrative boundaries and indicators related to territorial extension and demography from the Brazilian Institute of Geography and Statistics (IBGE)[1]. The boundaries served as shapefiles to crop data obtained from OpenStreetMap (OSM)[2], which is an open data repository and a social network of collaborative

[1] www.ibge.gov.br.

[2] www.openstreetmap.org.

street mapping. The OSM's data describe real-world abstractions represented by georeferenced objects. These objects are described by means of its relations, which, in turn, refer to the streets (edges) and crossings (nodes) of a city, which were turned into complex networks where the edges intersect only at the nodes.

2.3 Feature Extraction and Selection

Metrics of graphs, referred to as *features*, can be divided into local and global [15]; local metrics describe properties of individual elements that form the network,

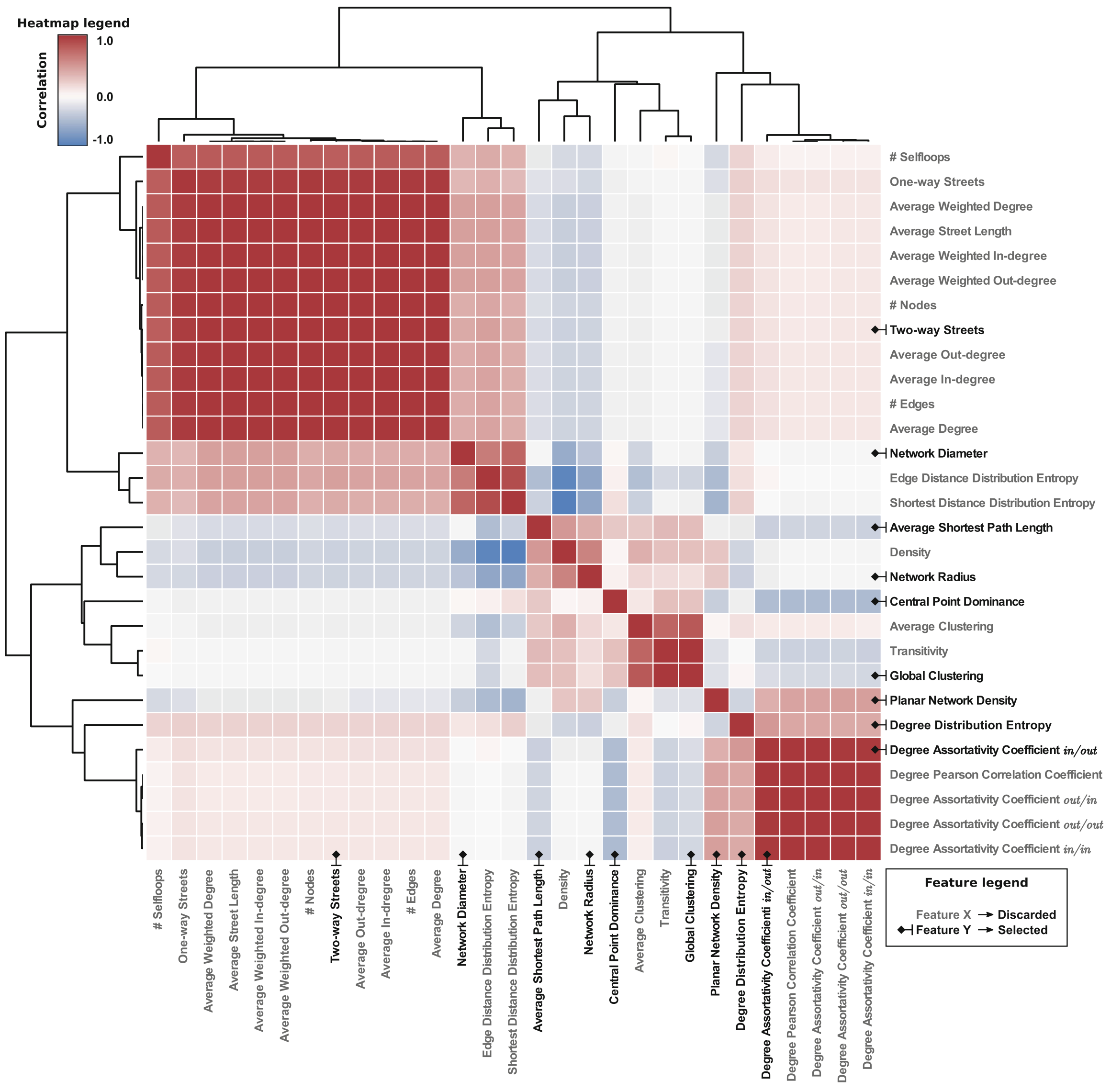


Fig. 2. A visualization of the mutual-correlation matrix of all the metrics we considered. The color describes the correlation between pairs of features. The metrics were hierarchically grouped through a dendrogram by means of the correlation of their values. Consequently, correlated metrics tend to stay in the same group; non-correlated metrics tend to be in separated groups. Additionally, the metrics we selected are colored in black and highlighted by a diamond marker. (Color figure online)

while global metrics characterize the whole network by a single value that is computed considering all of their elements. We rather use global metrics than local ones because they allow straightforward comparison between cities.

In order to gather these metrics, we designed a feature extractor to calculate a feature vector from any complex network given as input. First, we selected various metrics as candidates to render characteristics about cities, from which 29 metrics were chosen by their potential in providing insights about a given street network (see Fig. 2 for details). Such metrics were selected because they are linked to the network topology, which describes the streets of the cities.

After collecting all the metrics, we removed the non-relevant ones based on their mutual correlation. We computed the Pearson correlation coefficient [16] for each pair of metrics; such coefficient is defined in the interval $[-1.0, 1.0]$ where the extreme values indicate, respectively, the maximum negative and positive correlation, while 0.0 indicates no linear correlation at all. Following, we removed all the metrics with strong mutual correlation as indicated by the Pearson correlation in the interval $[-0.5, 0.5]$. In cases where any two metrics are outside this interval, one of the metrics was randomly discarded. Such process of metrics selection ensures that just metrics that are unique and non-related with the others will be used to describe the cities. Other processes of feature selection can be used in this step; even the multidimensional projection by itself can provide reasonable results. Notice that, the reduction of the dimensionality of the data was not our main priority, but rather to find the most complete set of metrics, that is the one that better characterizes the networks, without including redundant information; and, to this end, features correlation plays an import role. All metrics are depicted in Fig. 2; the ones that remained, 9 out of 29, were highlighted and are defined according to Costa *et al.* [17], as follows:

Degree Distribution Entropy ($\mathcal{H}$). The degree distribution of a network describes the probability of finding a vertex with a given degree. Whereas, the entropy represents the amount of uncertainty and randomness in a certain piece of information. By using the entropy in a city degree distribution, we can measure the uncertainty between street connections. Equation 1 describes such metric, where P_k represents the ratio of nodes with degree k.

Average Shortest Path ($\mathscr{L}$). It quantifies the average of all shortest paths (d^S_{ij}) that link all the pairs of nodes in a complex network (Eq. 2), it is used to quantify the capacity of locomotion through the shortest paths of a city.

$$\mathcal{H} = -\sum_{k=0}^{\infty} P_k \times \log(P_k) \quad (1) \qquad\qquad \mathscr{L} = \frac{\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} d^S_{ij}}{|V|(|V|-1)} \quad (2)$$

Degree Assortativity Coefficient ($\mathcal{R}$). It refers to the *in and/or out* degree correlation between pairs of nodes. That is, positive values indicate that nodes with similar degrees tend to connect to each other, while negative values indicate the same, but regarding nodes with different degrees. It can be understood as the probability of moving from an unimportant street to an important one based only on the number of adjacent streets to both of them. Equation 3 uses e_{xy} to refer to the fraction of edges that join together vertices with degree x and y, a_x and b_y to the fractions of edges that start and end at vertices with degree x and y; and, σ_a and σ_b to the standard deviations of the distributions a_x and b_y.

Eccentricity ($\mathcal{E}$). This metric is a local one, measuring for a vertex the longest shortest distance between all the other vertices of a given graph [18] (see Eq. 4). In a global perspective, the greatest eccentricity from a graph is known to be the *network diameter*, while the smallest one is regarded as the *network radius*. They can reveal cities that may suffer from access issues by being sparse if the radius of a network is too small when compared to its diameter.

$$\mathcal{R} = \frac{\sum_{xy} xy(e_{xy} - a_x b_y)}{\sigma_a \sigma_b} \quad (3) \qquad \mathcal{E}_i = \frac{1}{max\{d^S_{ij} | \forall j \in V\}} \quad (4)$$

Planar Network Density ($\mathcal{D}$). The density of a planar graph is defined as the ratio between the number of edges E and the number of all possible edges in a network with N nodes with no intersecting edges. It can be used to describe how dense is the street mesh of a city or a neighborhood. The metric is unique to each network, once the position of the nodes interferes in the number of edges. It is an algorithmic adaptation of the graph density [19], described in Eq. 5.

Central Point Dominance (C^P_D). This metric assesses the global centrality of a whole network by means of its network's betweenness deviation, which is a distance-based centrality metric. Values close to 0 indicate plenty of distance-efficient routes similar to the shortest one; whereas, values close to 1 indicate that the network might become vulnerable without its central node because the node might be used to connect different components, serving as an access point (*e.g.* bridges and tunnels). In Eq. 6, $\bar{v}$ is the node with the highest betweenness and $\mathcal{B}(v)$ is the normalized betweenness of the node v that lies in the range $[0, 1]$.

$$\mathcal{D} = \frac{|E|}{|N|\,(|N| - 1)} \quad (5) \qquad C^P_D = \frac{\sum_v^{|V|} \mathcal{B}_{\bar{v}} - \mathcal{B}_v}{|V|(|V| - 1)} \quad (6)$$

Two-way Streets ($\mathcal{T}_w$). It refers to the number of double edges in a network, which are edges that provide two-way routes between the same pair of nodes. This metric follows Eq. 8, in which f_{ij} is a clause-based auxiliary function.

Global Clustering ($\mathcal{G}_c$). The metric, which is described by Eq. 8, consists of the *fraction* of the number of triangles $\mathbb{N}_\triangle$ and triples $\mathbb{N}_3$ of the network. It refers to how the streets tend to cluster in the crossings of a given city, such that the greater the value the more possibilities of locomotion in fewer steps.

$$\mathcal{T}_w = \frac{\sum_{\langle i,j \rangle}^{E} f_{ij}}{2}, \quad f_{ij} = \begin{cases} 1, & \langle j, i \rangle \in E \\ 0, & otherwise \end{cases} \quad (7) \qquad \mathcal{G}_c = \frac{(3 \times \mathbb{N}_\triangle)}{\mathbb{N}_3} \quad (8)$$

2.4 Feature Vector Analysis

In this step, we focused on two methods from the data mining literature, the first one of multidimensional projection and the second one of clustering detection. Multidimensional projection allows the visualization of data by reducing its dimensional space, revealing particularities and behaviors to be explored through cluster-based analysis. Cluster analysis, in turn, focuses on the study of data interactions, inferring that two elements are similar because they are in the same cluster or dissimilar because they are in different ones. Consequently, the combination of these two methods contributes to the assessment of cities by their potential to reveal patterns that are not evident through an eye-based analysis.

Regarding multidimensional projection, our methodology consists of using two techniques [20]; the first one is named Isomap and the second one is known as Principal Components Analysis (PCA). Isomap is a nonlinear dimensionality reduction technique, which provides an embedding in a lower dimension while maintaining the geodesic distance between the data elements. Contrarily, PCA is a linear technique, which uses orthogonal conversions to turn a set of variables into linearly uncorrelated values with the largest possible variance. To choose both techniques, we used knowledge about the domain; we have kept track of some already-known dissimilar cities, seeking for approaches to distinguish them.

In the cluster analysis part, we used the technique KMeans [21], which splits the data into groups of equal variance, minimizing the sum-of-squares distance within clusters. The KMeans algorithm assumes that (i) the distribution of features within each cluster resembles spheres, which means that all features have equal variance and they are independent of each other; (ii) regarding the cluster size, the dataset is balanced; and, (iii) the density of the clusters is similar. The dataset we used consists of uncorrelated values and balanced instances of feature vectors, all of which have quasi-equal variance, meeting the algorithm requirements. In addition, KMeans is widely used in the related literature due to its robustness, versatility, and scalability. To validate our results we considered cluster quality metrics [22]. Their focus is to analyze the similarity between elements that have been assigned to the same cluster. We used a combination of the Silhouette score [23] and the Dunn index [24]; both of which are known to be internal-quality metrics, not requiring a pre-labeled dataset. The Silhouette is defined between $[-1, 1]$ for each cluster, the closer to 1 the better; it measures the cohesion and separation of clusters by evaluating how similar an element is in its own cluster when contrasted to other clusters. To further enhance the reliability of our analysis, we applied the Dunn index, which is a cluster distance-based quality metric that measures the separation among clusters, whose values are in between of $[0, \infty]$. In cases when the Dunn's index distance is greater than one, there is little or none cluster overlapping. Using both together, we have a double validation of quality by means of cohesion and separation of our set of clusters.

3 Results

3.1 Relationship of Population Size and Topological Features

With regard to the population density — see Fig. 3 for details —, the majority of the cities in our dataset is of tiny or small size, but the dataset has a substantial number of medium-sized cities and a small number of large-sized ones, including Sao Paulo — the biggest Brazilian city. Prior analyses can be done by observing Fig. 4, where cities (depicted as points) were sized by their number of nodes.

A first evidence that the topological features we selected can describe relevant knowledge about cities is the fact that Sao Paulo is isolated from the other ones in the PCA projection. A similar fact can be observed on a small scale considering the large-sized city of Campinas and the medium-sized cities of Marilia and Piracicaba, which are apart from the main group of cities located on the left part of the image. We believe that such behavior is connected to the demographics of the cities. On a large scale, topological features can predict demographic characteristics of a city, whereas, on a small scale, they can reflect the neighborhoods that are densely or sparsely populated. For a less unbalanced view, we removed Sao Paulo from the dataset, depicting in Fig. 5 the normalized values of the feature vectors of the cities that remained using both PCA and Isomap techniques.

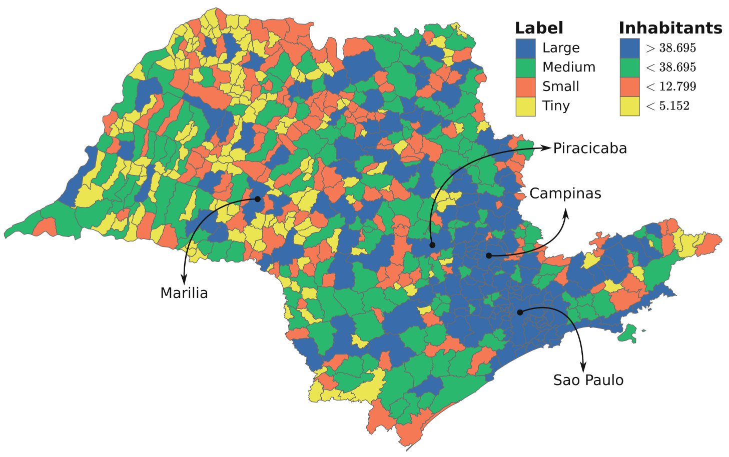


Fig. 3. Urban indicator related to the population density of the cities of the state of Sao Paulo. The cities were divided into four classes that describe the number of inhabitants of each one of them.

The two techniques show us that the majority of the data is concentrated in a small region, while the rest of it is sparse and distributed along the axes. The main difference between both of them is that Isomap implies multiple areas with considerable density, while PCA has a single dense area and many sparse data. This is evidence that tiny and small-sized cities tend to cluster isolating medium and large-sized cities that are too different from them. Despite the fact that such cities tend to cluster, Isomap shows that they have particularities that make them split into smaller clusters inside a bigger one. Also, it is safe to infer that by being scattered, medium-sized and large-sized cities have no clear pattern, but still, they may share common characteristics to be further explored with clustering algorithms. Even so, we can show, by using correlation, that the network's demography can be inferred from the city's topology – see Fig. 6.

To prove that the network's demography can be inferred from the city's topology, we measured the relationship between the topological features and the

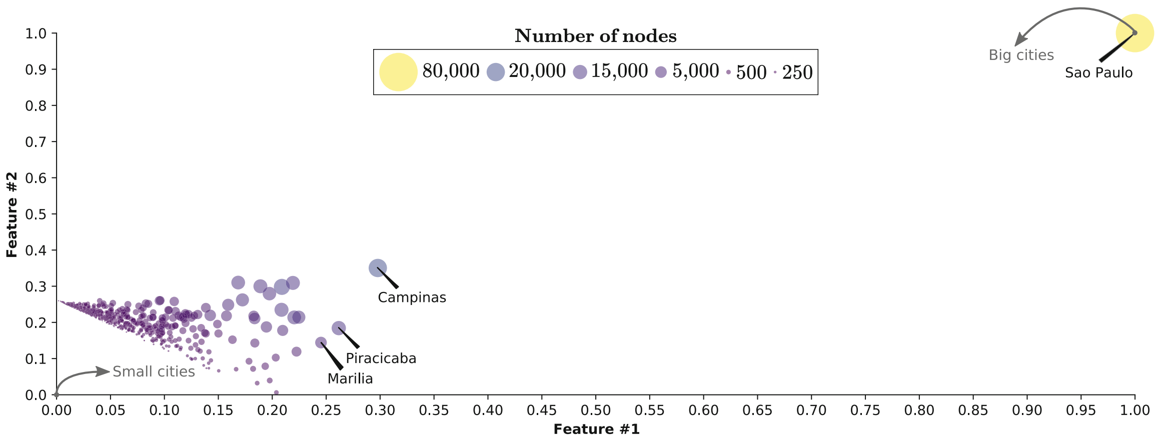


Fig. 4. Projection of feature vectors in two dimensions by using PCA; the size of the points refers to the number of nodes (intersections) in the cities' complex-network. The projected features reveal that the city of Sao Paulo (on the right-hand side) is an outlier when compared to the others (on the left-hand side).

demography by means of correlation. To this end, we reduced the dimensionality of the feature vectors of each city to one, using both techniques, PCA and Isomap, resulting in one single value for each one of the 645 cities. Next, we correlated such values with the size of their population. As a result, we got 0.803 and 0.799 of correlation for PCA and Isomap, respectively. Both values indicate that the data has a strong correlation, allowing us to state that in the case of the Brazilian state of Sao Paulo, topological features and demographics are strongly correlated. Such pattern opens doors for new investigations, as the ones placed by the dynamics of the social behavior; as in the case of criminality and mobility.

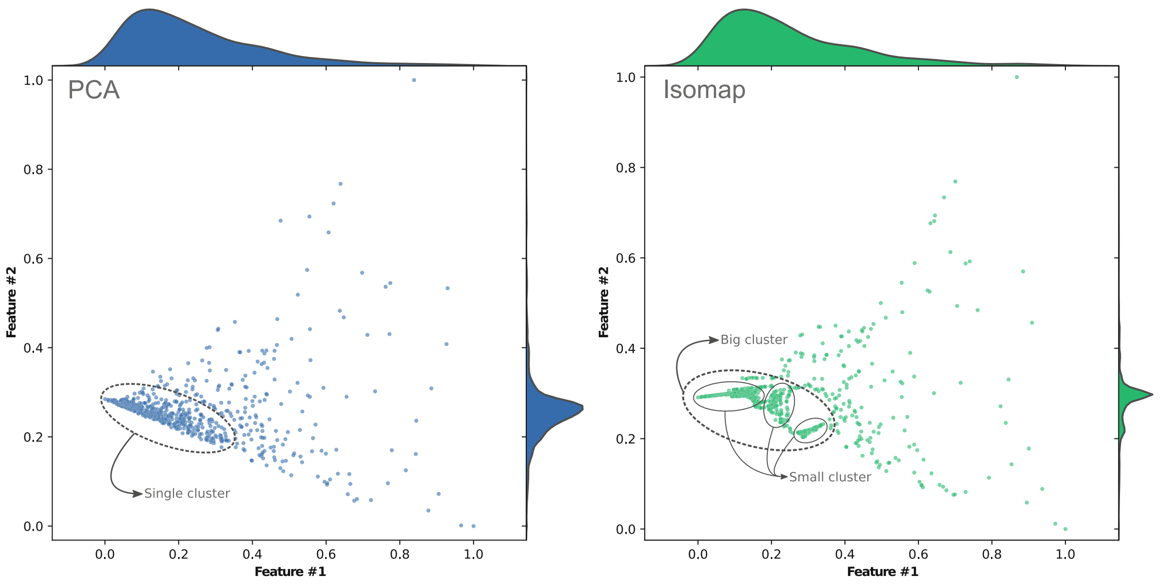


Fig. 5. Projection of the features, excluding Sao Paulo, using PCA and Isomap. PCA shows a single dense area with many sparse data, while Isomap shows multiple dense areas together with several sparse data. As a consequence, PCA implies a single cluster while Isomap points to an inherent hierarchy of clusters.

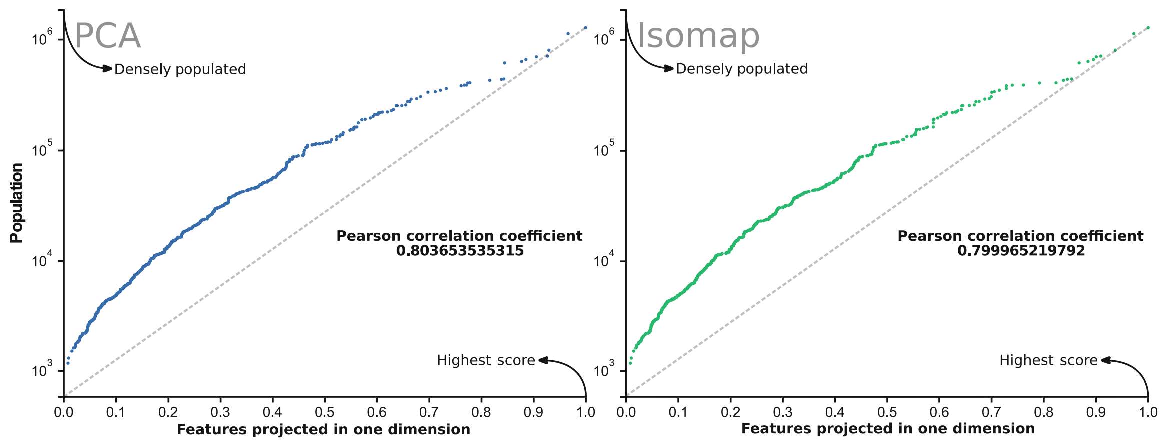


Fig. 6. Correlation test between the population density and a one-dimension projection of the cities' topological-features regarding both PCA and Isomap. Both images show a strong correlation, revealing that, on a large scale, the topological features of the cities can indicate, or even predict, their demography.

3.2 Relationship of Cluster Assignment and Territorial Extension

The cluster analysis aimed at the identification of the best number of clusters to describe our dataset. Consequently, we exhaustively tested the KMeans' cluster-quantity parameter from 2 to 644 clusters — the total number of cities without considering Sao Paulo. During the test, *we were seeking for the greatest average Silhouette score (AVG) only when the Dunn index (DNN) was larger than one.*

The previous experiment suggests that the best way to split our data is into two clusters. Such configuration has an AVG of 0.59 and a DNN of 1.10 (see Fig. 7). When dividing the data into two, the clusters are better balanced rather than when considering Sao Paulo — a big outlier — as part of the dataset.

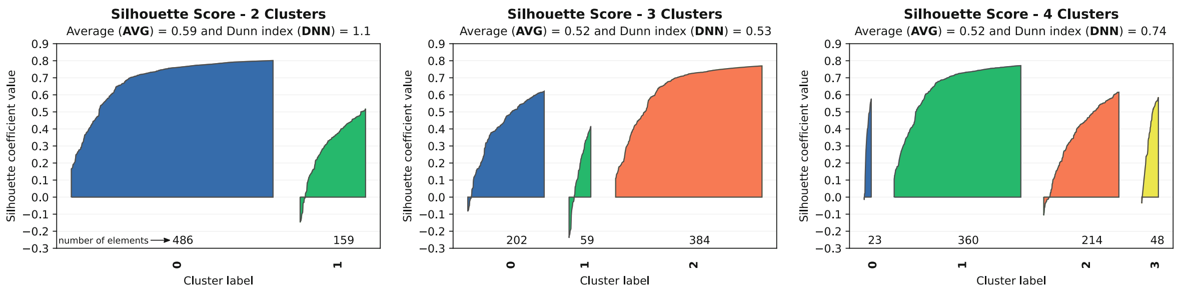
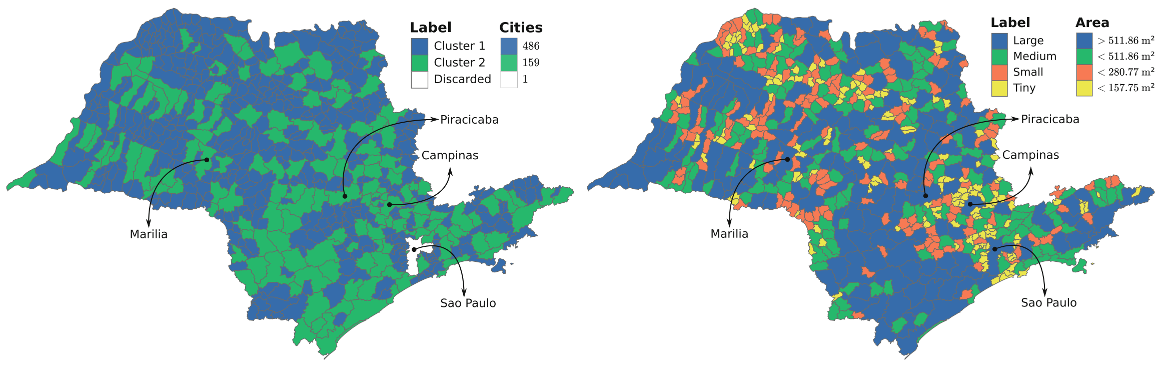


Fig. 7. Silhouette analysis of the subset of our data without Sao Paulo, in which clusters are represented as color-coded polygons. In each scenario that we have tested, the results were validated according to the Dunn index together with the Silhouette score. Although we have depicted the first three tested scenarios, which are also the best ones, the experiment considers a total of 643 scenarios. (Color figure online)

Subsequently, we investigated the reason why the cities were better arranged into only two clusters. By analyzing indicators related to population and territory extension, we found that 61.20% of the state's population is in the first cluster and 38.80% is in the second one (see Fig. 8a), and that the first cluster is mainly populated with cities that are considered to be of tiny or small territorial extension (see Fig. 8b), while the second cluster has the opposite behavior. Bearing in mind that our dataset does not imply any relationship between indicators of territorial extension and population density, we concluded that the relation that favored two clusters, as the arrangement with best values of Silhouette and Dunn index, was the territorial extension of the cities. Hence, we found evidence that there is a significant relationship between topological features, territorial extension, and demographics of the cities of Sao Paulo state.



(a) Clustering Sao Paulo's cities. **(b)** Sao Paulo's territorial extension.

Fig. 8. Investigating cities through clustering techniques; Fig. 8a shows the results of the clustering of topological features when removing the Sao Paulo city from the dataset; this layout has an average Silhouette of 0.59 and Dunn Index of 1.1. Figure 8b describe the area within the cities' administrative boundaries.

The relationship between the cluster arrangement and territorial extension can be understood as the way cities organize within their available space. In fact, regarding the territorial extension, 30.51% of the cities from the first cluster are tiny-sized, 31.13% are small-sized, 25.78% are medium-sized, and 12.58% are large-sized; whereas, 7.59% of the ones from the second cluster are tiny-sized, 6.32% are small-sized, 22.78% are medium-sized, and 63.29% are large-sized. Therefore, cities in the first cluster can be considered smaller and heavily populated, while the ones in the second cluster are larger and less populated.

3.3 Discussions on Results Generalization

We have chosen to present a joint of direct findings and analytical conclusions in our results section. This was done so that one can follow the practical application of the proposed methodology in a way that can be adapted and generalized for different domains and scenarios. Our methodology can also be used in non-urban applications, such as in the characterization of the topology in any group of complex networks, however, depending on the specificities of the domain, it may be necessary to use different network metrics and features to be more effective.

Additionally, while our findings cannot be generalized for any set of cities, we believe that the proposed methodology can be used to find non-trivial properties in different urban scenarios and not only to the cities that shape the state of Sao Paulo. Comprising a straightforward framework of analysis that can be useful to the academic community and cities' governing body, *e.g.* planners and designers.

Finally, our proposal has intricacies that can be explored in further studies: **(1)** using hierarchical clustering to reveal additional knowledge, which may also demand prior expertise about the cities (*e.g.* history and geography); and, **(2)** using more complex feature selection techniques such as fractal-dimension based methods or by applying ones related to mutual information. Notice that, this refinement might reveal other patterns of the data, but will not change the ones we discussed; and, **(3)** including non-topological features to capture different characteristics of cities, enhancing our methods capabilities and its versatility.

4 Conclusion

In this paper we proposed a three-folded method encompassing the data **A**cquisition, **M**odeling, and **C**omputation. Furthermore, our methodology comprises the following phases: *Data Acquisition and Preparation*, *Feature Extraction and Selection*, and *Feature Vector Analysis*; culminating in the use of multidimensional projection and cluster analysis algorithms to assess feature vectors of complex-network metrics. To validate our proposal, we investigated the following hypotheses: **(A)** *the network topology is a tool-set that can reveal groups of cities with similar characteristics, potentially revealing disparities*; **(B)** *although cities may share administrative boundaries with others, they cluster with cities with no apparent geographical similarity*; and, **(C)** *there might be interesting correlations between urban and/or territorial indicators and the features extracted from the street-network topology of a given set of cities.* Such hypotheses were investigated by analyzing relations between 645 cities that constitute the Brazilian state of Sao Paulo. Our main findings confirm the hypotheses of our work, allowing us to state that, on a large scale, the topological features of the cities can indicate, or even predict, their demography and that cities group themselves by means of their territorial extension, which describes the way that cities organize within their available space. Therefore, our main contributions are: **(i)** *the description of how the network topology is capable of revealing groups of cities with similar characteristics*; **(ii)** *the correlation analysis between the demography of the cities and their features*; and, **(iii)** *the discussion of why cities cluster with other cities distant apart instead of with those that they share boundaries with.* As a future work, we will measure the similarity between cities by means of non-topological features, looking for discrepancies in the collective behavior that emerges from this same set of cities.

Acknowledgment. We are thankful to CNPq (grant 167967/2017-7), FAPESP (grants 2014/25337-0, 2016/17078-0 and 2017/08376-0) and CAPES that supported this research.

References

1. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.: Complex networks: structure and dynamics. Phys. Rep. **424**(4–5), 175–308 (2006)
2. Masucci, A.P., Stanilov, K., Batty, M.: Limited urban growth: London's street network dynamics since the 18th century. PLoS ONE **8**(8), 1–10 (2013)
3. Blumer, H.: Social problems as collective behavior. Soc. Probl. **18**(3), 298–306 (1971)
4. Anderson, T.K.: Kernel density estimation and K-means clustering to profile road accident hotspots. Accid. Anal. Prev. **41**(3), 359–364 (2009)
5. Grauwin, S., Sobolevsky, S., Moritz, S., Gódor, I., Ratti, C.: Towards a comparative science of cities: using mobile traffic records in New York, London, and Hong Kong. In: Helbich, M., Jokar Arsanjani, J., Leitner, M. (eds.) Computational Approaches for Urban Environments. GE, vol. 13, pp. 363–387. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-11469-9_15
6. Crucitti, P., Latora, V., Porta, S.: Centrality measures in spatial networks of urban streets. Phys. Rev. E: Stat. Nonlinear Soft Matter Phys. **73**(3), 1–6 (2006)
7. Costa, L.F., Travençolo, B.A.N., Viana, M.P., Strano, E.: On the efficiency of transportation systems in large cities. EPL (Europhys. Lett.) **91**(1), 1–10 (2010)
8. Porta, S., Latora, V., Wang, F., Strano, E., Cardillo, A., Scellato, S., Iacoviello, V., Messora, R.: Street centrality and densities of retail and services in Bologna, Italy. Environ. Plan. B: Plan. Des. **36**(3), 450–465 (2009)
9. Strano, E., Nicosia, V., Latora, V., Porta, S., Barthélemy, M.: Elementary processes governing the evolution of road networks. Sci. Rep. **2**, 1–8 (2012)
10. Spadon, G., Gimenes, G., Rodrigues-Jr., J.F.: Identifying urban inconsistencies via street networks. In: International Conference on Computational Science, ICCS 2017, 12–14 June 2017, Zurich, Switzerland, vol. 108, pp. 18–27. Elsevier (2017)
11. Li, X., Parrott, L.: An improved genetic algorithm for spatial optimization of multi-objective and multi-site land use allocation. Comput. Environ. Urban Syst. **59**, 184–194 (2016)
12. Strano, E., Viana, M., Costa, L.F., Cardillo, A., Porta, S., Latora, V.: Urban street networks, a comparative analysis of ten European cities. Environ. Plan. B: Plan. Des. **40**(6), 1071–1086 (2013)
13. Domingues, G.S., Silva, F.N., Comin, C.H., Costa, L.F.: Topological characterization of world cities. arXiv preprint arXiv:1709.08244 (2017)
14. Pan, G., Qi, G., Zhang, W., Li, S., Wu, Z., Yang, L.T.: Trace analysis and mining for smart cities: issues, methods, and applications. IEEE Commun. Mag. **51**(6), 120–126 (2013)
15. Scripps, J., Nussbaum, R., Tan, P.N., Esfahanian, A.H.: Link-based network mining. In: Dehmer, M. (ed.) Structural Analysis of Complex Networks, pp. 403–419. Springer, Heidelberg (2011). https://doi.org/10.1007/978-0-8176-4789-6_16
16. Chiang, C.: Statistical Methods of Analysis. World Scientific, Singapore (2003)
17. Costa, L.F., Rodrigues, F.A., Travieso, G., Boas, P.R.V.: Characterization of complex networks: a survey of measurements. Adv. Phys. **56**(1), 167–242 (2007)
18. Hage, P., Harary, F.: Eccentricity and centrality in networks. Soc. Netw. **17**(1), 57–63 (1995)
19. Brath, R., Jonker, D.: Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data. Wiley, Hoboken (2015)
20. Spiwok, V., Oborský, P., Pazúriková, J., Kenek, A., Králová, B.: Nonlinear vs. linear biasing in trp-cage folding simulations. J. Chem. Phys. **142**(11), 1–8 (2015)

21. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297 (1967)
22. Kremer, H., Kranen, P., Jansen, T., Seidl, T., Bifet, A., Holmes, G., Pfahringer, B.: An effective evaluation measure for clustering on evolving data streams. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2011, pp. 868–876. ACM, New York (2011)
23. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**(1), 53–65 (1987)
24. Dunn, J.C.: Well-separated clusters and optimal fuzzy partitions. J. Cybern. **142**(1), 95–104 (1974)

Computer-assisted city touring for explorers

Gabriel Spadon and Jose F. Rodrigues-Jr

University of Sao Paulo, Sao Carlos – SP, Brazil
spadon@usp.br, junio@icmc.usp.br

Abstract. The basic purpose of a map is to trace shortest paths between two locations in a city. However, this is not always what a user needs. Consider a tourist in an unknown city, he/she might want to trace routes to visit multiple landmarks while passing through the main streets of the city, possibly more than once through the same street. Such functionality is not yet available in online map services, which are prone to provide shortest paths that connect all landmarks. Rather, is of common interest a tour that puts together the most central streets (topologically speaking), minimizes the trajectory and, at the same time, passes through such landmarks. To cope with this problem, it is possible to investigate techniques of Center-Piece Subgraph, Absorbing Random Walk Centrality and Spanning Edge-Betweenness; such techniques can be used to find induced subgraphs that optimize centrality measures for a set of referential nodes or edges, *i.e.* landmarks or streets. The results shall be in the form of optimized algorithms, and how to integrate them into online systems. Studies in this line can succeed if they can guarantee timely scalability at the same time that they provide algorithms that produce tours (1) considering all the known destinations; (2) including the main streets of a city; and, (3) ensuring the shortest routes.

Keywords: Complex Network; Urban Structure; City Tour.

1 Problem Statement

Digital maps are becoming available to everyone, anywhere, through computers, smartphones, car assistants, and electronic devices in general. The most common use of such maps is to trace routes, or shortest paths, among different locations. However, this basic use is not always what one needs. In the case of a tourist visiting an unknown city, for instance, the user is not looking for the fastest and/or shortest path. Rather, he/she is looking for a tour that visits multiple landmarks (*i.e.* points of interest) while passing, possibly more than once, through the main streets of the city. That is, the user not only has several destinations, he/she desires a route that favors hotspots of the city in what is usually called city-tour. For instance, imagine a tourist is on the east side of the Central Park in New York and wants to go all the way down to the World Trade Center; the fastest and shortest path is to go through the monotonous 7th avenue. For a tourist, differently, the best pick might be to go through Broadway, despite its traffic and higher distance. The idea then is to find the tour that puts together

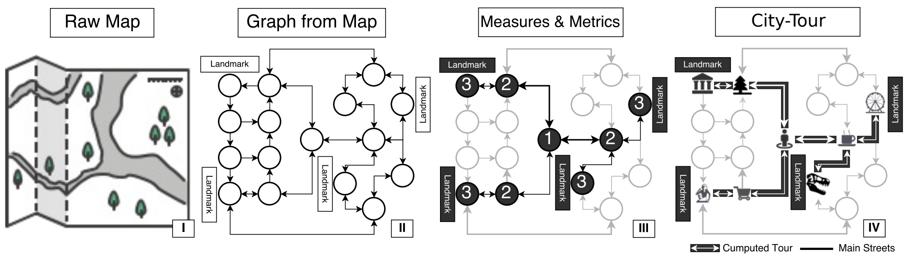


Fig. 1: An illustration of the methodology and desired results of the city-tour. It starts by **(I)** creating a graph from a map, **(II)** then the user informs some points of interest in the city; **(III)** these places are analyzed by means of their placement using metrics of graphs; **(IV)** then the algorithm returns a city-tour that passes through all of them and also through the main streets of the city.

the most central streets (topologically speaking), minimizes the trajectory and, at the same time, passes through points of interest. Differently, current systems are more likely to provide the user with shortest paths that connect all the points of interest. In other cases, the user does not even have destinations to visit; he/she wants to cruise through the city, discovering it iteratively. Such functionalities are not yet available in online map services and can be explored through complex-network tools by using a methodological process similar to the one presented in Figure 1.

2 Research Goals

The goal is to use graph-processing techniques to compute touristic routes with, or without, multiple destinations satisfying performance constraints. Given a city, or a region of a city, such routes might come as the answer to two queries: **(1)** to compute a multiple-destinations tour; or, **(2)** to compute a tour, even if the set of destinations is empty. The second query might sound odd, but it is not an unusual case, especially for explorers in unknown cities. Notice that, such queries depend on the investigation of techniques based on random-walk with restart [12], usually used to find hub nodes of a network, but that, by means of edge processing, might be used to find the most significant edges related to a set of nodes [7]. The queries also depend on techniques that identify the edges with the highest centrality, which, in the context of maps, must be concomitantly related to shortest paths so to reflect good routes — Hannah *et al.* [1] performed a study that explores these properties, but not considering the city-tour constraints.

3 Related Works

Besides the aforementioned work, others touched the issue of processing digital maps. Scellato *et al.* [8] investigated how it is possible to extract the backbone of a city using Spanning Trees based on Edge-Betweenness and Information Centrality; they proposed a method that allows extending the comprehension of the most important routes that affect the city flow, retails, land-use separation, and that impact upon collective behavior. However, the authors used a greedy algorithm, which fails in circumstances when the topology is not adequate to the greedy strategy, for instance, when the lengths of the streets strongly diverge in different regions. Additionally, Delling *et al.* [2] have focused on algorithms to solve problems from the commercial domain; this is the case when one needs to choose the best placement for a new store. Dibbelt, Pajor, and Wagner [3] explored a multi-modal common-route planning problem. These studies consider the possibility of using heterogeneous transportation to go from one point to the other (common-route problem), favoring only the criterion of shortest paths.

4 Suggested Methodology

Nowadays, digital maps can be extracted from many different sources, but some of them are not freely available. OpenStreetMap[1] is a collaborative street-mapping community and an open-source option for digital maps. Over OpenStreetMap, it is possible to investigate techniques related to the questions placed in Section 2. Such research questions can be explored by means of three techniques, as follows:

Center-Piece Subgraph (CEPS). The Center-Piece Subgraph technique summarizes a graph, producing an induced topology that connects a subset of referential nodes provided as input. It follows by using random-walks with restart and cost functions to measure the adequacy of the edges that will be part of the resulting topology [11]. The summarized graph is validated by analyzing the goodness of the graph nodes [6], such that, heuristically, the best simplification is the one that contains the essential elements according to a goodness criterion (as defined by Equation 1), where $r(q, j)$ is the goodness score for a given node j considering a query set $\mathbf{Q}$.

$$g(CP) = \sum_{j \,\in\, nodes(CP)} r(\mathbf{Q}, j) \tag{1}$$

The goodness criterion minimizes the sum of weights to connect all the edges in the subset while inducing a subgraph; however, it does not consider edge properties, leading to a subgraph with semantics related to a route in a map. The metric was not designed to provide a minimum set of edges, as necessary in multiple-destination routes, but only with as many edges as desired by the user.

Example. Let us consider an unknown city represented as a graph $\mathbf{G}(V, E)$ which has a couple of touristic attractions $\mathbf{T}$ and let T' be the subset of those that he/she wants to *visit* or, alternatively, *avoid*. The idea then is to extract a subgraph G' that confers the user a reduced network in which he/she can travel

[1] `www.openstreetmap.org`

back and forth to visit the desired destinations. The output of CEPS, in this particular case, shall be a subgraph indicating the easiest and most interesting way to travel from a source place t_i to a target one t_{i+1}, where $\{(t_i, t_{i+1}) \in T'\}$.

Absorbing Random Walk Centrality (ARwC). The Absorbing Random Walk Centrality technique is capable of evaluating the centrality of a subgraph considering a set of query nodes Q [5]; ARwC considers the number of steps needed to absorb a walker that starts from a query node $q \in Q$ and walks to all the other nodes C in the graph, denoted as $ac_Q^q(C)$. This metric is based on the *k-Arw-Centrality* optimization, which is an NP-hard problem; to solve that problem, ARwC uses a greedy approach that provides solutions with good approximation guarantees. Equation 2 presents the metric, whose result is the centrality of a set of nodes C with regard to a set of query nodes Q.

$$ac_Q(C) = \sum_{q \in Q} \frac{1}{|Q|} \times ac_Q^q(C) \tag{2}$$

Example. Consider an unknown city with some touristic attractions represented as a graph. It is possible to assess the centrality of these attraction points inside the subset T' using ARwC, which will provide a set of values $\{C_t^{rw}\ \forall\ t \in T'\}$ that represent the importance of an attraction with respect to the others. Consequently, the result of this process is a hierarchical city view, which can: **(1)** represent how critical a node is among the others; **(2)** quantify how drastically he/she needs to avoid critical nodes to improve his/her mobility in a city; and, **(3)** classify the touristic attraction which will receive more or fewer visits considering its positioning.

Spanning Edge-Betweenness (SEB). A Spanning Tree (ST) is a graph structure that contains all the nodes connected by a subset of non-cyclic edges. It can be produced with weighted or non-weighted edges. In the context of a map, an accurately computed ST (not necessarily the minimum) generates a *Backbone* representation of a city, which conforms to the problem of computing a tour without a set of destinations. To this end, SEB is a centrality metric computed by considering all the STs of a graph; it works by quantifying the number of times that each edge pertains to an ST. The result of this metric is a hierarchical formation of streets according to their importance.

SEB was firstly defined over undirected and weighted graphs to improve the analysis of phylogenetic trees [9]; it has been shown to be a powerful tool able to evaluate the most relevant edges of a graph that, if removed, might disrupt the network structure [10]; notice that, the metric is not only to be used in the analysis of phylogenetic trees but also over massive graphs with millions of nodes [4]. Equation 3 formally defines SEB, where τ_G is the number of STs for a graph G, and $\tau_G(e)$ is the number of different STs where edge e occurs.

$$\delta_G(e) = \frac{\tau_G(e)}{\tau_G} \tag{3}$$

Example. Suppose that it is desired to detect the most common and interesting route that connects the nodes of an unknown city. That is, one wants not only the streets with the minimum length, but also wants those that, given the network topology, render better routes more frequently. This process can be achieved by using SEB whenever it is possible to infer weights to the set of network edges.

5 Expected Results

The results shall be in the form of optimized algorithms, and how to integrate them into online systems, including how to extract and prepare the network (maps are not necessarily represented as graphs), how to provide input to these algorithms, and how to interpret the outputs. The results shall be validated through extensive testing over a significant number of representative cities. Studies in this line can succeed if they can process queries over a map within seconds, and if the algorithms produce tours that **(1)** consider all the destinations (if known); **(2)** include the main streets of a city (higher centralities); and, **(3)** minimize the length of the paths (good routes). It is possible to verify those conditions in large scale by using brute-force algorithms or multi-objective optimization techniques; and, in small scales, by considering case studies of known cities and their tours. Research following these lines shall pave the way for future map processing, turning tours into a novel concept on electronic-trajectory computing. The results have the potential to be reported in international conferences and journals of Data Mining, Transportation Systems, and Physics.

6 Further Research Lines

The possibilities of research and investigation are not limited to the calculation of city-tours. It is possible, for instance, to analyze different sets of cities by extracting feature vectors that describe the complex-networks of their tours. This is because feature vectors can describe cities by detailing their topology and structural peculiarities, which would enable further analysis based on clustering detection, similarity search, multidimensional projection, and fractal analysis. All of which have the potential to enhance the understanding both about cities and tourist behaviors in the face of different types of landmarks that can be found in a given city. More specifically, by investigating feature vectors through these tools, it is possible: **(i)** to analyze cities according to their similarities and differences; **(ii)** to determine characteristics shared between cities or groups of cities; and, **(iii)** to disclose routing problems that are exclusive to peculiar cities. In fact, all these activities help in the designing of the urban space, they also help in its improvement and comprehension because similar cities might share the valuable characteristics; meanwhile, exceptional cities might indicate routing problems that can be further studied through a different set of tools.

Acknowledgments

We would like to thank the Brazilian agencies CNPq, FAPESP, and CAPES that fully supported this research.

Bibliography

[1] Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P., Wagner, D., Werneck, R.F.: Route planning in transportation networks. In: Kliemann, L., Sanders, P. (eds.) Algorithm Engineering, pp. 19–80. Springer International Publishing (2016)

[2] Delling, D., Goldberg, A.V., Pajor, T., Werneck, R.F.: Customizable Route Planning in Road Networks. Transportation Science (may 2015)

[3] Dibbelt, J., Pajor, T., Wagner, D.: User-Constrained Multi-Modal Route Planning. In: 2012 Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX), pp. 118–129. Society for Industrial & Applied Mathematics (SIAM) (jan 2012)

[4] Mavroforakis, C., Garcia-Lebron, R., Koutis, I., Terzi, E.: Spanning edge centrality. In: Proceedings of the 24th International Conference on World Wide Web - WWW'15. pp. 732–742. ACM Press (2015)

[5] Mavroforakis, C., Mathioudakis, M., Gionis, A.: Absorbing random-walk centrality: Theory and algorithms. In: 2015 IEEE International Conference on Data Mining. pp. 901–906. IEEE (nov 2015)

[6] Rodrigues-Jr, J.F., Tong, H., Pan, J.Y., Traina, A.J.M., Traina, C., Faloutsos, C.: Large Graph Analysis in the GMine System. IEEE Transactions on Knowledge and Data Engineering **25**(1), 106–118 (jan 2013)

[7] Rodrigues-Jr, J.F., Tong, H., Traina, A., Faloutsos, C., Leskovec, J.: GMine: A System for Scalable, Interactive Graph Visualization and Mining. In: VLDB. ACM, Seul, South Corea (jun 2006)

[8] Scellato, S., Cardillo, A., Latora, V., Porta, S.: The backbone of a city. The European Physical Journal B - Condensed Matter and Complex Systems **50**(1-2), 221–225 (feb 2006)

[9] Teixeira, A.S., Monteiro, P.T., Carriço, J.A., Ramirez, M., Francisco, A.P.: Spanning edge betweenness. In: Workshop on Mining and Learning with Graphs. vol. 24, pp. 27–31 (2013)

[10] Teixeira, A.S., Santos, F.C., Francisco, A.P.: Spanning Edge Betweenness in Practice. In: Studies in Computational Intelligence, pp. 3–10. Springer Nature (2016)

[11] Tong, H., Faloutsos, C.: Center-piece subgraphs. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD'06. pp. 404–413. KDD '06, ACM Press (2006)

[12] Tong, H., Faloutsos, C., yu Pan, J.: Fast random walk with restart and its applications. In: Sixth International Conference on Data Mining (ICDM'06). pp. 613–622. IEEE (dec 2006)

APPENDIX B

SUPPLEMENTARY MATERIAL

Subsequently, we reproduce the supplementary material of the following papers:

1. **Spadon G.**, Carvalho A. C. P. L. F., Rodrigues-Jr J. F., and Alves L. G. A., *Reconstructing Commuters Network using Machine Learning and Urban Indicators*. Scientific Reports. Springer Nature, 2019 [Spadon *et al.* 2019].

2. **Spadon G.**, Hong S., Brandoli B., Matwin S., Rodrigues-Jr J. F., and Sun J., *Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning*. Transactions on Pattern Analysis and Machine Intelligence. IEEE, 2021 [Spadon *et al.* 2021].

Both documents are reproduced under the *Rights and Permissions* stated below:

Supplementary Information

Reconstructing commuters network using machine learning and urban indicators

Gabriel Spadon, Andre C. P. L. F. de Carvalho,
Jose F. Rodrigues-Jr, and Luiz G. A.Alves

List of Classifiers

#	Acronym	Name	Result
1	AdaBoost	Adaptive Boosting Classifier	✓
2	Bagging	Bagging Classifier	✓
3	BernoulliNB	Bernoulli Naive Bayes Classifier	✓
4	CalibratedCV	Calibrated Classifier with in-built cross-validation	✓
5	CatBoost	CatBoost Classifier	✓
6	ComplementNB	Complement Naive Bayes Classifier	✓
7	DecisionTree	Decision Tree Classifier	✓
8	ExtraTrees	Extremely Randomized Trees Classifier	✓
9	GaussianNB	Gaussian Naive Bayes Classifier	✓
10	GaussianProcess	Gaussian Processes Classifier	✗
11	GradientBoosting	Gradient Boosting Classifier	✓
12	HistGradientBoosting	Histogram-based Gradient Boosting Classification Tree	✓
13	KNeighbors	K-Nearest Neighbors Classifier	✓
14	LGBM	Light Gradient Boosting Machine Classifier	✓
15	LabelPropagation	Label Propagation Classifier	✗
16	LabelSpreading	Label Spreading Classifier	✗
17	LinearDA	Linear Discriminant Analysis Classifier	✓
18	LinearSVC	Linear Support Vector Classification	✓
19	Logistic	Logistic Regression Classifier	✓
20	LogisticCV	Logistic Regression Classifier with in-built cross-validation	✓
21	MLP	Multi-layer Perceptron Classifier	✓
22	MultinomialNB	Multinomial Naive Bayes Classifier	✓
23	NearestCentroid	Nearest Centroid Classifier	✓
24	NuSVC	Nu-Support Vector Classification	✗
25	PassiveAggressive	Passive Aggressive Classifier	✓
26	Perceptron	Perceptron Classifier	✓
27	QuadraticDA	Quadratic Discriminant Analysis Classifier	✓
28	RadiusNeighbors	Radius Neighbors Classifier	✗
29	RandomForest	Random Forest Classifier	✓

30	Ridge	Ridge Classifier	✓
31	RidgeCV	Ridge Classifier with in-built cross-validation	✓
32	SGD	Stochastic Gradient Descent Classifier	✗
33	SVC	Support Vector Classification	✗
34	XGBoost	Extreme Gradient Boosting Classifier	✓
Number of selected classifiers:			27/34

Table 1: List of classifiers tested during the unweighted link prediction of the commuters network. The *Acronym* column presents the short name of the algorithms, the *Name* column shows the full name of the algorithms, and the *Results* column marks with ✓ the classifiers that do not require early hyperparameter tuning and have passed for the subsequent testing phase, and with ✗ the discarded ones.

List of Regressors

#	Acronym	Name	Result
1	ARD	Automatic Relevance Determination Regression	✗
2	AdaBoost	Adaptive Boosting Regressor	✓
3	Bagging	Bagging Regressor	✓
4	BayesianRidge	Bayesian Ridge Regressor	✓
5	CCA	Canonical Correlation Analysis Regressor	✗
6	CatBoost	CatBoost Regressor	✓
7	DecisionTree	Decision Tree Regressor	✓
8	ElasticNet	Elastic-Net Regressor	✓
9	ElasticNetCV	Elastic-Net Regressor with in-built cross-validation	✓
10	ExtraTrees	Extremely Randomized Trees Regressor	✓
11	GaussianProcess	Gaussian Processes Regressor	✗
12	GradientBoosting	Gradient Boosting Regressor	✓
13	HistGradientBoosting	Histogram-based Gradient Boosting Regression Tree	✓
14	Huber	Huber Regressor	✓
15	Isotonic	Isotonic Regression	✗
16	KNeighbors	K-Nearest Neighbors Regressor	✓
17	KernelRidge	Kernel Ridge Regressor	✗
18	LGBM	Light Gradient Boosting Machine Regressor	✓
19	Lars	Lars Regressor	✗
20	LarsCV	Lars Regressor with in-built cross-validation	✗
21	Lasso	Lasso Regressor	✗
22	LassoCV	Lasso Regressor with in-built cross-validation	✓
23	LassoLars	Lasso-Lars Regressor	✗
24	LassoLarsCV	Lasso-Lars Regressor with in-built cross-validation	✓
25	LassoLarsIC	Lasso-Lars Regressor with information criterion	✓
26	Linear	Linear Regression	✗
27	LinearSVR	Linear Support Vector Regression	✗
28	MLP	Multi-layer Perceptron Regressor	✓
29	NuSVR	Nu-Support Vector Regression	✗

30	OrthogonalMP	Orthogonal Matching Pursuit Regrssor	✓
31	OrthogonalMPCV	Orthogonal Matching Pursuit Regrssor with in-built cross-validation	✗
32	PLSCanonical	Partial Least Squares Canonical Regressor	✗
33	PLS	Partial Least Squares Regressor	✓
34	PassiveAggressive	Passive Aggressive Regressor	✗
35	RANSAC	Random Sample Consensus Regressor	✗
36	RadiusNeighbors	Radius Neighbors Regressor	✗
37	RandomForest	Random Forest Regressor	✓
38	Ridge	Ridge Regressor	✓
39	RidgeCV	Ridge Regressor with in-built cross-validation	✓
40	SGD	Stochastic Gradient Descent Regressor	✗
41	SVR	Support Vector Regression	✗
42	TheilSen	Theil-Sen Regressor	✗
43	TransformedTarget	Transformed Target Regressor	✗
44	XGBoost	Extreme Gradient Boosting Regressor	✓
Number of selected regressors:			23/44

Table 2: List of regressors tested during the weighted link prediction of the commuters network. The *Acronym* column presents the short name of each algorithm, the *Name* column shows the full name of all algorithms, and the *Results* column marks with ✓ the regressors that do not require early hyperparameter tuning and have passed for the subsequent testing phase, and with ✗ the discarded ones.

Classifiers' Performance

#	Algorithm	Mean	Standard Deviation (σ)	Variance (σ^2)
1	CatBoost	0.87938	0.00270	0.00001
2	XGBoost	0.87523	0.00282	0.00001
3	LGBM	0.87391	0.00306	0.00001
4	HistGradientBoosting	0.87382	0.00298	0.00001
5	GradientBoosting	0.87288	0.00298	0.00001
6	AdaBoost	0.85866	0.00468	0.00002
7	Bagging	0.85620	0.00523	0.00003
8	RandomForest	0.83895	0.01334	0.00018
9	DecisionTree	0.81229	0.00924	0.00009
10	ExtraTrees	0.74033	0.02400	0.00058
11	LogisticCV	0.70626	0.02802	0.00079
12	Ridge	0.70278	0.01123	0.00013
13	LinearDA	0.69882	0.01014	0.00010
14	Logistic	0.67249	0.05846	0.00342
15	MLP	0.64219	0.02921	0.00085
16	QuadraticDA	0.63175	0.00419	0.00002
17	BernoulliNB	0.62663	0.00316	0.00001
18	KNeighbors	0.61093	0.00778	0.00006
19	LinearSVC	0.59470	0.06886	0.00474

20	CalibratedCV	0.58282	0.04873	0.00237
21	NearestCentroid	0.57061	0.00853	0.00007
22	GaussianNB	0.56790	0.01159	0.00013
23	ComplementNB	0.56444	0.04092	0.00167
24	MultinomialNB	0.56191	0.04357	0.00190
25	PassiveAggressive	0.52767	0.03754	0.00141
26	RidgeCV	0.51654	0.07597	0.00577
27	Perceptron	0.50213	0.02163	0.00047

Table 3: Statistics from the bootstrap sampling describing the *Mean*, *Standard Deviation*, and *Variance* of a thousand predictions made on different random samples of the training set. The experiment was carried out using the classifiers that have passed the first testing phase, see Table 1.

Regressors' Performance

#	Algorithm	Mean	Standard Deviation (σ)	Variance (σ^2)
1	XGBoost	0.65623	0.01124	0.00013
2	GradientBoosting	0.65430	0.01139	0.00013
3	LGBM	0.63696	0.01102	0.00012
4	HistGradientBoosting	0.63630	0.01102	0.00012
5	ExtraTrees	0.62010	0.01405	0.00002
6	RandomForest	0.60985	0.01727	0.00003
7	Bagging	0.60943	0.01675	0.00028
8	CatBoost	0.58423	0.01269	0.00016
9	AdaBoost	0.49586	0.04169	0.00174
10	MLP	0.49051	0.04793	0.00230
11	BayesianRidge	0.47401	0.00049	0.00002
12	ElasticNetCV	0.47311	0.00047	0.00002
13	LassoLarsCV	0.47300	0.00571	0.00003
14	LassoCV	0.47268	0.00529	0.00003
15	RidgeCV	0.47175	0.00604	0.00004
16	LassoLarsIC	0.47120	0.00629	0.00004
17	OrthogonalMP	0.47020	0.00511	0.00003
18	Ridge	0.46906	0.00639	0.00004
19	Huber	0.42919	0.01558	0.00024
20	PLS	0.36433	0.03680	0.00135
21	KNeighbors	0.34232	0.01488	0.00022
22	DecisionTree	0.31286	0.05175	0.00268
23	ElasticNet	0.20615	0.03905	0.00153

Table 4: Bootstrap sampling performance describing the *Mean*, *Standard Deviation*, and *Variance* of a thousand predictions made on different random samples of the training set. The experiment was carried out using the regressors that have passed the first testing phase, as described in Table 2.

List of Features and Selected Features

		Classifier		Regressor	
#	Features	City$_s$	City$_t$	City$_s$	City$_t$
1	Area	✓	✓	✓	✓
2	Child labor	✓	✗	✗	✓
3	Domestic violence	✗	✗	✗	✗
4	Elderly population	✗	✓	✓	✓
5	Female population	✓	✗	✓	✗
6	Gross Domestic Product	✗	✓	✗	✓
7	Homicides	✗	✗	✓	✗
8	Human Development Index	✓	✓	✗	✓
9	Illiteracy	✓	✓	✓	✗
10	Income	✗	✓	✗	✓
11	Male population	✗	✗	✗	✗
12	Minimum wage	✗	✓	✓	✓
13	Population	✗	✗	✗	✗
14	Population density	✓	✗	✓	✓
15	Sanitary sewage	✗	✓	✗	✓
16	Sanitation	✗	✗	✗	✗
17	Street arborization	✓	✓	✓	✓
18	Street urbanization	✓	✗	✓	✗
19	Suicides	✗	✗	✗	✓
20	Traffic accidents	✗	✓	✗	✓
21	Tuition rate	✓	✗	✗	✗
22	Unemployment	✓	✗	✓	✗
23	Distance	✓		✓	
Number of selected features:		21/45		23/45	

Table 5: List of features selected by the threshold-based feature selection process applied to both models inferred from the XGBoost classifier and regressor. In the table, we use City$_s$ and City$_t$ to denote the source and target cities, ✓ to indicate the remaining features after feature selection and ✗ to indicate the removed ones. Notice that the distance between two given cities is the same regardless of the city of departure. As a consequence, the distance represents a single feature, and there is no distinction between the distance from the source to the target city and vice-versa.

Supplementary Material
Pay Attention to Evolution: Time Series Forecasting with Deep Graph-Evolution Learning

Gabriel Spadon, Shenda Hong, Bruno Brandoli,
Stan Matwin, Jose F. Rodrigues-Jr, and Jimeng Sun

List of Tables

Notes. Table 1 list the acronym and full name of all algorithms we tested during the baselines' computation. Tables 2 to 6 present detailed information from the experiments discussed along with the main manuscript. The following tables regard the tests using Transfer Learning on the SARS-CoV-2 dataset, in which a new network was trained every 15 days starting on 45 days after the pandemic started and up to 120 days of its duration.

Extended Methods

Cosine Similarity. The cosine similarity, which has been widely applied in learning approaches, accounts for the similarity between two non-zero vectors based on their orientation in an inner product space [1]. The underlying idea is that the similarity is a function of the cosine angle θ between vectors $\mathbf{u} = [u_1, u_2, \ldots, u_N] \in \mathbb{R}^{N\times 1}$ and $\mathbf{v} = [v_1, v_2, \ldots, v_N] \in \mathbb{R}^{N\times 1}$. Hence, when $\theta = 1$, the two vectors in the inner product space have the same orientation, when $\theta = 0$, these vectors are oriented a 90° relative to each other, and when $\theta = -1$, the vectors are diametrically opposed. The cosine similarity between the vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as:

$$cos_\theta(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \circ \|\mathbf{v}\|} \tag{1}$$

where $\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{N} u_i v_i$ denotes the dot product between $\mathbf{u}$ and $\mathbf{v}$, and $\|\mathbf{u}\|$ represents the norm of the vector $\mathbf{u} = \sqrt{u \cdot u}$, while u_i is the i-th variable of u. In this work's scope, the cosine similarity is used to build similarity adjacency matrices, which measures per-nodes similarity in a variables' co-occurrence graph. The similarity between two nodes in the graph describes how likely those two variables co-occur in time. In this case, the similarity ends up acting as an intermediate activation function, enabling the graph evolution process by maintaining relationships' similarity between pairs of nodes. Thus, the cosine-matrix similarity is defined as:

$$cos_\theta(\mathbf{A}) = \frac{\mathbf{A} \cdot \mathbf{A}^{\mathrm{T}}}{\|\mathbf{A}\| \circ \|\mathbf{A}\|^{\mathrm{T}}} \tag{2}$$

where $\mathbf{A} \cdot \mathbf{A}^{\mathrm{T}}$ denotes the dot product between the adjacency matrix $\mathbf{A}$ and the transposed $\mathbf{A}^{\mathrm{T}}$, while $\|\mathbf{A}\|$ represents the norm of that same matrix with respect to any of its ranks. The resulting matrix of using the cosine-similarity activation is symmetric and referred to along with the main manuscript as *Evolution Weights*.

Horizon Forecasting. It stands for an approach used for making non-continuous predictions by accounting for a future gap in the data. It is useful in a range of applications by considering, for instance, that recent data is not available or too costly to be collected. Thereby, it is possible to optimize a model that disregards the near future and focuses on the far-away future. However, such an approach abdicates from additional information that could be learned from continuous timestamp predictions [2]. By not considering the near past as a variable that influences the near future, we might result in a non-stochastic view of time, meaning that the algorithm focuses on long-term dependencies rather than both long-and short-term dependencies. Along these lines, both the LSTNet [3] and DSANet [4] comply with horizon forecasting, and to make our results comparable, we set the horizon to one on both of them. Thus, we started assessing the test results right after the algorithms' last validation step because as closer to the horizon, the more accurate the horizon-based models should be.

Time-Series Segmentation. A simplistic yet effective approach to train time-series algorithms is through the Sliding Window technique [5], which is also referred to as Rolling Window (see Fig. 1). Such a technique

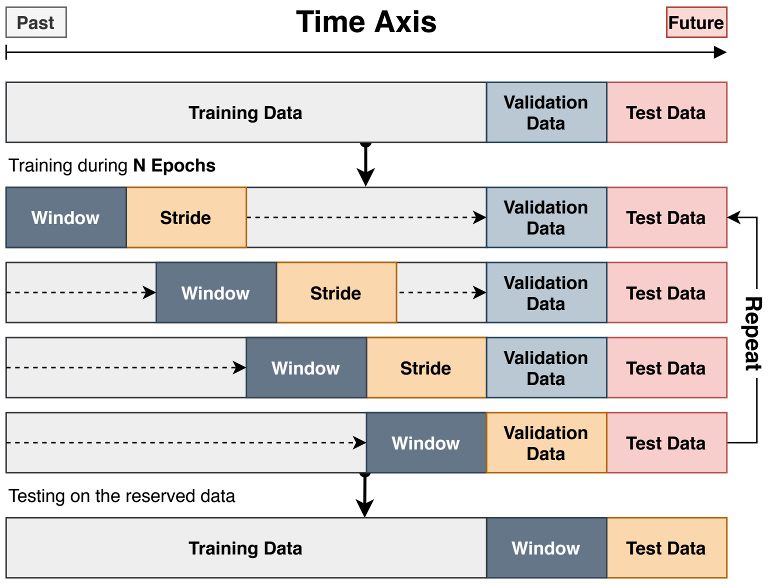


Figure 1: Sliding Window technique for training neural networks.

fixes a window size, which slides over the time axis, predicting a predefined number of future steps, referred to as stride. Some time-series studies have been using a variant technique known as Expanding Sliding Window [6, 7]. This variant starts with a prefixed window size, which grows as it slides, showing more information to the algorithm as time goes by. ReGENN holds for the traditional technique as it is bounded to the tensor weights dimension. Those dimensions are of a preset size and cannot be effortlessly changed during training, as it comes with increased uncertainty by continuously changing the number of internal parameters, such that a conventional neural network optimizer cannot handle it properly. Nevertheless, the window size of the Sliding Window is well known to be a highly sensitive hyperparameter [8,9]; to avoid an increased number of hyperparameter, we followed a non-tunable approach, in which we set the window size before the experiments taking into consideration the context of the datasets; such values were even across all window-based trials, including the baselines and ablation.

Optimization Strategy. ReGENN operates on a three-dimensional space shared between samples, time, and variables. In such a space, it carries out a time-based optimization strategy. The training process iterates over the time-axis of the dataset, showing to the network how the variables within a subset of time-series behave as time goes by, and later repeating the process through subsets of different samples. The network's weights are shared among the entire dataset and optimized towards best generalization simultaneously across samples, time, and variables. The dataset $\mathbf{T} \in \mathbb{R}^{s \times t \times v}$ is sliced into training $\widetilde{\mathbf{T}} \in \mathbb{R}^{s \times w \times v}$ and testing $\widehat{\mathbf{T}} \in \mathbb{R}^{s \times z \times v}$ as follows:

$$\widetilde{\mathbf{T}} = \begin{bmatrix} \mathbf{T}_{1,1,v} & \mathbf{T}_{1,2,v} & \mathbf{T}_{1,2,v} & \cdots & \mathbf{T}_{1,w,v} \\ \mathbf{T}_{2,1,v} & \mathbf{T}_{2,2,v} & \mathbf{T}_{2,2,v} & \cdots & \mathbf{T}_{2,w,v} \\ \mathbf{T}_{3,1,v} & \mathbf{T}_{3,2,v} & \mathbf{T}_{3,2,v} & \cdots & \mathbf{T}_{3,w,v} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}_{b,1,v} & \mathbf{T}_{b,2,v} & \mathbf{T}_{b,2,v} & \cdots & \mathbf{T}_{b,w,v} \end{bmatrix} \qquad \widehat{\mathbf{T}} = \begin{bmatrix} \mathbf{T}_{1,1+z,v} & \mathbf{T}_{1,2+z,v} & \mathbf{T}_{1,2+z,v} & \cdots & \mathbf{T}_{1,w+z,v} \\ \mathbf{T}_{2,1+z,v} & \mathbf{T}_{2,2+z,v} & \mathbf{T}_{2,2+z,v} & \cdots & \mathbf{T}_{2,w+z,v} \\ \mathbf{T}_{3,1+z,v} & \mathbf{T}_{3,2+z,v} & \mathbf{T}_{3,2+z,v} & \cdots & \mathbf{T}_{3,w+z,v} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}_{b,1+z,v} & \mathbf{T}_{b,2+z,v} & \mathbf{T}_{b,2+z,v} & \cdots & \mathbf{T}_{b,w+z,v} \end{bmatrix}$$

Once the data is sliced, we follow by using a gradient descent-based algorithm to optimize the model. In this work's scope, we used Adam [10] as the optimizer, as it is the most common optimizer among time-series forecasting problems. As the optimization criterion, we used the Mean Absolute Error (MAE), which is a generalization of the Support Vector Regression [11] with soft-margin criterion formalized as it follows:

$$\underset{\mathbf{w}}{\textbf{\textit{minimize}}} \; \left(\frac{1}{2}\,\|\mathbf{w}\|_{\mathrm{F}}^{2} + \mathcal{C}\right) \times \sum_{i=1}^{n}(\xi_i + \xi_i^*) \qquad \textbf{\textit{subject to}} \; \begin{aligned} y_i - (\mathbf{w}\cdot\mathbf{x}_i) - b &\le \rho + \xi_i, \\ (\mathbf{w}\cdot\mathbf{x}_i) + b - y_i &\le \rho + \xi_i^*, \\ \xi_i, \xi_i^* &\ge 0. \end{aligned}$$

where $\mathbf{w}$ is the set of optimizable parameters, $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, and both $\mathcal{C}$ and ρ are hyperparameters. The idea, then, is to find $\mathbf{w}$ that better fit $y_i, \mathbf{x}_i \forall i \in [1, n]$ so that all values are in $[\rho + \xi_i, \rho + \xi_i^*]$, where ξ_i and ξ_i^* are the two farther opposite points in the dataset. A similar formulation on the Linear SVR implementation for horizon forecasting was presented by *Lai et al.* [3]. Due to the higher-dimensionality among the multiple multivariate time-series used in this study, in which we consider the time to be continuous, the problem becomes:

$$\underset{\Omega}{\textbf{\textit{minimize}}} \; \left(\frac{1}{2}\,\|\Omega\|_{\mathrm{F}}^{2} + \mathcal{C}\right) \times \sum_{i=1}^{s}\sum_{j=1}^{w}\xi_i \qquad \textbf{\textit{subject to}} \; \left|\widehat{\mathbf{Y}}_{i,j} - \widehat{\mathbf{T}}_{i,j}\right| \le \rho + \xi_{i,j},\ \xi_{i,j} \ge 0.$$

where Ω is the set of internal parameters of ReGENN, $\widehat{\mathbf{Y}}$ is the network's output and $\widehat{\mathbf{T}}$ the ground truth. When disregarding $\mathcal{C}$ and setting ρ as zero, we can reduce the problem to the MAE loss formulation:

$$\underset{\Omega}{\textbf{\textit{minimize}}} \; \sum_{i=1}^{s}\sum_{j=1}^{w}\left|\widehat{\mathbf{Y}}_{i,j} - \widehat{\mathbf{T}}_{i,j}\right|$$

Square-and logarithm-based criteria can also be used with ReGENN. We avoid doing so, as this is a decision that should be made based on each dataset. Contrarily, we follow the SVR path towards the evaluation of absolute values, which is less sensitive to outliers and enables ReGENN to be applied to a range of applications.

Transfer-Learning Approach. We adopted a Transfer Learning approach to train the network on the SARS-CoV-2 dataset that, although different, resembles Online Deep Learning [12]. The idea is to train the network on incremental slices of the time-axis, such that the pre-trained weights of a previous slice are used to initialize the weights of the network in the next slice (see Fig. 2). This technique aims not only to achieve better performance towards the network but also to show that ReGENN is useful throughout the pandemic.

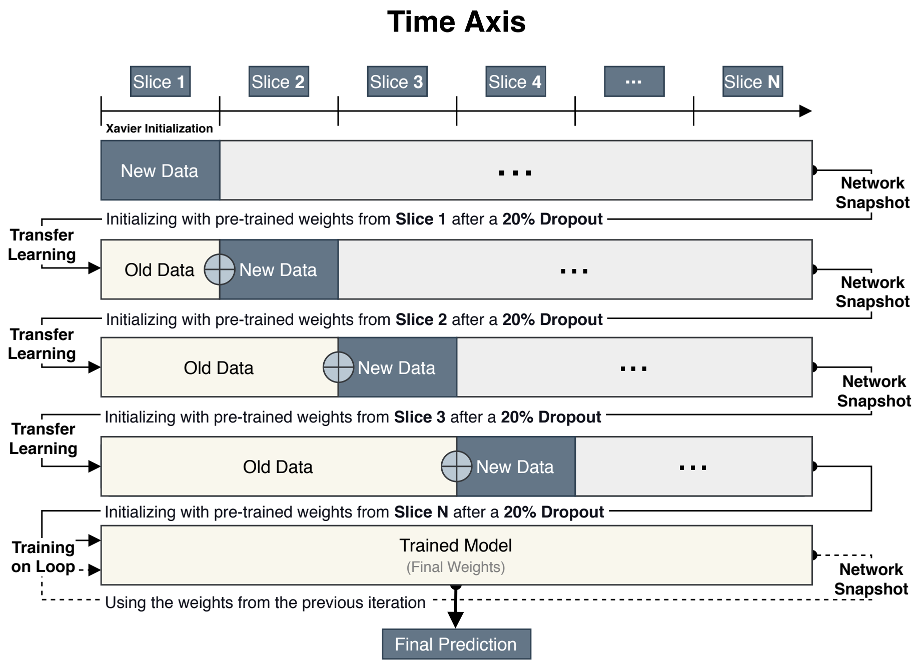

Time Axis
Slice 1
Slice 2
Slice 3
Slice 4
...
Slice N
Xavier Initialization
New Data
...
Network Snapshot
Initializing with pre-trained weights from Slice 1 after a 20% Dropout
Transfer Learning
Old Data
New Data
Initializing with pre-trained weights from Slice 2 after a 20% Dropout
Initializing with pre-trained weights from Slice 3 after a 20% Dropout
Initializing with pre-trained weights from Slice N after a 20% Dropout
Training on Loop
Trained Model
(Final Weights)
Using the weights from the previous iteration
Final Prediction

Figure 2: Transfer Learning used for streaming time-series.

Hyperparameters adjustment is usually required when transferring the weights from one network to another, mainly the learning rate; for the list of hyperparameters we used, see Tab. 3. Besides, we deliberately applied a 20% Dropout on all tensor weights outside the network architecture and before starting the training. The aim behind that decision was to insert randomness in the pipeline and avoid local optima. It worth mentioning that we did not observe any decrease in performance, but the optimizer's convergence was slower in some cases.

Baselines Algorithms. Open-source Python libraries provided the time series and machine learning algorithms used along with the experiments. Time series algorithms came from the statsmodels[1], while the machine learning ones majorly from the Scikit-Learn[2]. Further algorithms, such as XGBoost[3], LGBM[4], and CatBoost[5],

[1]Available at `https://www.statsmodels.org/stable/index.html`.
[2]Available at `https://scikit-learn.org/stable/`.
[3]Available at `https://xgboost.readthedocs.io/en/latest/`.
[4]Available at `https://lightgbm.readthedocs.io/en/latest/`.
[5]Available at `https://catboost.ai/`.

have a proprietary open-source implementation, which was preferred over the others. We used the default hyper-parameters over all the experiments, performing no fine-tuning. However, because all the datasets we tested are strictly positive, we forced all the negative output to become zero, such as made by a ReLU activation function.

A list with names and algorithms tested along with the experiments is provided in Tab 1, which contains more algorithms than we reported in the main paper. That because we are listing all algorithms, even those removed from the pipeline due to being incapable of working with the input data and yielding exceptions.

Table 1: List of algorithms tested during the baselines' computation. The **Acronym** column presents each algorithm's short name, and the **Name** column shows the full name of all algorithms (when applicable).

#	**Acronym**	**Name**
0	AdaBoost	Adaptive Boosting
1	ARD	Automatic Relevance Determination
2	ARIMA	Autoregressive Integrated Moving Average
3	ARMA	Autoregressive Moving Average
4	Autoregressive	—
5	Bagging	—
6	BayesianRidge	—
7	CatBoost	—
8	CCA	Canonical Correlation Analysis
9	Decision Tree	—
10	DSANet	Dual Self-Attention Network
11	Elman RNN	Elman's Recurrent Neural Network
12	Exponential Smoothing	Single Exponential Smoothing
13	Extra Tree	Extremely Randomized Tree
14	Extra Trees	Extremely Randomized Trees
15	Gaussian Process	—
16	Gradient Boosting	—
17	GRU	Gated Recurrent Unit
18	Histogram Grad. Boosting	Histogram Gradient Boosting
19	Historical Average	Dummy Regressor
20	Huber	—
21	Isotonic	—
22	Kernel Ridge	—
23	KNeighbors	k-Nearest Neighbors
24	Lars	Least Angle Regression
25	Lasso-Lars	Least Absolute Shrinkage and Selection Operator w/ Lars
26	Lasso-Lars-IC	Lasso-Lars w/ Information Criterion
27	LGBM	Light Gradient Boosting Machine
28	Linear Regression	—
29	Linear SVR	Linear Support Vector Regression
30	LSTM	Long Short Term Memory
31	LSTNet	Long-and Short Term time-series Network
32	MLCNN	Multi-Level Construal Neural Network
33	Moving Average	—
34	Multi-Task Elastic-Net	—
35	Multi-Task Lasso	—

36	NuSVR	Nu-Support Vector Regression
37	Orthogonal Matching Pursuit	—
38	Passive Aggressive	—
39	PLS Canonical	Partial Least Squares Canonical
40	PLS	Partial Least Squares
41	Radius Neighbors	—
42	Random Forest	—
43	RANSAC	Random Sample Consensus Regressor
44	ReGENN	Recurrent Graph Evolution Neural Network
45	Ridge	—
46	SARIMA	Seasonal Autoregressive Integrated Moving Average
47	SGD	Stochastic Gradient Descent
48	SVR	Support Vector Regression
49	Theil-Sen	—
50	Transformed Target	—
51	Vector Autoregression	—
52	XGBoost	Extreme Gradient Boosting

References

[1] P.-N. Tan, M. Steinbach, and V. Kumar, "Data Mining Introduction," 2006.

[2] J. Cheng, K. Huang, and Z. Zheng, "Towards better forecasting by fusing near and distant future visions," *arXiv*, 2019.

[3] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* New York, NY, USA: ACM, jun 2018, pp. 95–104.

[4] S. Huang, X. Wu, D. Wang, and A. Tang, "DSANet: Dual self-attention network for multivariate time series forecasting," in *International Conference on Information and Knowledge Management, Proceedings*, ser. CIKM'19. New York, NY, USA: ACM, nov 2019, pp. 2129–2132.

[5] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: a Survey and Novel Approach," in *Data Mining in Time Series Databases*, ser. Series in Machine Perception and Artificial Intelligence. World Scientific, jun 2004, vol. Volume 57, pp. 1–21.

[6] E. Egrioglu, E. Bas, U. Yolcu, and M. Y. Chen, "Picture fuzzy time series: Defining, modeling and creating a new forecasting method," *Engineering Applications of Artificial Intelligence*, vol. 88, feb 2020.

[7] A. Babii, R. T. Ball, E. Ghysels, and J. Striaukas, "Machine learning panel data regressions with an application to nowcasting price earnings ratios," *arXiv*, 2020.

[8] R. Frank, N. Davey, and S. Hunt, "Input window size and neural network predictors," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 2. IEEE, 2000, pp. 237–242.

[9] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 31, no. 1-3, pp. 91–103, 2001.

[10] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[11] V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 281–287.

[12] D. Sahoo, Q. Pham, J. Lu, and S. C. H. Hoi, "Online Deep Learning: Learning Deep Neural Networks on the Fly," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence.* California: International Joint Conferences on Artificial Intelligence Organization, jul 2018, pp. 2660–2666.

Hyperparameters for the Main Experiments

Table 2: List of hyperparameters used during the main experiments.

	——————| ReGENN |——————		
	SARS-CoV-2	**Brazilian Weather**	**PhysioNet**
autoregression	True	True	True
batch-size	32	64	256
bias	True	True	True
bidirectional-gate	False	False	False
bidirectional-sequencer	False	False	False
clip-norm	85.0	10.0	25.0
criterion	MAE	MAE	MAE
dropout	0.0	0.0	0.0
early-stop	250	100	10
epochs	2500	1000	100
evolution-function	Identity	Identity	Identity
gate	LSTM	LSTM	LSTM
iterator	Time	Time	Time
learning-rate	0.001	0.0001	0.001
load-weights	True	False	False
no-encoder	False	False	False
no-sequencer	False	False	False
normalization-axis	2	2	2
normalization-type	Maximum	Maximum	Maximum
optimizer	Adam	Adam	Adam
output-function	ReLU	ReLU	ReLU
random-seed	0	0	0
scheduler-factor	0.95	0.95	0.2
scheduler-min-lr	0.0	0.0	0.0
scheduler-patience	25	50	40
scheduler-threshold	0.1	0.1	0.1
sequencer	LSTM	LSTM	LSTM
stride	14	56	6
seed	0	0	0
validation-stride	7	28	6
watch-axis	2	2	2
window	7	84	12
	——————| MLCNN |——————		
	SARS-CoV-2	**Brazilian Weather**	**PhysioNet**
batch-size	32	64	256
clip-norm	10	10	10
collaborate-span	2	2	2
collaborate-stride	1	1	1
criterion	MAE	MAE	MAE
dropout	0.2	0.2	0.2
epochs	2500	1000	100
hidden-CNN	100	100	100
hidden-RNN	100	100	100
highway-window	1	1	1
input-size	7	84	12
kernel-size	5	5	5
learning-rate	0.001	0.001	0.001
mode	Continuous	Continuous	Continuous
num-CNN	10	10	10
normalization	1	1	1
optimizer	Adam	Adam	Adam
output-function	ReLU	ReLU	ReLU
output-size	14	56	6
seed	0	0	0

Table 2 continued from previous page

	—————| **DSANet** |—————		
	SARS-CoV-2	**Brazilian Weather**	**PhysioNet**
batch-size	32	64	256
clip-norm	10	10	10
criterion	MAE	MAE	MAE
dim-inner	2048	2048	2048
dim-k	64	64	64
dim-model	512	512	512
dim-v	64	64	64
dropout	0.1	0.1	0.1
early-stop	250	100	10
epochs	2500	1000	100
horizon	1	1	1
learning-rate	0.001	0.001	0.001
local	3	3	3
num-heads	8	8	8
num-kernels	32	32	32
num-layers	6	6	6
normalization	2	2	2
optimizer	Adam	Adam	Adam
output-function	ReLU	ReLU	ReLU
seed	0	0	0
w-kernel	1	1	1
window	7	84	12
	—————| **LSTNet** |—————		
	SARS-CoV-2	**Brazilian Weather**	**PhysioNet**
batch-size	32	64	256
clip-norm	10	10	10
CNN-kernel	6	6	6
criterion	MAE	MAE	MAE
dropout	0.2	0.2	0.2
early-stop	250	100	10
epochs	2500	1000	100
hidden-CNN	100	100	100
hidden-RNN	100	100	100
hidden-Skip	7	84	12
highway-window	7	84	12
horizon	1	1	1
learning-rate	0.001	0.001	0.001
normalization	2	2	2
optimizer	Adam	Adam	Adam
output-function	ReLU	ReLU	ReLU
seed	0	0	0
skip-steps	2	2	2
window	7	84	12

Hyperparameters for the Ablation Tests

Table 3: List of hyperparameters used during the ablation experiments.

	ReGENN	**PyTorch' Deafult**	**Literature's Default**
clip-norm	—	0	10
dropout	—	0	0.1
learning-rate	—	0.001	0.001
scheduler-factor	—	0.1	0.95
scheduler-patience	—	10	25
scheduler-threshold	—	0.001	0.1

Note. The hyperparameters from above are shared across all the datasets; the other ones follow as in Tab. 2.

Detailed Result on the Main Experiments

Table 4: Detailed results for the main experiments.

Legend: Algorithms with best performance are in **bold**, the ones noted as — yielded exceptions, and others as *** were suppressed due to poor performance.

	SARS-CoV-2						Brazilian Weather						PhysioNet					
	MAE	± STD	RMSE	± STD	MSLE	± STD	MAE	± STD	RMSE	± STD	MSLE	± STD	MAE	± STD	RMSE	± STD	MSLE	± STD
Deep Learning Algorithms																		
ReGENN	**165.41**	**30.37**	**915.92**	**294.06**	**0.05**	**0.02**	**1.92**	**0.02**	**4.86**	**0.13**	**0.28**	**0.01**	**18.22**	**1.04**	**47.31**	**6.27**	**1.37**	**0.12**
MLCNN	5298.30	22152.36	5779.21	23677.93	16.27	32.19	3.46	3.32	5.74	4.25	0.70	1.52	20.81	13.49	43.23	28.42	3.32	3.00
DSANet	18428.63	81222.22	20131.28	91468.66	52.90	36.86	18.20	2.42	21.65	2.67	7.73	0.79	50.31	17.32	74.96	28.57	10.82	3.39
LSTNet	10749.65	64733.89	13172.62	79729.95	15.55	23.33	4.16	5.48	6.34	6.21	1.17	2.47	28.25	18.15	50.70	31.89	4.35	4.04
Multi-Output and Multi-Task Algorithms																		
CCA	5,566.74	618.65	48,941.52	3,308.99	0.50	0.22	3.60	0.86	5.33	1.85	0.49	0.71	21.38	20.70	39.26	34.60	2.00	1.27
Decision Tree	1,098.77	163.13	6,055.55	2,122.73	0.27	0.05	3.47	0.98	6.05	2.96	0.47	0.65	25.25	22.65	48.81	41.02	3.19	1.49
Extra Tree	851.65	12.55	3,316.89	276.68	0.25	0.05	3.62	0.74	6.30	2.42	0.44	0.58	25.04	22.69	48.29	40.68	3.15	1.54
Extra Trees	771.33	120.86	3,823.39	5.44	0.14	0.08	**2.82**	**1.02**	**4.42**	**2.00**	**0.37**	**0.53**	**20.35**	**18.11**	**36.36**	**29.24**	**2.01**	**1.21**
Gaussian Process	2,730.59	309.13	14,295.71	3,034.59	0.57	0.47	15.51	8.24	17.51	6.55	4.69	2.77	132.00	122.99	681.47	708.10	5.00	2.92
Historical Average	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09	3.20	1.10	4.84	1.86	0.32	0.42	29.58	16.93	42.34	28.20	2.24	1.33
Kernel Ridge	797.81	78.97	2,917.93	162.53	0.11	0.04	3.20	1.12	4.91	2.13	0.43	0.61	—	—	—	—	—	—
KNeighbors	967.37	36.99	5,414.50	1,014.23	0.13	0.07	3.09	1.02	4.84	2.12	0.46	0.69	21.33	20.01	39.04	32.76	2.19	1.28
Lars	***	***	***	***	***	***	***	***	***	***	***	***	20.48	18.94	36.77	29.78	2.03	1.31
Lasso-Lars	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09	3.20	1.10	4.84	1.86	0.32	0.42	29.58	16.93	42.34	28.20	2.24	1.33
Linear Regression	***	***	***	***	***	***	3.59	1.14	5.42	2.19	0.51	0.74	20.47	18.94	36.77	29.78	2.03	1.31
Multi-Task Elastic-Net	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09	3.20	1.10	4.84	1.86	0.32	0.42	29.25	17.04	41.98	28.35	2.24	1.33
Multi-Task Lasso	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09	3.20	1.10	4.84	1.86	0.32	0.42	29.58	16.93	42.34	28.20	2.24	1.33
Orthogonal Matching Pursuit	2,142.70	897.09	19,881.75	10,964.98	0.17	0.09	5.79	1.99	9.86	5.90	0.83	1.12	20.89	18.75	37.23	29.50	2.03	1.31
PLS Canonical	8,743.59	2,964.01	84,481.55	58,065.34	0.39	0.11	6.01	1.07	8.33	1.47	0.76	0.82	26.31	20.30	46.43	34.86	2.41	1.22
PLS	1,341.96	155.97	6,461.25	322.42	0.50	0.01	2.89	0.97	4.51	1.88	0.38	0.55	21.04	18.71	36.75	29.63	2.07	1.32
Radius Neighbors	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Random Forest	**736.11**	**53.37**	**3,623.38**	**690.48**	**0.14**	**0.07**	2.89	1.02	4.53	2.00	0.38	0.55	20.39	18.08	36.22	29.51	1.99	1.21
RANSAC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Ridge	804.49	86.67	2,957.27	191.77	0.14	0.08	3.13	1.12	4.86	2.15	0.45	0.66	20.47	18.94	36.77	29.78	2.03	1.31
Single-Target Algorithms on Chain Ensemble																		
AdaBoost	1,099.84	337.60	7,184.23	5,335.13	0.15	0.09	2.82	0.97	4.51	1.96	0.34	0.47	—	—	—	—	—	—
ARD	889.47	30.26	3,368.50	207.97	0.16	0.08	4.35	1.42	6.65	2.97	0.63	0.91	—	—	—	—	—	—
Bagging	862.94	115.31	4,244.29	116.49	0.17	0.06	3.05	1.12	4.73	2.29	0.44	0.66	21.28	18.96	37.78	31.30	2.02	1.24
Bayesian Ridge	828.14	12.06	3,031.29	145.24	0.16	0.06	3.14	1.06	4.88	1.96	0.40	0.57	20.47	18.93	36.74	29.74	2.03	1.31
CatBoost	1,115.88	85.29	4,129.19	139.31	0.14	0.08	3.40	0.73	4.97	1.57	0.35	0.50	19.95	18.03	35.94	29.16	1.98	1.19
Gradient Boosting	**800.29**	**1.54**	**4,722.74**	**1,271.82**	**0.16**	**0.06**	3.05	1.13	4.84	2.44	0.45	0.66	—	—	—	—	—	—
Histogram Grad. Boosting	866.30	38.80	4,612.11	44.68	0.16	0.08	3.09	1.15	4.75	2.28	0.47	0.70	20.90	18.49	37.57	30.70	2.00	1.19
Huber	2,032.72	41.78	17,770.19	610.28	0.37	0.08	3.43	1.09	5.21	2.13	0.48	0.70	19.87	18.04	38.10	28.49	2.14	1.19
Isotonic	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Lasso-Lars-IC	810.03	30.97	3,929.81	129.59	0.14	0.08	2.91	1.07	4.59	1.94	0.31	0.42	20.46	18.94	36.72	29.75	2.03	1.32
LGBM	1,110.17	198.23	7,650.02	3,822.07	0.28	0.01	—	—	—	—	—	—	20.12	18.14	36.30	29.41	1.98	1.19
Linear SVR	875.51	97.79	3,540.90	247.08	0.16	0.01	3.44	1.19	5.21	2.30	0.50	0.73	**19.72**	**17.71**	**38.65**	**29.06**	**2.18**	**1.08**
NuSVR	809.97	19.17	2,973.63	21.61	0.12	0.06	**2.79**	**0.97**	**4.40**	**1.96**	**0.34**	**0.49**	—	—	—	—	—	—
Passive Aggressive	1,730.12	113.18	8,310.46	432.48	0.18	0.12	3.22	0.89	4.82	1.98	0.46	0.69	31.27	31.35	55.82	58.27	2.76	1.84
SGD	815.97	11.37	3,106.47	40.02	0.13	0.06	***	***	***	***	***	***	20.53	19.00	36.78	29.88	2.04	1.32
SVR	1,344.80	244.56	5,511.70	1,539.93	0.12	0.06	3.38	1.20	4.63	2.05	0.56	0.86	—	—	—	—	—	—
Theil-Sen	91,149.14	59,396.23	1,292,430.04	858,201.91	6.45	2.47	3.65	1.22	5.44	2.21	0.51	0.73	20.61	19.56	37.32	31.23	2.05	1.31
Transformed Target	68,530.56	55,751.11	868,586.61	989,030.54	16.39	1.00	3.58	1.13	5.42	2.19	0.51	0.74	20.47	18.94	36.77	29.78	2.03	1.31
XGBoost	806.03	37.02	5,033.10	1,599.32	0.16	0.06	3.04	1.12	4.81	2.41	0.44	0.66	19.94	18.05	35.92	29.08	1.98	1.19
Time Series Algorithms																		
Autoregressive	11,529.79	6,752.42	76,118.62	48,374.82	9.15	6.14	2.69	1.44	4.68	1.73	0.37	0.54	24.44	9.68	53.92	16.37	2.08	0.95
ARIMA	11,578.80	6,798.84	76,044.64	48,478.91	9.22	5.23	2.68	1.49	4.81	1.78	0.34	0.49	***	***	***	***	***	***
ARMA	12,578.49	6,091.86	79,263.71	45,555.46	16.71	4.79	3.21	0.82	6.22	1.71	0.58	0.47	182.01	189.23	20,124.41	29,967.38	2.61	0.87
Moving Average	7,757.33	2,887.33	73,013.22	42,555.86	2.54	1.17	2.70	1.48	4.49	1.75	0.47	0.73	—	—	—	—	—	—
SARIMA	***	***	***	***	***	***	2.68	1.50	4.82	1.78	0.34	0.49	25.05	9.48	55.89	16.04	2.18	0.93
Exponential Smoothing	**667.33**	**940.85**	**2,607.29**	**3,634.47**	**0.27**	**0.17**	**2.66**	**1.47**	**4.78**	**1.78**	**0.34**	**0.48**	**23.77**	**9.48**	**52.89**	**15.52**	**2.09**	**0.92**
Vector Autoregression	6,395.14	2,397.58	75,292.43	39,167.56	3.21	0.05	2.70	1.51	4.46	1.76	0.49	0.76	***	***	***	***	***	***

Table 5: Ablation on the main experiments with PyTorch's hyperparameters.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	SARS-CoV-2						**Brazilian Weather**						**PhysioNet**					
	MAE	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**
	Recurrent Neural Network (RNN)																	
${}_{\mathrm{B}}\mathrm{RU}$	434.18	250.54	2678.49	2046.82	0.09	0.05	2.03	0.03	4.91	0.10	0.28	0.01	19.94	1.18	50.34	7.73	1.39	0.13
$\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}$	3680.89	2404.90	25773.58	22768.94	7.47	0.80	2.84	0.09	6.49	0.16	0.55	0.02	19.01	0.97	48.70	6.16	1.42	0.13
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1533.62	949.53	18013.32	16584.35	3.82	0.16	2.44	0.06	6.19	0.10	0.51	0.02	**18.68**	**1.06**	**48.02**	**5.70**	**1.38**	**0.13**
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	**245.06**	**65.03**	**1359.34**	**462.61**	**0.09**	**0.05**	2.71	0.05	6.42	0.13	0.54	0.02	18.81	1.17	48.92	7.67	1.39	0.12
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1386.30	859.67	16848.20	15758.33	3.72	0.45	4.56	0.08	9.81	0.11	1.48	0.03	18.86	1.13	49.54	7.22	1.40	0.13
$\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}$	3171.99	1445.00	21117.23	16502.10	3.92	0.10	2.37	0.11	5.35	0.22	0.33	0.02	19.25	0.97	49.18	6.65	1.45	0.12
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	350.75	202.78	1985.39	1228.78	0.11	0.03	2.22	0.13	5.61	0.53	0.41	0.03	18.90	1.26	48.77	7.38	1.41	0.15
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1386.73	641.20	17459.45	14421.01	3.70	0.30	**1.97**	**0.10**	**4.84**	**0.38**	**0.27**	**0.02**	19.07	1.34	48.89	7.24	1.41	0.14
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1375.60	988.31	15987.09	16378.54	3.69	0.34	5.07	0.04	10.51	0.04	1.73	0.01	19.04	1.07	50.05	7.24	1.45	0.13
${}_{\mathrm{U}}\mathrm{RU}$	1277.24	1053.33	7948.96	7430.65	2.89	0.96	2.13	0.04	5.13	0.05	0.32	0.01	20.52	1.16	49.72	7.29	1.43	0.12
	Gated Recurrent Unit (GRU)																	
${}_{\mathrm{B}}\mathrm{RU}$	3167.33	1458.27	28834.18	23983.55	7.80	0.49	2.00	0.09	4.89	0.34	0.27	0.03	24.88	1.04	56.74	6.09	3.01	0.12
$\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}$	950.11	433.35	5133.01	2301.34	0.19	0.14	2.35	0.08	5.34	0.21	0.33	0.02	18.93	1.08	49.30	7.19	1.40	0.12
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	311.09	119.17	1873.65	1082.02	0.10	0.07	2.29	0.05	5.68	0.09	0.41	0.01	18.58	1.09	47.94	6.01	1.38	0.13
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	650.06	185.77	3443.73	1270.04	0.10	0.04	2.23	0.05	5.61	0.11	0.41	0.01	18.60	1.07	48.55	5.86	1.38	0.14
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1377.91	517.07	17145.02	14067.11	3.66	0.10	4.85	0.12	10.14	0.25	1.60	0.03	**18.56**	**1.05**	**48.54**	**6.96**	**1.39**	**0.11**
$\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}$	1360.35	737.84	7611.46	4036.40	0.27	0.11	2.20	0.06	5.11	0.22	0.28	0.02	19.40	1.05	50.73	8.38	1.43	0.13
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	**240.35**	**57.07**	**1335.96**	**343.59**	**0.08**	**0.04**	**1.98**	**0.11**	**4.87**	**0.51**	**0.28**	**0.03**	18.70	1.06	48.74	6.14	1.39	0.10
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	3377.44	1741.87	25335.28	22336.83	7.44	1.07	2.56	0.15	6.30	0.21	0.54	0.02	18.79	0.97	49.56	8.40	1.37	0.12
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1592.62	895.15	18207.40	15612.58	3.77	0.39	4.95	0.07	10.32	0.09	1.62	0.02	18.88	0.98	49.72	8.45	1.40	0.11
${}_{\mathrm{U}}\mathrm{RU}$	2319.57	1793.72	21536.86	21418.12	6.26	0.55	3.05	0.10	7.24	0.17	0.73	0.02	30.06	0.89	62.71	5.67	4.54	0.09
	Long Short-Term Memory (LSTM)																	
${}_{\mathrm{B}}\mathrm{RU}$	3000.14	1954.01	25989.65	24313.81	7.65	0.78	**2.31**	**0.04**	**5.68**	**0.04**	**0.41**	**0.01**	19.57	1.23	50.33	7.28	1.40	0.11
$\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}$	619.31	356.13	3094.25	1878.42	0.17	0.11	2.32	0.08	5.20	0.27	0.29	0.03	19.08	1.15	49.08	7.15	1.41	0.14
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	**216.01**	**89.11**	**1146.94**	**465.90**	**0.10**	**0.07**	2.52	0.05	6.28	0.18	0.54	0.01	18.66	0.97	49.47	8.82	1.37	0.12
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	2863.94	1345.36	27905.87	22477.61	7.69	0.62	3.68	0.10	8.52	0.21	1.07	0.02	**18.55**	**0.99**	**49.12**	**6.05**	**1.37**	**0.14**
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	4060.62	2655.67	31538.83	28960.85	11.35	0.99	3.73	0.07	8.53	0.15	1.08	0.02	18.64	1.09	48.76	5.96	1.39	0.12
$\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}$	3199.26	2436.81	12581.64	11769.69	0.26	0.07	2.47	0.06	5.10	0.21	0.28	0.02	19.78	1.15	50.21	7.31	1.42	0.13
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1351.42	644.34	17065.24	14155.37	3.67	0.30	3.38	0.05	7.89	0.06	0.93	0.03	18.69	1.28	49.27	8.58	1.38	0.12
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1431.78	946.29	17235.49	16431.19	3.79	0.38	5.16	0.08	10.61	0.08	1.74	0.02	18.70	1.13	49.85	6.90	1.38	0.11
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	222.06	85.96	1265.74	642.42	0.08	0.03	7.48	0.13	13.34	0.14	2.81	0.04	18.86	1.11	49.88	7.80	1.41	0.12
${}_{\mathrm{U}}\mathrm{RU}$	4943.62	2892.09	28696.47	24921.66	7.59	0.30	4.80	0.02	9.74	0.08	1.46	0.02	20.80	0.95	50.13	7.28	1.37	0.12

Table 6: Ablation on the main experiments with hyperparameters commonly used in the related literature.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	SARS-CoV-2						**Brazilian Weather**						**PhysioNet**					
	MAE	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**
							Recurrent Neural Network (RNN)											
$_B$RU	1,895.13	995.98	7,184.26	5,885.75	0.08	0.03	3.34	0.11	5.49	0.42	0.29	0.02	20.67	0.93	48.09	6.89	1.37	0.11
E → $_B$RU	2,510.11	1,324.54	9,521.55	6,720.47	0.19	0.05	4.12	0.21	6.40	0.29	0.40	0.06	19.83	1.04	48.97	6.18	1.44	0.14
(E → $_B$RU + $_B$RU) + AR	317.13	184.28	1,806.87	1,269.50	0.10	0.07	2.16	0.11	5.01	0.20	0.28	0.02	18.84	1.14	49.50	6.27	1.37	0.12
(E → $_B$RU + $_U$RU) + AR	279.38	115.14	1,569.99	836.28	0.09	0.07	**2.07**	**0.07**	**4.95**	**0.29**	**0.28**	**0.02**	**18.77**	**1.07**	**48.89**	**6.17**	**1.38**	**0.14**
(E → $_B$RU) + AR	304.52	141.03	2,029.00	1,595.46	0.10	0.04	3.53	0.12	8.05	0.29	0.94	0.04	18.88	1.04	49.21	6.81	1.40	0.13
E → $_U$RU	2,692.19	2,133.65	8,934.01	8,459.59	0.29	0.18	3.94	0.21	6.26	0.45	0.35	0.08	20.04	1.09	47.82	7.02	1.45	0.13
(E → $_U$RU + $_B$RU) + AR	280.09	142.50	1,669.51	1,096.35	0.10	0.03	2.79	0.11	6.46	0.05	0.55	0.03	18.96	1.00	48.06	6.35	1.42	0.09
(E → $_U$RU + $_U$RU) + AR	1,423.25	738.45	17,459.46	14,825.39	3.72	0.39	2.16	0.15	4.98	0.45	0.28	0.04	18.99	0.94	49.17	6.33	1.42	0.11
(E → $_U$RU) + AR	**240.34**	**93.62**	**1,386.81**	**832.15**	**0.10**	**0.06**	4.60	0.05	9.78	0.14	1.47	0.02	19.11	1.18	50.14	6.98	1.46	0.11
$_U$RU	1,954.90	1,154.82	7,310.14	6,297.24	0.21	0.13	3.31	0.08	5.53	0.29	0.29	0.02	20.93	1.15	48.50	6.71	1.42	0.12
							Gated Recurrent Unit (GRU)											
$_B$RU	3,094.99	2,293.39	18,854.48	19,286.26	3.89	0.31	3.56	0.05	5.93	0.13	0.34	0.01	30.27	0.89	60.74	5.45	4.53	0.12
E → $_B$RU	2,702.04	2,483.15	10,715.26	13,157.31	0.18	0.10	3.93	0.10	6.38	0.16	0.40	0.02	19.86	1.11	48.37	7.14	1.41	0.12
(E → $_B$RU + $_B$RU) + AR	219.16	106.06	1,276.78	881.66	0.07	0.03	2.32	0.07	5.69	0.22	0.41	0.02	18.72	0.98	48.11	6.47	1.40	0.11
(E → $_B$RU + $_U$RU) + AR	259.91	149.24	1,597.42	1,273.26	0.11	0.07	**2.08**	**0.18**	**4.94**	**0.41**	**0.28**	**0.04**	**18.56**	**0.99**	**48.32**	**6.78**	**1.37**	**0.13**
(E → $_B$RU) + AR	505.81	582.37	3,916.95	5,297.45	0.11	0.08	3.53	0.13	8.06	0.12	0.94	0.03	18.72	0.97	48.54	6.37	1.37	0.10
E → $_U$RU	2,380.63	1,475.81	8,555.18	6,771.74	0.23	0.12	2.92	0.06	5.44	0.12	0.36	0.01	20.29	1.09	50.32	7.57	1.49	0.15
(E → $_U$RU + $_B$RU) + AR	245.62	159.01	1,367.82	966.48	0.07	0.03	2.93	0.07	6.93	0.17	0.67	0.02	18.68	1.09	48.85	6.29	1.37	0.14
(E → $_U$RU + $_U$RU) + AR	**216.81**	**107.29**	**1,280.58**	**846.01**	**0.11**	**0.04**	3.17	0.08	7.46	0.09	0.81	0.02	18.71	1.09	49.42	7.21	1.38	0.11
(E → $_U$RU) + AR	248.10	166.88	1,508.53	1,184.28	0.09	0.04	4.60	0.08	9.87	0.03	1.49	0.03	19.01	1.15	49.86	7.30	1.46	0.12
$_U$RU	1,875.92	1,328.11	6,933.49	6,460.49	0.11	0.08	4.00	0.05	6.72	0.20	0.52	0.01	35.91	0.93	66.81	4.86	6.21	0.11
							Long Short-Term Memory (LSTM)											
$_B$RU	3,042.70	1,987.48	18,539.83	17,591.06	3.80	0.41	3.68	0.02	6.23	0.13	0.42	0.01	20.34	0.87	48.42	6.44	1.37	0.10
E → $_B$RU	2,309.35	1,269.98	7,953.65	6,114.91	0.23	0.15	3.55	0.01	5.70	0.14	0.30	0.01	19.85	1.13	48.19	7.10	1.41	0.13
(E → $_B$RU + $_B$RU) + AR	256.32	59.69	1,412.55	487.27	0.12	0.07	**2.31**	**0.15**	**5.61**	**0.46**	**0.41**	**0.03**	**18.57**	**1.39**	**48.78**	**7.67**	**1.37**	**0.14**
(E → $_B$RU + $_U$RU) + AR	**244.81**	**128.06**	**1,233.11**	**666.28**	**0.18**	**0.21**	3.17	0.05	7.51	0.17	0.81	0.01	18.63	1.30	49.18	7.93	1.37	0.12
(E → $_B$RU) + AR	1,554.31	1,416.54	17,417.78	18,060.72	3.90	0.51	4.95	0.11	10.27	0.18	1.62	0.03	18.69	1.14	48.87	6.62	1.37	0.12
E → $_U$RU	3,169.42	2,063.50	12,461.42	11,674.12	0.17	0.05	3.75	0.16	6.26	0.22	0.44	0.04	20.74	1.02	49.10	7.14	1.46	0.14
(E → $_U$RU + $_B$RU) + AR	1,375.17	1,053.14	15,406.60	15,929.81	3.69	0.34	4.36	0.10	9.39	0.19	1.34	0.02	18.69	1.18	49.36	7.07	1.38	0.13
(E → $_U$RU + $_U$RU) + AR	285.02	121.63	1,802.48	1,118.25	0.11	0.06	5.27	0.05	10.69	0.02	1.76	0.01	18.78	1.10	49.68	6.64	1.39	0.15
(E → $_U$RU) + AR	265.80	112.59	1,437.64	686.42	0.14	0.09	7.91	0.14	13.67	0.20	2.95	0.04	18.84	1.17	49.71	7.81	1.40	0.11
$_U$RU	4,942.10	3,425.34	25,691.11	25,415.68	7.38	0.97	4.36	0.02	7.41	0.06	0.69	0.01	21.27	1.06	48.57	6.40	1.37	0.11

Table 7: Ablation on the main experiments with ReGENN's hyperparameters.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	SARS-CoV-2						**Brazilian Weather**						**PhysioNet**					
	MAE	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**
							Recurrent Neural Network (RNN)											
${}_{\text{B}}\text{RU}$	424.32	201.00	2819.11	2020.00	0.07	0.04	2.56	0.04	5.90	0.10	0.45	0.02	19.95	1.08	50.47	7.86	1.38	0.11
$\text{E} \rightarrow {}_{\text{B}}\text{RU}$	4161.35	3030.00	28755.33	28700.00	11.08	1.31	2.70	0.13	5.66	0.19	0.37	0.02	19.11	1.19	49.16	7.55	1.41	0.12
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	1624.44	1220.00	17245.24	17200.00	3.84	0.49	**2.18**	**0.06**	**5.52**	**0.29**	**0.37**	**0.02**	**18.72**	**0.89**	**48.60**	**6.57**	**1.37**	**0.13**
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	**257.16**	**149.00**	**1438.72**	**823.00**	**0.09**	**0.06**	2.84	0.15	6.47	0.15	0.55	0.01	18.97	1.27	49.59	7.78	1.39	0.11
$(\text{E} \rightarrow {}_{\text{B}}\text{RU}) + \text{AR}$	1471.49	953.00	17050.08	15700.00	3.74	0.34	3.59	0.08	8.13	0.08	0.98	0.02	18.81	1.23	50.20	7.62	1.38	0.12
$\text{E} \rightarrow {}_{\text{U}}\text{RU}$	1949.65	1190.00	16925.69	16100.00	3.84	0.41	3.12	0.11	5.86	0.04	0.32	0.01	19.47	1.24	50.24	7.48	1.44	0.15
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	372.81	190.00	2155.37	1320.00	0.11	0.07	2.36	0.12	5.69	0.24	0.41	0.03	18.76	1.05	48.81	6.47	1.39	0.14
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	1472.05	917.00	16491.81	15700.00	3.72	0.26	2.52	0.10	5.83	0.25	0.42	0.02	19.02	1.00	49.75	7.80	1.43	0.13
$(\text{E} \rightarrow {}_{\text{U}}\text{RU}) + \text{AR}$	1350.35	818.00	16673.11	15300.00	3.69	0.33	4.30	0.06	9.38	0.06	1.34	0.03	18.98	1.20	48.67	7.06	1.45	0.13
${}_{\text{U}}\text{RU}$	1175.99	769.00	6716.42	5340.00	2.08	0.73	2.40	0.07	5.32	0.19	0.32	0.01	20.58	1.00	50.32	7.02	1.42	0.12
							Gated Recurrent Unit (GRU)											
${}_{\text{B}}\text{RU}$	3130.47	2070.00	28063.07	24400.00	7.78	0.75	2.19	0.06	5.00	0.26	0.28	0.02	24.80	1.16	56.43	6.27	3.01	0.11
$\text{E} \rightarrow {}_{\text{B}}\text{RU}$	596.77	404.00	3446.67	2750.00	0.17	0.16	2.33	0.13	5.19	0.26	0.28	0.01	24.17	0.97	54.81	5.65	3.00	0.11
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	388.93	335.00	1888.70	1750.00	0.10	0.03	2.38	0.13	5.72	0.27	0.41	0.02	18.55	0.93	47.86	6.80	1.39	0.12
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	366.06	175.00	2158.32	1210.00	0.10	0.04	2.24	0.07	5.05	0.40	0.28	0.02	**18.54**	**0.94**	**48.90**	**5.63**	**1.37**	**0.13**
$(\text{E} \rightarrow {}_{\text{B}}\text{RU}) + \text{AR}$	1502.20	1010.00	16799.97	14800.00	3.67	0.39	3.31	0.06	7.57	0.29	0.82	0.03	18.67	0.91	48.59	6.26	1.39	0.11
$\text{E} \rightarrow {}_{\text{U}}\text{RU}$	968.83	674.00	5982.35	5430.00	0.22	0.07	2.81	0.15	5.69	0.11	0.34	0.01	19.42	1.26	51.03	7.90	1.44	0.15
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	**208.88**	**120.00**	**1141.56**	**576.00**	**0.08**	**0.04**	2.24	0.22	5.11	0.34	0.28	0.03	18.65	0.96	48.64	6.99	1.38	0.12
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	4052.71	2630.00	31384.80	28700.00	11.30	1.27	**2.10**	**0.10**	**4.97**	**0.23**	**0.28**	**0.01**	18.64	1.28	48.81	8.07	1.38	0.12
$(\text{E} \rightarrow {}_{\text{U}}\text{RU}) + \text{AR}$	2852.28	1740.00	25869.18	23700.00	7.60	0.59	3.25	0.08	7.55	0.12	0.81	0.02	18.82	0.93	49.98	7.61	1.41	0.13
${}_{\text{U}}\text{RU}$	1825.66	651.00	20400.04	15800.00	4.95	0.20	3.26	0.15	7.45	0.35	0.77	0.02	30.86	1.16	63.30	5.42	4.69	0.13
							Long Short-Term Memory (LSTM)											
${}_{\text{B}}\text{RU}$	1800.23	929.00	20264.27	17500.00	3.95	0.15	**2.20**	**0.03**	**5.04**	**0.14**	**0.28**	**0.01**	19.49	1.30	50.47	8.71	1.40	0.12
$\text{E} \rightarrow {}_{\text{B}}\text{RU}$	704.80	343.00	4088.05	2890.00	0.20	0.10	2.57	0.20	5.42	0.41	0.29	0.02	18.88	1.14	48.55	7.13	1.40	0.14
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	1598.03	884.00	19056.28	16900.00	3.89	0.17	2.44	0.12	5.75	0.25	0.41	0.01	**18.42**	**1.17**	**47.78**	**6.06**	**1.37**	**0.11**
$(\text{E} \rightarrow {}_{\text{B}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	4064.52	2700.00	31196.10	28700.00	11.30	1.01	3.17	0.20	7.42	0.26	0.80	0.03	18.46	0.85	48.33	6.48	1.36	0.13
$(\text{E} \rightarrow {}_{\text{B}}\text{RU}) + \text{AR}$	**211.93**	**83.40**	**1181.31**	**570.00**	**0.08**	**0.06**	3.61	0.10	8.19	0.10	0.96	0.01	18.54	1.16	47.35	6.65	1.38	0.11
$\text{E} \rightarrow {}_{\text{U}}\text{RU}$	2778.55	1110.00	12830.03	10200.00	0.21	0.06	2.96	0.15	5.54	0.12	0.30	0.01	19.78	1.02	51.29	8.67	1.44	0.14
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{B}}\text{RU}) + \text{AR}$	1380.99	864.00	16370.41	15000.00	3.66	0.32	2.47	0.06	5.77	0.14	0.41	0.03	18.55	0.87	48.28	6.58	1.38	0.12
$(\text{E} \rightarrow {}_{\text{U}}\text{RU} + {}_{\text{U}}\text{RU}) + \text{AR}$	1713.72	1030.00	18531.68	15900.00	3.81	0.39	4.88	0.05	10.24	0.07	1.61	0.01	18.63	1.20	48.93	7.08	1.37	0.14
$(\text{E} \rightarrow {}_{\text{U}}\text{RU}) + \text{AR}$	260.66	215.00	1513.92	1430.00	0.10	0.03	5.25	0.08	10.69	0.12	1.75	0.01	18.88	0.99	50.31	7.53	1.39	0.10
${}_{\text{U}}\text{RU}$	3723.88	2540.00	21022.60	20000.00	2.83	0.31	2.83	0.02	6.17	0.07	0.50	0.02	24.85	0.95	56.41	6.61	2.92	0.10

Hyperparameters used for Transfer Learning

Table 8: List of hyperparameters used during Transfer Learning.

	Transfer Learning					
	45 Days	**60 Days**	**75 Days**	**90 Days**	**105 Days**	**120 Days**
autoregression	True	True	True	True	True	True
batch-size	32	32	32	32	32	32
bias	True	True	True	True	True	True
bidirectional-gate	False	False	False	False	False	False
bidirectional-sequencer	False	False	False	False	False	False
clip-norm	15.0	0.0	10.0	0.0	0.0	85.0
criterion	MAE	MAE	MAE	MAE	MAE	MAE
dropout	0.25	0.0	0.35	0.1	0.0	0.0
early-stop	250	250	250	250	250	250
epochs	2500	5000	2500	2500	2500	2500
evolution-function	Identity	Identity	Identity	Identity	Identity	Identity
gate	LSTM	LSTM	LSTM	LSTM	LSTM	LSTM
iterator	Time	Time	Time	Time	Time	time
learning-rate	0.006	0.009	0.001	0.002	0.0002	0.001
load-weights	False	True	True	True	True	True
no-encoder	False	False	False	False	False	False
no-sequencer	False	False	False	False	False	False
normalization-axis	2	2	2	2	2	2
normalization-type	Maximum	Maximum	Maximum	Maximum	Maximum	Maximum
optimizer	Adam	Adam	Adam	Adam	Adam	Adam
output-function	ReLU	ReLU	ReLU	ReLU	ReLU	ReLU
random-seed	0	0	0	0	0	0
scheduler-factor	0.95	0.95	0.95	0.65	0.05	0.95
scheduler-min-lr	0.0	0.0	0.0	0.0	0.0	0.0
scheduler-patience	25	20	25	60	25	25
scheduler-threshold	0.1	0.01	0.1	0.1	0.1	0.1
sequencer	LSTM	LSTM	LSTM	LSTM	LSTM	LSTM
stride	14	14	14	14	14	14
validation-stride	7	7	7	7	7	7
watch-axis	2	2	2	2	2	2
window	7	7	7	7	7	7

Table 9: Detailed results for the first three slices of the SARS-CoV-2.

Legend: Algorithms with best performance are in **bold**, the ones noted as — yielded exceptions, and others as *** were suppressed due to poor performance.

	45 Days						60 Days						75 Days					
	MAE	± STD	RMSE	± STD	MSLE	± STD	MAE	± STD	RMSE	± STD	MSLE	± STD	MAE	± STD	RMSE	± STD	MSLE	± STD
Deep Learning Algorithms																		
ReGENN	**32.13**	**57.55**	**371.29**	**676.19**	**0.31**	**0.13**	**20.61**	**23.25**	**179.22**	**210.21**	**0.79**	**0.16**	**67.85**	**66.11**	**490.51**	**493.93**	**0.31**	**0.05**
MLCNN	151.22	1816.69	181.94	2081.31	1.43	4.54	353.39	1982.21	482.66	2420.29	6.31	10.00	1574.14	6425.31	2246.30	9462.22	14.44	14.70
DSANet	378.54	4759.23	411.23	4990.41	2.86	10.69	764.39	5758.71	917.00	5979.91	10.13	16.67	2746.73	11622.21	3471.10	14806.59	22.99	24.39
LSTNet	89.65	852.78	110.26	940.48	1.57	5.72	330.07	1538.96	475.37	2137.49	6.98	11.61	2023.90	9475.81	2648.69	12510.47	13.54	18.76
Multi-Output and Multi-Task Algorithms																		
CCA	387.21	127.88	3,812.86	670.27	1.96	0.52	572.70	112.76	5,423.93	423.77	2.25	0.49	1,768.42	194.76	11,192.02	2,977.72	1.98	0.52
Decision Tree	81.28	1.64	997.48	131.29	0.54	0.31	197.50	34.61	1,503.36	179.76	0.56	0.03	**606.66**	**257.03**	**3,639.93**	**1,636.15**	**0.67**	**0.12**
Extra Tree	87.90	6.31	1,010.82	121.86	0.55	0.31	228.55	80.70	1,617.34	23.65	0.70	0.12	819.14	367.86	5,297.72	2,520.36	0.90	0.12
Extra Trees	51.88	10.09	515.05	6.04	0.56	0.33	195.56	87.78	1,432.03	402.45	0.54	0.05	696.51	355.07	3,566.17	1,737.84	0.43	0.17
Gaussian Process	209.79	104.24	2,710.77	1,301.22	0.50	0.28	274.72	98.87	1,865.20	69.19	0.68	0.16	1,201.62	689.68	6,393.49	3,403.30	2.43	1.08
Historical Average	328.54	102.80	4,204.59	1,175.56	0.46	0.07	542.31	141.89	4,808.31	449.91	0.56	0.06	1,471.47	618.25	7,256.21	1,906.30	0.67	0.03
Kernel Ridge	57.95	9.15	609.72	0.58	0.67	0.29	230.48	113.90	1,605.85	565.99	1.26	0.50	856.45	395.81	4,384.56	1,488.20	0.81	0.33
KNeighbors	55.86	21.10	566.80	125.88	0.59	0.36	208.37	45.67	1,858.56	19.75	0.66	0.10	766.19	361.97	4,176.97	1,480.61	0.54	0.31
Lars	—	—	—	—	—	—	***	***	***	***	***	***	***	***	***	***	***	***
Lasso-Lars	328.54	102.80	4,204.59	1,175.56	0.46	0.07	542.31	141.89	4,808.31	449.91	0.56	0.06	1,471.47	618.25	7,256.21	1,906.30	0.67	0.03
Linear Regression	282.75	159.14	3,846.25	2,141.83	1.56	1.03	***	***	***	***	***	***	***	***	***	***	***	***
Multi-Task Elastic-Net	328.54	102.80	4,204.59	1,175.56	0.46	0.07	542.31	141.89	4,808.31	449.91	0.56	0.06	1,471.47	618.25	7,256.21	1,906.30	0.67	0.03
Multi-Task Lasso	328.54	102.80	4,204.59	1,175.56	0.46	0.07	542.31	141.89	4,808.31	449.91	0.56	0.06	1,471.47	618.25	7,256.21	1,906.30	0.67	0.03
Orthogonal Matching Pursuit	**41.91**	**4.57**	**426.72**	**47.62**	**0.56**	**0.34**	557.52	82.20	7,826.10	4,217.06	0.63	0.09	3,105.51	2,409.81	34,411.01	38,075.02	1.29	0.07
PLS Canonical	1,338.79	508.42	22,679.50	10,092.75	1.30	0.52	2,328.60	2,238.32	43,325.47	50,723.61	1.54	0.02	4,986.37	3,773.55	72,919.82	78,510.01	1.91	0.75
PLS	360.54	166.60	5,007.97	2,916.95	0.68	0.35	965.53	363.99	10,693.05	6,439.21	0.74	0.02	5,404.75	5,528.56	60,923.11	75,197.91	1.01	0.35
Radius Neighbors	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Random Forest	63.66	21.68	719.19	192.72	0.54	0.32	**193.86**	**84.33**	**1,405.11**	**372.31**	**0.51**	**0.04**	684.15	312.72	3,469.58	1,305.83	0.43	0.16
RANSAC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Ridge	57.67	9.48	608.22	3.14	0.52	0.31	230.04	113.79	1,608.98	568.38	0.69	0.18	851.08	396.66	4,375.78	1,486.11	0.45	0.17
Single-Target Algorithms on Chain Ensemble																		
AdaBoost	64.66	31.74	914.45	573.35	0.58	0.35	232.07	116.02	1,570.25	490.25	0.66	0.17	913.16	468.54	4,555.41	2,216.05	0.50	0.16
ARD	249.15	145.03	3,492.20	2,013.14	1.22	0.80	746.99	328.16	9,975.92	8,745.04	1.14	0.06	2,070.55	1,112.22	24,256.52	24,504.36	0.97	0.10
Bagging	74.38	18.62	877.74	175.47	0.53	0.32	214.39	106.38	1,632.65	586.02	0.48	0.02	687.52	321.81	3,503.36	1,481.42	0.45	0.17
Bayesian Ridge	168.97	88.19	2,732.62	1,495.76	0.94	0.61	385.57	185.21	3,149.31	927.97	1.03	0.17	896.91	414.00	4,532.26	1,632.95	0.46	0.17
CatBoost	90.59	31.85	1,031.37	292.43	0.52	0.31	259.99	104.90	2,375.76	716.46	0.46	0.02	606.33	227.25	3,378.25	998.34	0.42	0.15
Gradient Boosting	49.80	0.63	546.89	148.35	0.56	0.33	**190.73**	**83.11**	**1,243.65**	**210.91**	**0.51**	**0.03**	**563.49**	**253.34**	**3,087.26**	**1,419.53**	**0.47**	**0.19**
Histogram Grad. Boosting	88.14	67.19	1,045.45	1,058.30	0.64	0.20	317.10	181.62	2,144.85	1,149.52	0.82	0.25	573.71	178.47	3,040.38	507.34	0.88	0.19
Huber	**43.58**	**2.91**	**439.65**	**184.34**	**0.69**	**0.30**	272.56	143.23	1,911.75	677.08	1.11	0.40	890.99	450.34	4,991.10	2,033.22	1.06	0.23
Isotonic	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Lasso-Lars-IC	139.88	63.08	2,327.89	1,089.33	0.63	0.35	439.78	218.16	4,455.69	958.65	0.92	0.34	2,108.88	1,105.91	24,748.55	25,219.81	0.89	0.09
LGBM	100.85	58.20	1,163.55	974.79	0.64	0.20	283.25	151.31	2,066.16	1,015.90	0.61	0.09	681.31	277.35	3,714.49	995.86	0.48	0.11
Linear SVR	64.43	9.42	749.39	15.07	0.66	0.27	208.63	104.34	1,330.17	373.68	1.26	0.51	898.33	488.94	4,434.00	2,103.88	0.84	0.35
NuSVR	69.58	7.74	668.88	75.24	0.69	0.31	271.07	131.83	2,146.04	748.33	0.63	0.07	821.21	385.28	4,172.45	1,534.14	0.53	0.14
Passive Aggressive	109.67	52.64	1,835.38	908.67	0.34	0.13	298.56	135.27	2,121.33	509.49	0.81	0.38	1,192.12	636.62	6,796.09	3,098.40	1.11	0.15
SGD	76.74	34.66	815.18	631.38	0.51	0.26	282.62	147.38	2,071.76	952.88	0.46	0.03	934.62	420.51	4,603.72	1,524.15	0.44	0.13
SVR	79.75	13.15	784.25	17.13	0.44	0.23	244.21	101.24	1,973.38	513.88	0.71	0.25	812.32	400.33	4,135.83	1,547.69	0.46	0.19
Theil-Sen	91.26	48.34	1,047.89	808.99	0.62	0.27	240.39	125.96	2,033.75	963.63	0.60	0.10	842.23	415.84	5,116.64	2,179.12	0.97	0.16
Transformed Target	282.75	159.14	3,846.25	2,141.83	1.56	1.03	***	***	***	***	***	***	***	***	***	***	***	***
XGBoost	45.40	8.43	499.40	5.96	0.56	0.34	207.96	89.25	1,372.12	233.16	0.49	0.01	604.45	272.50	3,454.03	1,583.76	0.47	0.20
Time Series Algorithms																		
Autoregressive	348.60	141.89	4,511.58	1,800.83	1.95	1.04	565.61	276.92	4,912.10	2,148.18	5.19	2.79	2,269.85	1,375.06	13,416.69	8,254.88	7.92	3.76
ARIMA	349.35	141.59	4,516.17	1,794.31	1.94	1.04	419.66	171.52	4,625.91	1,941.85	3.31	1.51	1,826.65	1,039.10	12,448.50	7,422.94	6.22	2.46
ARMA	73.97	92.88	1,010.59	1,311.22	1.16	0.06	667.46	186.38	5,897.04	763.92	6.87	3.23	1,860.71	818.25	13,319.47	5,341.02	11.30	4.78
Moving Average	353.80	136.70	4,589.26	1,691.54	1.50	0.76	485.71	166.86	4,920.53	1,549.24	2.84	1.25	1,357.32	650.09	11,027.22	6,049.53	3.38	0.61
SARIMA	**54.29**	**67.17**	**716.26**	**892.21**	**0.80**	**0.21**	72.83	73.86	689.27	839.55	2.37	0.70	145.76	146.20	789.46	799.24	2.38	0.80
Exponential Smoothing	54.79	70.22	737.75	926.05	0.68	0.12	**71.39**	**84.12**	**709.93**	**937.92**	**2.33**	**0.73**	**127.92**	**169.73**	**711.72**	**906.47**	**2.16**	**0.51**
Vector Autoregression	364.13	111.03	4,811.29	1,376.14	1.73	0.39	478.40	58.18	5,603.67	557.96	2.61	0.42	1,148.39	362.46	12,038.19	4,524.07	3.18	0.22

Table 10: Detailed results for the last three slices of the SARS-CoV-2.

Legend: Algorithms with best performance are in **bold**, the ones noted as — yielded exceptions, and others as *** were suppressed due to poor performance.

	90 Days						**105 Days**						**120 Days**					
	MAE	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**	**MAE**	**± STD**	**RMSE**	**± STD**	**MSLE**	**± STD**
	Deep Learning Algorithms																	
ReGENN	**152.61**	**113.07**	**986.61**	**827.13**	**0.19**	**0.03**	**179.53**	**62.95**	**1,241.91**	**594.23**	**0.09**	**0.03**	**165.41**	**30.37**	**915.92**	**294.06**	**0.05**	**0.02**
MLCNN	4976.73	26967.49	6301.34	34530.94	17.72	18.67	4525.83	14092.01	5551.06	17189.83	12.78	19.26	10749.65	64733.89	13172.62	79729.95	15.55	23.33
DSANet	7762.14	35162.14	8875.20	41308.52	36.96	31.45	12466.20	58693.35	13910.18	67326.81	45.43	34.40	18428.63	81222.22	20131.28	91468.66	52.90	36.86
LSTNet	5698.94	32591.40	6658.76	39067.23	18.44	27.38	6080.57	53703.70	7050.81	62425.81	15.62	27.35	5298.30	22152.36	5779.21	23677.93	16.27	32.19
	Multi-Output and Multi-Task Algorithms																	
CCA	2,341.82	651.84	11,696.07	3,087.75	1.32	0.67	2,309.74	394.06	10,585.08	1,772.80	0.97	0.97	5,566.74	618.65	48,941.52	3,308.99	0.50	0.22
Decision Tree	**1,070.55**	**362.37**	**4,702.47**	**1,440.38**	**0.38**	**0.20**	1,618.26	206.36	8,656.78	1,525.96	0.27	0.09	1,098.77	163.13	6,055.55	2,122.73	0.27	0.05
Extra Tree	1,298.45	457.81	6,664.91	2,554.38	0.38	0.12	1,403.07	566.50	7,279.55	3,330.30	0.26	0.02	851.65	12.55	3,316.89	276.68	0.25	0.05
Extra Trees	1,125.67	487.87	5,238.64	2,386.86	0.22	0.10	1,213.77	281.65	5,371.02	1,217.81	0.15	0.02	771.33	120.86	3,823.39	5.44	0.14	0.08
Gaussian Process	1,727.28	781.46	7,647.19	2,479.27	2.19	0.99	2,137.51	411.08	8,600.10	541.87	1.38	0.07	2,730.59	309.13	14,295.71	3,034.59	0.57	0.47
Historical Average	2,127.99	606.38	10,609.89	3,223.15	0.34	0.19	2,079.79	413.43	9,428.33	2,131.05	0.22	0.00	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09
Kernel Ridge	1,371.02	337.03	7,263.13	2,026.30	0.26	0.18	1,295.63	318.42	5,289.67	1,241.99	0.17	0.03	797.81	78.97	2,917.93	162.53	0.11	0.04
KNeighbors	1,342.60	537.35	7,314.54	3,096.09	0.22	0.12	1,197.44	249.97	4,798.34	862.09	0.17	0.02	967.37	36.99	5,414.50	1,014.23	0.13	0.07
Lars	***	***	***	***	***	***	—	—	—	—	—	—	***	***	***	***	***	***
Lasso-Lars	2,127.99	606.38	10,609.89	3,223.15	0.34	0.19	2,079.79	413.42	9,428.33	2,131.05	0.22	0.00	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09
Linear Regression	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***	***
Multi-Task Elastic-Net	2,127.99	606.38	10,609.89	3,223.15	0.34	0.19	2,079.79	413.43	9,428.33	2,131.05	0.22	0.00	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09
Multi-Task Lasso	2,127.99	606.38	10,609.89	3,223.15	0.34	0.19	2,079.79	413.43	9,428.33	2,131.05	0.22	0.00	2,204.09	496.63	9,540.29	2,570.76	0.16	0.09
Orthogonal Matching Pursuit	5,124.70	3,058.36	62,230.48	56,134.60	0.28	0.18	3,123.49	802.13	29,161.34	5,194.85	0.19	0.00	2,142.70	897.09	19,881.75	10,964.98	0.17	0.09
PLS Canonical	7,129.60	3,477.83	106,376.76	78,979.43	0.84	0.82	9,943.51	5,019.18	135,467.09	100,867.91	0.63	0.28	8,743.59	2,964.01	84,481.55	58,065.34	0.39	0.11
PLS	3,818.23	2,606.97	39,133.11	37,842.39	0.29	0.23	2,316.76	1,496.43	20,758.84	22,298.29	0.19	0.02	1,341.96	155.97	6,461.25	322.42	0.50	0.01
Radius Neighbors	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Random Forest	1,144.34	416.79	5,292.91	1,902.90	0.21	0.10	**949.42**	**132.46**	**4,309.11**	**704.34**	**0.15**	**0.00**	**736.11**	**53.37**	**3,623.38**	**690.48**	**0.14**	**0.07**
RANSAC	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Ridge	1,373.02	355.28	7,295.49	2,075.05	0.21	0.11	1,307.54	317.33	5,339.62	1,231.87	0.16	0.00	804.49	86.67	2,957.27	191.77	0.14	0.08
	Single-Target Algorithms on Chain Ensemble																	
AdaBoost	1,335.46	518.11	6,967.35	2,706.46	0.21	0.11	1,129.36	200.58	4,745.23	771.75	0.19	0.02	1,099.84	337.60	7,184.23	5,335.13	0.15	0.09
ARD	3,673.37	1,994.35	43,202.60	40,595.97	0.27	0.19	1,765.81	130.95	13,621.08	1,163.70	0.18	0.00	889.47	30.26	3,368.50	207.97	0.16	0.08
Bagging	**1,120.38**	**425.26**	**5,256.17**	**2,099.54**	**0.22**	**0.11**	1,024.42	124.09	4,538.16	604.34	0.15	0.02	862.94	115.31	4,244.29	116.49	0.17	0.06
Bayesian Ridge	1,354.00	341.24	7,238.96	2,090.06	0.22	0.13	1,288.36	308.33	5,354.24	1,223.32	0.17	0.01	828.14	12.06	3,031.29	145.24	0.16	0.06
CatBoost	1,360.99	437.43	6,909.28	1,865.61	0.22	0.12	1,328.43	303.93	5,494.43	940.83	0.16	0.00	1,115.88	85.29	4,129.19	139.31	0.14	0.08
Gradient Boosting	1,182.57	409.10	5,680.21	2,013.45	0.23	0.13	1,055.45	99.35	5,995.73	1,150.59	0.16	0.02	**800.29**	**1.54**	**4,722.74**	**1,271.82**	**0.16**	**0.06**
Histogram Grad. Boosting	1,284.27	330.98	6,143.09	1,249.58	0.43	0.37	1,341.97	277.69	9,450.66	3,843.12	0.23	0.00	866.30	38.80	4,612.11	44.68	0.16	0.08
Huber	1,433.68	438.15	7,571.89	2,061.11	0.93	0.22	1,688.87	117.01	10,555.93	808.14	0.62	0.04	2,032.72	41.78	17,770.19	610.28	0.37	0.08
Isotonic	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Lasso-Lars-IC	3,844.71	2,113.02	46,007.40	43,846.83	0.27	0.18	2,808.14	844.84	30,186.71	9,531.58	0.18	0.01	810.03	30.97	3,929.81	129.59	0.14	0.08
LGBM	1,151.12	217.85	5,854.56	1,281.29	0.28	0.21	**870.05**	**15.88**	**4,611.39**	**516.98**	**0.17**	**0.01**	1,110.17	198.23	7,650.02	3,822.07	0.28	0.01
Linear SVR	1,349.30	265.00	7,580.89	1,602.24	0.29	0.22	917.52	9.41	3,773.66	127.97	0.19	0.05	875.51	97.79	3,540.90	247.08	0.16	0.01
NuSVR	1,505.71	493.60	7,609.47	2,106.44	0.24	0.13	943.65	152.83	3,635.38	312.43	0.16	0.01	809.97	19.17	2,973.63	21.61	0.12	0.06
Passive Aggressive	1,513.43	811.55	9,090.06	5,056.45	0.29	0.13	2,131.91	827.72	12,195.75	7,398.69	0.32	0.13	1,730.12	113.18	8,310.46	432.48	0.18	0.12
SGD	1,152.69	218.01	6,126.69	1,382.46	0.22	0.14	958.29	93.84	3,742.62	146.27	0.15	0.00	815.97	11.37	3,106.47	40.02	0.13	0.06
SVR	1,561.92	562.85	7,624.46	2,272.31	0.23	0.12	1,216.19	298.18	4,831.88	1,087.95	0.16	0.01	1,344.80	244.56	5,511.70	1,539.93	0.12	0.06
Theil-Sen	2,330.21	1,126.08	11,210.84	4,841.23	4.04	2.43	4,800.68	900.31	21,180.70	1,794.50	6.07	1.18	91,149.14	59,396.23	1,292,430.04	858,201.91	6.45	2.47
Transformed Target	10,464.84	1,595.88	119,669.67	56,077.14	14.32	0.93	36,788.18	37,380.39	721,732.88	945,437.62	16.99	4.01	68,530.56	55,751.11	868,586.61	989,030.54	16.39	1.00
XGBoost	1,252.61	426.10	6,089.49	2,145.76	0.30	0.22	1,070.54	142.59	5,820.65	1,367.38	0.16	0.01	806.03	37.02	5,033.10	1,599.32	0.16	0.06
	Time Series Algorithms																	
Autoregressive	5,621.04	3,216.40	35,169.53	21,541.40	8.58	4.33	8,788.66	5,007.74	57,016.07	35,534.58	8.94	5.44	11,529.79	6,752.42	76,118.62	48,374.82	9.15	6.14
ARIMA	3,548.24	1,712.65	32,183.46	19,248.04	5.48	1.49	8,845.58	5,039.42	57,008.21	35,546.27	9.83	4.19	11,578.80	6,798.84	76,044.64	48,478.91	9.22	5.23
ARMA	4,913.53	2,525.52	34,314.04	20,370.73	10.11	1.05	8,719.13	3,958.49	58,083.03	32,762.97	14.45	3.20	12,578.49	6,091.86	79,263.71	45,555.46	16.71	4.79
Moving Average	3,521.66	1,540.40	31,763.66	18,590.82	3.10	0.27	5,780.07	2,266.98	53,814.10	31,461.03	2.85	0.70	7,757.33	2,887.33	73,013.22	42,555.86	2.54	1.17
SARIMA	429.33	489.53	2,047.34	2,073.95	1.18	1.39	11,171.46	6,702.33	279,332.27	192,747.63	1.03	0.40	***	***	***	***	***	***
Exponential Smoothing	**381.55**	**538.84**	**1,692.42**	**2,384.16**	**1.06**	**1.21**	**576.39**	**814.72**	**2,300.48**	**3,242.27**	**0.55**	**0.71**	**667.33**	**940.85**	**2,607.29**	**3,634.47**	**0.27**	**0.17**
Vector Autoregression	***	***	***	***	***	***	12,985.56	9,735.09	270,265.90	274,777.24	3.23	0.02	6,395.14	2,397.58	75,292.43	39,167.56	3.21	0.05

Table 11: Ablation on the first 45 days of the SARS-CoV-2 dataset.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	PyTorch's Hyperparameters						**Literature's Hyperparameters**						ReGENN's **Hyperparameters**					
	MAE	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**
							Recurrent Neural Network (RNN)											
${}_{B}$RU	92.38	168.24	983.14	1,820.04	0.60	0.14	81.60	146.62	759.77	1,410.46	0.44	0.05	112.86	202.94	842.04	1,532.05	0.50	0.19
E → ${}_{B}$RU	154.44	294.30	1,312.81	2,517.02	0.87	0.33	134.87	260.04	1,232.46	2,394.62	0.86	0.56	227.58	426.24	1,667.99	3,102.15	1.36	0.84
(E → ${}_{B}$RU + ${}_{B}$RU) + AR	123.91	236.71	1,081.02	2,072.41	0.78	0.61	**38.92**	**73.72**	**389.25**	**753.11**	**0.39**	**0.09**	43.86	76.12	385.74	689.40	0.39	0.13
(E → ${}_{B}$RU + ${}_{U}$RU) + AR	130.66	244.62	1,127.02	2,128.93	0.83	0.42	38.93	72.00	401.29	758.97	0.40	0.06	40.10	71.31	369.97	668.56	0.35	0.08
(E → ${}_{B}$RU) + AR	117.76	224.79	1,096.91	2,125.03	0.78	0.39	86.93	167.43	928.81	1,812.63	0.67	0.31	52.23	94.22	519.21	947.12	0.38	0.11
E → ${}_{U}$RU	161.50	301.14	1,393.98	2,596.00	1.03	0.44	114.50	218.48	969.78	1,869.53	0.74	0.44	120.86	213.88	888.87	1,580.65	0.76	0.40
(E → ${}_{U}$RU + ${}_{B}$RU) + AR	125.39	236.08	968.55	1,836.12	0.73	0.44	60.47	110.75	717.60	1,351.65	0.48	0.08	33.97	60.20	378.05	685.64	0.31	0.08
(E → ${}_{U}$RU + ${}_{U}$RU) + AR	143.91	252.03	1,365.39	2,416.99	1.11	0.47	42.22	74.12	388.16	689.13	0.40	0.17	**33.47**	**59.33**	**346.66**	**616.35**	**0.30**	**0.10**
(E → ${}_{U}$RU) + AR	**73.49**	**133.81**	**895.15**	**1,689.02**	**0.58**	**0.21**	41.19	76.65	423.19	810.52	0.36	0.07	33.56	59.79	381.37	697.52	0.30	0.17
${}_{U}$RU	152.68	283.80	1,419.21	2,639.93	1.06	0.64	107.50	194.83	961.91	1,773.30	0.63	0.17	132.75	248.92	890.12	1,664.24	0.68	0.36
							Gated Recurrent Unit (GRU)											
${}_{B}$RU	139.53	251.27	1,351.34	2,425.51	1.06	0.30	110.63	193.52	1,050.93	1,867.69	0.86	0.33	131.26	229.29	1,036.19	1,772.92	0.68	0.10
E → ${}_{B}$RU	119.39	234.78	928.39	1,825.68	0.39	0.19	70.22	132.42	612.60	1,172.71	0.48	0.13	71.47	131.55	560.74	1,045.89	0.42	0.14
(E → ${}_{B}$RU + ${}_{B}$RU) + AR	266.39	493.08	1,837.22	3,355.49	0.92	0.28	**37.16**	**67.72**	**412.10**	**757.24**	**0.29**	**0.14**	42.77	69.71	380.37	639.67	0.38	0.12
(E → ${}_{B}$RU + ${}_{U}$RU) + AR	252.54	464.95	1,801.24	3,242.68	1.24	0.33	60.86	116.71	705.36	1,373.44	0.50	0.20	**35.42**	**62.81**	**396.71**	**720.01**	**0.35**	**0.05**
(E → ${}_{B}$RU) + AR	140.96	254.21	1,311.44	2,259.16	0.92	0.29	39.24	73.68	414.09	797.69	0.38	0.06	57.46	106.34	700.14	1,334.55	0.45	0.14
E → ${}_{U}$RU	**110.85**	**202.79**	**1,127.41**	**2,015.28**	**0.92**	**0.39**	119.25	217.88	1,044.76	1,833.26	0.70	0.20	137.64	259.08	1,054.46	1,947.95	0.75	0.29
(E → ${}_{U}$RU + ${}_{B}$RU) + AR	131.94	245.52	1,218.18	2,240.15	1.02	0.43	41.17	73.11	449.23	797.88	0.47	0.13	42.33	73.97	402.23	721.30	0.34	0.10
(E → ${}_{U}$RU + ${}_{U}$RU) + AR	219.26	408.35	1,698.29	3,173.21	1.66	0.70	62.52	115.16	744.72	1,396.55	0.51	0.15	90.67	163.53	1,007.57	1,829.93	0.75	0.29
(E → ${}_{U}$RU) + AR	156.25	275.91	1,494.13	2,650.03	1.21	0.49	90.03	163.76	986.54	1,824.13	0.69	0.28	104.06	194.04	1,104.26	2,083.88	0.85	0.34
${}_{U}$RU	138.25	252.42	1,321.32	2,399.06	1.05	0.58	180.26	340.49	1,440.16	2,736.77	1.23	0.88	163.53	296.28	1,213.40	2,181.95	0.91	0.42
							Long Short-Term Memory (LSTM)											
${}_{B}$RU	171.79	328.51	1,457.40	2,772.59	1.24	0.82	152.31	271.20	1,370.13	2,445.64	1.09	0.37	150.75	274.94	1,152.45	2,085.77	0.83	0.17
E → ${}_{B}$RU	114.78	206.10	1,068.62	1,939.87	0.79	0.32	70.51	130.64	562.77	1,059.56	0.27	0.05	108.36	193.72	775.35	1,402.93	0.51	0.15
(E → ${}_{B}$RU + ${}_{B}$RU) + AR	211.74	385.14	1,704.56	3,117.28	1.39	0.88	85.36	162.34	939.39	1,804.93	0.65	0.41	83.51	153.19	952.25	1,770.62	0.62	0.22
(E → ${}_{B}$RU + ${}_{U}$RU) + AR	**92.34**	**175.30**	**979.40**	**1,868.96**	**0.58**	**0.29**	62.69	115.65	729.74	1,383.14	0.50	0.14	37.11	67.01	410.27	754.33	0.32	0.14
(E → ${}_{B}$RU) + AR	131.72	247.70	1,277.84	2,410.56	0.89	0.37	81.91	151.00	939.29	1,758.27	0.65	0.28	139.78	258.71	1,325.73	2,444.80	0.98	0.52
E → ${}_{U}$RU	200.08	363.84	1,563.10	2,787.79	1.14	0.44	159.26	295.13	1,237.13	2,243.83	0.81	0.32	119.47	206.38	769.78	1,319.61	0.50	0.23
(E → ${}_{U}$RU + ${}_{B}$RU) + AR	134.88	249.97	1,309.42	2,359.58	1.14	0.42	92.05	167.31	1,054.34	1,881.80	0.87	0.40	39.04	73.06	416.84	798.12	0.33	0.12
(E → ${}_{U}$RU + ${}_{U}$RU) + AR	113.29	213.19	1,120.50	2,112.08	0.84	0.53	88.22	159.65	1,002.30	1,834.26	0.71	0.19	61.23	113.29	730.30	1,381.64	0.47	0.11
(E → ${}_{U}$RU) + AR	145.47	262.58	1,432.06	2,606.44	1.13	0.63	**39.04**	**67.51**	**420.52**	**750.02**	**0.33**	**0.17**	**32.79**	**58.34**	**378.05**	**688.07**	**0.31**	**0.09**
${}_{U}$RU	155.38	277.69	1,277.61	2,286.99	0.81	0.26	199.23	363.06	1,554.73	2,793.46	1.18	0.37	173.46	321.89	1,247.96	2,344.72	0.75	0.33

Table 12: Ablation on the first 60 days of the SARS-CoV-2 dataset.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	PyTorch's Hyperparameters						**Literature's Hyperparameters**						ReGENN's **Hyperparameters**					
	MAE	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**
							Recurrent Neural Network (RNN)											
$_{B}$RU	**43.57**	**38.12**	**323.55**	**297.46**	**0.71**	**0.09**	144.34	118.88	1,275.31	1,270.70	1.43	0.32	86.60	97.98	446.65	498.60	0.61	0.12
E $\rightarrow$ $_{B}$RU	253.53	374.90	1,752.47	2,739.24	2.90	1.18	219.04	304.26	1,575.91	2,457.67	2.09	0.44	241.50	315.27	1,601.64	2,244.13	2.36	0.51
(E $\rightarrow$ $_{B}$RU + $_{B}$RU) + AR	175.65	219.43	1,211.78	1,656.35	1.70	0.24	53.65	59.24	638.43	935.69	0.66	0.07	**30.77**	**16.65**	**243.40**	**185.19**	**0.92**	**0.20**
(E $\rightarrow$ $_{B}$RU + $_{U}$RU) + AR	93.77	121.73	922.43	1,329.31	1.05	0.15	**39.10**	**35.06**	**346.88**	**331.93**	**0.78**	**0.12**	35.03	20.29	239.61	195.37	0.80	0.13
(E $\rightarrow$ $_{B}$RU) + AR	189.30	229.91	1,693.76	2,266.29	2.38	0.34	133.75	165.63	1,479.53	2,016.71	1.63	0.32	70.69	61.18	961.55	1,201.27	1.30	0.20
E $\rightarrow$ $_{U}$RU	83.38	94.75	1,041.24	1,249.44	1.29	0.34	106.04	147.65	495.28	658.96	0.76	0.18	80.69	90.97	430.62	516.05	1.51	0.49
(E $\rightarrow$ $_{U}$RU + $_{B}$RU) + AR	143.76	224.86	1,285.79	2,042.16	1.86	0.46	42.94	21.19	336.45	202.77	0.99	0.10	31.13	22.61	212.93	173.76	0.68	0.22
(E $\rightarrow$ $_{U}$RU + $_{U}$RU) + AR	85.20	117.27	827.96	1,314.00	1.27	0.36	81.56	79.32	924.11	1,224.77	1.37	0.27	31.03	19.91	217.80	169.85	0.71	0.22
(E $\rightarrow$ $_{U}$RU) + AR	116.63	153.58	1,209.29	1,793.28	1.66	0.34	40.78	35.11	542.23	490.54	0.72	0.15	33.34	19.26	251.21	193.90	0.72	0.09
$_{U}$RU	72.64	84.28	730.01	880.44	1.05	0.19	91.29	68.33	477.49	391.91	0.74	0.10	104.92	81.72	767.81	525.68	1.42	0.50
							Gated Recurrent Unit (GRU)											
$_{B}$RU	122.88	106.19	1,335.11	1,227.14	1.87	0.15	196.72	166.02	1,811.57	1,657.98	2.36	0.27	162.40	174.15	1,259.63	1,431.54	1.89	0.13
E $\rightarrow$ $_{B}$RU	70.41	83.18	536.49	592.37	0.77	0.11	94.89	74.57	564.44	568.49	0.80	0.19	175.12	229.00	1,371.57	1,934.60	1.16	0.34
(E $\rightarrow$ $_{B}$RU + $_{B}$RU) + AR	132.02	112.80	807.68	900.76	1.08	0.20	36.06	19.09	300.60	198.17	0.82	0.21	29.73	22.89	224.72	220.17	0.90	0.18
(E $\rightarrow$ $_{B}$RU + $_{U}$RU) + AR	154.86	260.17	892.88	1,498.91	1.28	0.14	36.37	23.99	351.25	321.62	0.73	0.09	**26.37**	**16.57**	**186.60**	**169.99**	**0.76**	**0.15**
(E $\rightarrow$ $_{B}$RU) + AR	101.44	156.22	838.37	1,386.84	1.18	0.29	56.13	67.31	534.99	667.50	0.70	0.07	105.82	146.13	1,109.72	1,768.27	1.65	0.39
E $\rightarrow$ $_{U}$RU	153.46	160.30	1,534.67	1,336.70	1.88	0.34	168.26	154.76	1,464.07	1,294.01	1.74	0.31	249.24	213.83	1,989.17	1,935.53	2.41	0.37
(E $\rightarrow$ $_{U}$RU + $_{B}$RU) + AR	**59.72**	**59.97**	**470.12**	**544.68**	**1.41**	**0.10**	**30.07**	**18.19**	**293.46**	**189.92**	**0.60**	**0.11**	102.63	120.04	1,213.07	1,410.27	1.72	0.32
(E $\rightarrow$ $_{U}$RU + $_{U}$RU) + AR	138.05	110.69	1,430.42	1,252.14	2.32	0.35	40.99	45.09	326.81	369.76	0.77	0.21	136.89	117.29	1,702.69	1,589.37	2.20	0.31
(E $\rightarrow$ $_{U}$RU) + AR	228.16	306.27	1,892.92	2,534.10	3.12	0.82	63.48	91.29	636.59	945.50	0.65	0.08	69.33	89.52	823.69	1,271.88	1.36	0.30
$_{U}$RU	201.01	156.47	1,980.91	1,866.96	2.72	0.83	249.49	377.04	1,712.88	2,439.47	2.88	1.07	203.73	207.45	1,796.38	2,009.88	2.33	0.31
							Long Short-Term Memory (LSTM)											
$_{B}$RU	343.04	362.51	2,505.45	2,804.50	4.57	1.22	351.66	352.76	2,597.14	2,669.53	4.41	0.68	160.29	147.31	1,332.31	1,276.44	1.62	0.17
E $\rightarrow$ $_{B}$RU	140.21	171.49	1,327.95	1,795.97	1.53	0.34	111.11	127.42	607.34	751.98	0.63	0.27	149.57	204.37	1,034.95	1,524.12	1.24	0.37
(E $\rightarrow$ $_{B}$RU + $_{B}$RU) + AR	137.21	134.69	1,424.55	1,717.85	1.94	0.39	93.71	87.38	1,182.30	1,293.40	1.47	0.18	110.13	140.48	1,269.03	1,768.95	2.12	0.38
(E $\rightarrow$ $_{B}$RU + $_{U}$RU) + AR	137.24	152.13	1,381.15	1,753.82	1.77	0.24	94.28	105.97	1,123.98	1,422.39	1.22	0.17	67.52	54.95	965.37	1,188.71	1.28	0.19
(E $\rightarrow$ $_{B}$RU) + AR	**85.07**	**98.50**	**897.68**	**1,271.08**	**1.18**	**0.38**	94.10	97.26	1,143.15	1,248.17	1.39	0.17	245.25	323.28	1,898.61	2,562.27	3.47	0.86
E $\rightarrow$ $_{U}$RU	211.64	191.72	1,674.10	1,372.85	1.83	0.32	146.44	129.21	819.64	774.25	0.69	0.22	138.36	105.50	776.30	734.64	0.99	0.30
(E $\rightarrow$ $_{U}$RU + $_{B}$RU) + AR	224.49	329.07	1,942.68	2,672.95	2.48	0.77	72.96	82.00	890.21	1,273.83	1.18	0.13	166.00	292.10	1,510.17	2,698.45	1.74	0.84
(E $\rightarrow$ $_{U}$RU + $_{U}$RU) + AR	215.84	307.51	1,943.65	2,726.70	2.47	0.69	135.95	172.25	1,362.01	1,831.27	2.00	0.39	124.01	139.02	1,366.26	1,754.75	2.05	0.42
(E $\rightarrow$ $_{U}$RU) + AR	177.16	132.98	1,977.65	1,606.18	2.55	0.44	**47.56**	**25.51**	**529.98**	**269.66**	**0.77**	**0.27**	**26.79**	**11.42**	**212.39**	**150.62**	**0.77**	**0.16**
$_{U}$RU	267.83	346.86	1,792.51	2,281.50	2.72	0.57	326.20	433.08	2,150.92	2,754.01	3.38	0.82	150.72	113.44	861.31	708.08	1.03	0.25

Table 13: Ablation on the first 75 days of the SARS-CoV-2 dataset.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	PyTorch's Hyperparameters						**Literature's Hyperparameters**						ReGENN's **Hyperparameters**					
	MAE	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**
	Recurrent Neural Network (RNN)																	
$_{B}$RU	161.14	97.42	1,435.75	1,110.46	0.66	0.14	411.17	303.24	2,037.13	1,755.05	0.71	0.07	504.53	388.17	2,589.88	1,795.09	0.78	0.23
E → $_{B}$RU	718.88	325.71	5,240.10	2,714.46	6.51	0.60	683.60	356.26	4,074.54	2,244.10	4.12	0.37	797.03	273.84	4,669.25	2,198.02	5.29	0.36
(E → $_{B}$RU + $_{B}$RU) + AR	318.26	308.94	2,813.68	2,311.01	2.13	0.46	170.41	119.06	1,594.96	1,416.35	1.89	0.35	182.35	143.06	1,692.18	1,759.38	2.15	0.38
(E → $_{B}$RU + $_{U}$RU) + AR	**151.45**	**49.83**	**1,339.26**	**731.89**	**0.71**	**0.11**	180.85	183.45	1,609.37	1,852.03	2.01	0.56	183.33	158.36	1,752.28	1,692.14	1.77	0.40
(E → $_{B}$RU) + AR	524.40	236.82	4,032.76	2,150.00	5.09	0.52	289.20	192.77	2,835.31	1,756.82	2.15	0.30	356.14	198.64	3,558.44	1,649.24	2.38	0.50
E → $_{U}$RU	653.11	115.79	6,928.49	2,979.28	4.48	0.46	448.82	340.43	2,034.42	1,708.50	1.45	0.21	404.58	296.14	1,711.90	1,160.99	1.33	0.38
(E → $_{U}$RU + $_{B}$RU) + AR	391.36	282.20	3,203.38	2,431.20	3.31	0.51	**123.58**	**64.91**	**1,039.56**	**450.26**	**2.10**	**0.47**	**145.41**	**106.91**	**1,100.23**	**810.37**	**1.59**	**0.29**
(E → $_{U}$RU + $_{U}$RU) + AR	250.54	198.90	2,111.15	1,554.28	1.92	0.14	161.39	100.01	1,668.05	1,432.03	0.81	0.30	157.09	128.19	1,623.64	1,582.04	1.00	0.45
(E → $_{U}$RU) + AR	336.16	169.55	2,928.13	1,643.90	3.43	0.45	134.50	69.89	1,069.85	758.32	0.70	0.15	149.06	72.80	1,392.70	916.71	0.96	0.25
$_{U}$RU	174.78	121.62	1,371.81	905.07	1.65	0.48	518.13	300.14	3,090.42	2,026.79	1.79	0.41	504.14	264.01	3,091.64	1,803.63	1.51	0.14
	Gated Recurrent Unit (GRU)																	
$_{B}$RU	713.73	305.34	6,470.68	3,376.01	4.39	0.42	1,042.15	894.79	6,395.98	5,717.38	6.21	1.08	818.92	479.63	6,154.47	3,878.30	4.36	0.61
E → $_{B}$RU	191.61	135.71	1,470.70	1,165.24	0.86	0.18	388.02	258.39	1,624.89	1,018.48	0.99	0.18	459.37	447.34	2,092.11	2,324.37	1.00	0.23
(E → $_{B}$RU + $_{B}$RU) + AR	**163.11**	**136.68**	**1,102.59**	**880.54**	**0.61**	**0.11**	**134.85**	**158.37**	**1,177.67**	**1,542.84**	**0.53**	**0.14**	152.27	123.36	1,527.13	1,609.60	0.56	0.16
(E → $_{B}$RU + $_{U}$RU) + AR	311.42	161.97	2,924.49	1,658.64	2.11	0.39	149.88	162.32	1,554.70	1,972.45	0.52	0.29	169.54	152.49	1,662.18	1,594.09	1.82	0.42
(E → $_{B}$RU) + AR	227.22	227.73	1,796.87	1,618.45	1.74	0.24	241.01	129.01	2,481.91	1,363.01	1.92	0.33	143.98	102.04	1,397.98	1,238.76	1.24	0.22
E → $_{U}$RU	195.72	74.66	1,728.65	1,166.72	0.95	0.15	426.13	210.99	2,117.59	1,303.15	0.60	0.16	659.88	438.00	5,309.42	3,948.28	2.37	0.29
(E → $_{U}$RU + $_{B}$RU) + AR	206.54	126.84	1,381.13	1,060.64	0.87	0.16	176.50	114.42	1,814.09	1,510.40	1.55	0.33	**141.97**	**65.56**	**1,386.72**	**807.02**	**1.80**	**0.29**
(E → $_{U}$RU + $_{U}$RU) + AR	693.87	478.19	5,567.19	4,299.04	4.46	0.83	359.02	295.11	3,667.50	2,315.53	3.23	0.53	309.70	167.80	3,485.05	1,702.31	2.46	0.34
(E → $_{U}$RU) + AR	568.63	279.26	5,961.40	3,709.67	4.14	0.39	393.74	115.74	5,773.68	2,751.32	2.49	0.21	802.04	344.69	7,315.72	4,095.29	6.01	0.97
$_{U}$RU	627.82	328.31	5,932.93	3,530.15	4.56	0.65	359.47	125.72	1,649.94	659.50	0.39	0.10	834.36	512.41	5,994.85	3,940.99	4.75	0.41
	Long Short-Term Memory (LSTM)																	
$_{B}$RU	138.53	117.29	917.36	840.42	0.47	0.07	649.80	438.23	4,846.82	3,199.61	2.47	0.51	1,050.28	657.98	7,328.83	4,749.73	6.11	1.13
E → $_{B}$RU	380.34	314.46	2,744.68	2,147.57	3.14	0.52	553.72	549.20	2,518.26	2,639.77	2.19	0.41	516.29	180.14	2,854.97	1,122.73	1.83	0.23
(E → $_{B}$RU + $_{B}$RU) + AR	102.62	20.06	836.03	153.62	0.46	0.12	374.85	268.30	3,495.07	2,567.92	3.81	0.45	349.46	202.43	3,512.31	2,416.90	2.52	0.18
(E → $_{B}$RU + $_{U}$RU) + AR	296.93	194.02	2,727.79	1,834.36	2.12	0.25	474.71	295.70	4,039.41	2,350.60	4.99	0.75	191.60	154.19	1,769.89	1,568.53	1.83	0.29
(E → $_{B}$RU) + AR	443.01	350.65	3,607.06	3,112.68	3.55	0.45	302.57	61.93	3,981.58	1,071.91	2.27	0.17	307.15	384.35	2,834.08	3,028.40	2.20	0.48
E → $_{U}$RU	558.91	337.01	2,976.82	1,697.81	1.28	0.18	623.12	396.10	2,943.44	1,825.41	0.49	0.05	550.42	511.73	2,189.06	2,087.18	0.51	0.13
(E → $_{U}$RU + $_{B}$RU) + AR	270.76	118.92	2,365.14	1,401.32	2.49	0.26	165.43	141.71	1,456.72	1,484.11	2.31	0.59	157.76	143.09	1,123.17	1,015.65	2.90	0.78
(E → $_{U}$RU + $_{U}$RU) + AR	315.88	192.72	2,975.82	1,611.11	2.05	0.09	345.42	219.75	3,470.74	1,963.87	3.36	0.79	194.90	130.19	1,965.46	1,608.19	1.68	0.67
(E → $_{U}$RU) + AR	**100.52**	**79.84**	**743.39**	**567.97**	**0.46**	**0.08**	**84.27**	**35.68**	**654.90**	**326.59**	**0.44**	**0.06**	**78.74**	**48.82**	**614.42**	**377.49**	**0.39**	**0.12**
$_{U}$RU	716.32	508.83	3,639.20	2,751.68	2.78	0.38	821.10	300.26	4,518.70	1,855.99	3.79	0.55	751.09	527.12	3,599.69	2,691.51	2.73	0.58

Table 14: Ablation on the first 90 days of the SARS-CoV-2 dataset.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	PyTorch's Hyperparameters						**Literature's Hyperparameters**						ReGENN's **Hyperparameters**					
	MAE	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**
	Recurrent Neural Network (RNN)																	
$_{\mathrm{B}}$RU	**242.94**	**140.17**	**1,541.55**	**1,048.30**	**0.23**	**0.02**	1,511.05	738.58	10,258.68	7,629.47	3.21	0.22	4,126.78	1,477.31	19,562.50	13,001.12	3.40	0.15
E $\rightarrow$ $_{\mathrm{B}}$RU	2,264.19	1,633.01	15,315.27	12,110.71	10.15	1.95	1,886.29	1,371.64	10,610.35	9,311.24	5.73	0.96	3,887.42	2,694.59	15,666.70	14,370.03	3.02	0.45
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{B}}$RU) + AR	778.99	310.60	8,999.00	5,753.81	2.76	0.27	869.49	708.33	7,935.13	7,653.96	3.54	0.59	**193.31**	**176.40**	**1,202.10**	**992.75**	**0.22**	**0.06**
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{U}}$RU) + AR	318.65	232.28	1,707.38	1,300.59	0.24	0.08	362.09	161.48	3,352.83	1,248.02	1.31	0.26	199.27	165.91	1,348.27	1,154.47	0.21	0.06
(E $\rightarrow$ $_{\mathrm{B}}$RU) + AR	742.97	492.51	7,333.20	6,379.55	2.70	0.45	237.05	118.27	1,876.90	410.60	0.20	0.10	206.64	130.16	1,515.35	1,080.97	0.22	0.11
E $\rightarrow$ $_{\mathrm{U}}$RU	738.67	474.03	7,120.58	5,497.46	2.84	0.44	1,473.40	1,176.38	5,987.45	5,215.71	0.78	0.32	3,646.65	1,630.25	16,109.81	11,125.85	5.74	1.34
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{B}}$RU) + AR	283.06	75.59	2,054.32	903.32	0.28	0.14	279.72	123.54	2,286.41	1,084.17	1.05	0.54	207.86	110.68	1,493.80	917.88	0.19	0.03
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{U}}$RU) + AR	673.46	209.93	7,651.99	3,975.96	2.70	0.31	740.28	478.88	7,154.20	5,232.24	3.64	0.92	201.83	177.38	1,256.22	1,254.76	0.23	0.12
(E $\rightarrow$ $_{\mathrm{U}}$RU) + AR	703.48	527.47	7,034.98	5,485.51	2.56	0.21	**190.42**	**190.32**	**1,144.52**	**1,230.53**	**0.22**	**0.08**	197.50	64.81	1,534.59	742.72	0.21	0.07
$_{\mathrm{U}}$RU	365.99	326.64	1,992.32	1,814.87	2.11	0.71	906.45	577.09	3,622.20	2,874.36	1.21	0.44	3,926.71	2,129.55	17,143.74	12,706.58	0.72	0.08
	Gated Recurrent Unit (GRU)																	
$_{\mathrm{B}}$RU	890.38	580.01	9,857.40	9,307.51	3.08	0.32	1,442.15	837.67	10,937.08	8,662.15	3.06	0.41	4,308.93	1,709.03	20,693.26	13,930.29	5.33	0.23
E $\rightarrow$ $_{\mathrm{B}}$RU	426.15	373.83	2,881.20	2,435.46	0.36	0.14	1,096.77	857.15	4,092.96	3,413.09	0.45	0.08	3,427.21	1,147.82	15,347.22	9,743.89	1.26	0.48
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{B}}$RU) + AR	249.97	92.04	1,688.83	890.59	0.28	0.15	252.57	136.37	2,052.86	1,448.23	0.24	0.11	190.44	143.42	1,209.26	873.04	0.18	0.07
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{U}}$RU) + AR	791.02	517.33	7,613.47	6,661.82	2.72	0.44	461.40	218.69	4,287.87	1,452.17	2.06	0.61	190.61	139.96	1,315.81	981.13	0.21	0.07
(E $\rightarrow$ $_{\mathrm{B}}$RU) + AR	**228.32**	**117.23**	**1,457.24**	**969.59**	**0.23**	**0.08**	**239.36**	**158.83**	**1,734.29**	**1,297.86**	**0.42**	**0.16**	211.31	186.86	1,298.85	1,191.48	0.21	0.09
E $\rightarrow$ $_{\mathrm{U}}$RU	342.96	259.87	2,110.70	1,719.47	0.40	0.15	2,161.34	1,746.48	11,637.07	11,359.42	1.01	0.37	3,832.96	3,514.14	16,232.39	14,729.52	4.53	0.70
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{B}}$RU) + AR	951.85	439.86	11,261.48	8,436.19	3.12	0.26	305.29	309.49	2,145.37	2,199.81	2.11	0.52	848.35	505.80	10,591.76	9,146.85	3.17	0.72
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{U}}$RU) + AR	1,839.28	1,245.97	14,492.23	12,161.61	8.00	0.94	723.14	456.53	8,472.29	7,068.90	3.06	0.55	**179.35**	**102.26**	**1,253.62**	**801.38**	**0.18**	**0.06**
(E $\rightarrow$ $_{\mathrm{U}}$RU) + AR	285.47	78.53	2,432.23	871.10	0.24	0.08	869.42	690.38	8,641.86	10,078.24	0.91	0.38	1,238.96	924.89	11,673.27	9,827.31	5.38	0.45
$_{\mathrm{U}}$RU	883.60	468.32	10,247.84	7,336.02	3.62	0.46	1,107.06	700.25	6,164.05	4,731.55	1.65	0.18	3,888.44	1,606.01	17,143.93	12,597.02	0.58	0.11
	Long Short-Term Memory (LSTM)																	
$_{\mathrm{B}}$RU	**257.99**	**143.04**	**1,730.68**	**1,018.27**	**0.22**	**0.02**	2,010.31	1,131.95	14,467.37	12,147.42	5.88	0.69	4,069.37	3,919.60	16,311.66	17,267.56	3.37	0.19
E $\rightarrow$ $_{\mathrm{B}}$RU	365.42	184.23	2,534.62	1,653.02	0.31	0.12	1,367.59	1,110.26	5,612.86	5,986.90	0.62	0.21	3,692.45	2,757.03	14,303.79	13,581.50	1.38	0.49
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{B}}$RU) + AR	1,657.74	818.46	13,233.88	9,759.93	7.61	0.85	738.94	399.30	8,983.37	7,134.58	3.28	0.49	194.23	100.19	1,342.74	762.09	0.18	0.06
(E $\rightarrow$ $_{\mathrm{B}}$RU + $_{\mathrm{U}}$RU) + AR	1,207.08	834.76	10,220.21	8,595.59	5.08	0.56	825.39	382.22	8,795.38	5,960.91	4.06	1.62	184.65	152.93	1,204.69	940.83	0.19	0.09
(E $\rightarrow$ $_{\mathrm{B}}$RU) + AR	882.08	801.18	8,251.39	8,533.21	2.90	0.34	1,274.05	1,083.48	11,276.86	10,900.65	5.61	0.77	183.95	162.25	1,125.99	1,060.32	0.20	0.07
E $\rightarrow$ $_{\mathrm{U}}$RU	1,848.50	854.54	11,982.81	8,193.23	3.11	0.34	1,988.35	1,111.96	8,604.09	6,648.45	0.63	0.26	4,234.82	2,935.90	19,072.87	16,114.08	6.08	1.08
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{B}}$RU) + AR	649.64	489.43	6,142.64	5,304.13	2.57	0.30	191.33	118.30	1,614.64	1,416.72	0.47	0.12	191.16	171.66	1,191.72	1,058.59	0.20	0.08
(E $\rightarrow$ $_{\mathrm{U}}$RU + $_{\mathrm{U}}$RU) + AR	669.24	423.71	7,508.73	6,691.47	2.72	0.38	856.88	512.69	8,460.02	6,132.46	4.24	1.12	**183.63**	**103.46**	**1,384.82**	**802.17**	**0.20**	**0.13**
(E $\rightarrow$ $_{\mathrm{U}}$RU) + AR	277.93	227.02	1,991.10	1,646.43	0.23	0.07	**162.88**	**76.44**	**1,160.18**	**752.93**	**0.21**	**0.11**	187.40	122.37	1,283.08	921.00	0.21	0.08
$_{\mathrm{U}}$RU	2,850.77	2,301.93	15,645.06	15,051.94	9.65	0.93	2,044.83	1,351.42	10,417.92	8,664.39	4.92	0.57	3,769.08	2,845.18	15,525.91	14,128.42	0.55	0.08

Table 15: Ablation on the first 105 days of the SARS-CoV-2 dataset.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	PyTorch's Hyperparameters						**Literature's Hyperparameters**						ReGENN's **Hyperparameters**					
	MAE	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**
Recurrent Neural Network (RNN)																		
$_{\mathrm{B}}$RU	341.96	160.69	2,004.99	1,108.56	0.12	0.02	1,261.25	1,129.84	4,716.83	4,379.85	0.13	0.05	1,323.09	883.58	5,040.30	4,548.85	0.14	0.05
E → $_{\mathrm{B}}$RU	4,341.20	2,825.80	25,086.49	23,858.93	12.92	0.84	3,126.49	2,291.22	17,483.74	16,935.89	6.56	0.77	3,404.41	1,317.70	20,373.84	14,638.73	6.55	0.55
(E → $_{\mathrm{B}}$RU + $_{\mathrm{B}}$RU) + AR	1,120.61	877.22	12,934.67	12,210.30	3.35	0.36	1,166.00	601.41	13,690.65	11,482.17	3.33	0.17	238.42	90.06	1,582.80	531.20	0.11	0.04
(E → $_{\mathrm{B}}$RU + $_{\mathrm{U}}$RU) + AR	**246.59**	**148.21**	**1,474.80**	**1,032.02**	**0.11**	**0.04**	320.71	171.59	2,169.15	1,011.90	0.10	0.02	227.72	111.45	1,333.91	713.54	0.12	0.03
(E → $_{\mathrm{B}}$RU) + AR	2,190.95	1,807.99	17,646.09	16,804.20	6.54	0.69	**239.77**	**141.32**	**1,556.19**	**959.49**	**0.09**	**0.01**	**226.43**	**90.79**	**1,455.35**	**585.70**	**0.11**	**0.05**
E → $_{\mathrm{U}}$RU	2,947.62	2,982.07	20,010.83	24,909.44	3.52	0.43	2,391.16	1,265.33	9,282.64	5,811.69	0.44	0.17	1,993.13	1,126.18	7,824.99	5,893.62	0.64	0.22
(E → $_{\mathrm{U}}$RU + $_{\mathrm{B}}$RU) + AR	278.32	169.27	1,477.52	1,030.39	0.17	0.04	280.10	135.78	2,033.65	1,390.79	0.10	0.02	261.75	120.82	1,512.66	719.22	0.11	0.03
(E → $_{\mathrm{U}}$RU + $_{\mathrm{U}}$RU) + AR	986.76	718.41	11,505.97	10,866.77	3.21	0.40	262.21	116.82	1,723.81	781.72	0.11	0.03	249.63	124.28	1,494.06	836.21	0.11	0.05
(E → $_{\mathrm{U}}$RU) + AR	981.33	653.20	11,512.57	10,876.46	3.20	0.23	253.41	132.72	1,621.04	846.46	0.13	0.07	269.68	129.55	1,620.35	848.37	0.12	0.04
$_{\mathrm{U}}$RU	997.57	753.72	6,170.68	5,317.17	2.86	0.72	1,374.42	851.34	5,262.59	4,588.21	0.23	0.19	1,344.69	571.12	5,549.24	4,109.40	0.18	0.05
Gated Recurrent Unit (GRU)																		
$_{\mathrm{B}}$RU	1,197.76	614.55	14,024.42	12,312.72	3.48	0.42	3,138.88	2,547.76	20,082.44	20,202.11	6.86	0.79	2,248.76	1,391.69	15,338.81	15,264.58	3.57	0.42
E → $_{\mathrm{B}}$RU	941.01	859.05	5,899.12	6,879.50	0.28	0.10	3,282.37	2,882.56	17,989.68	22,475.37	0.43	0.22	1,693.16	1,403.68	6,452.83	6,431.29	0.24	0.08
(E → $_{\mathrm{B}}$RU + $_{\mathrm{B}}$RU) + AR	260.26	143.14	1,580.15	877.78	0.12	0.04	362.61	165.44	2,560.90	1,559.77	0.55	0.38	245.13	99.07	1,526.87	747.33	0.12	0.04
(E → $_{\mathrm{B}}$RU + $_{\mathrm{U}}$RU) + AR	242.25	118.67	1,471.29	633.96	0.12	0.04	301.90	108.46	1,978.05	808.96	0.18	0.05	272.90	164.68	1,587.65	817.19	0.12	0.04
(E → $_{\mathrm{B}}$RU) + AR	1,064.23	593.75	11,838.02	9,917.30	3.16	0.21	267.71	81.15	2,014.32	663.13	0.24	0.20	251.47	208.25	1,321.24	1,038.05	0.11	0.04
E → $_{\mathrm{U}}$RU	1,141.02	1,270.93	9,951.23	11,855.78	0.39	0.08	2,414.07	1,048.40	9,049.15	4,455.50	0.48	0.10	1,904.25	1,246.60	7,347.94	5,395.31	0.23	0.05
(E → $_{\mathrm{U}}$RU + $_{\mathrm{B}}$RU) + AR	**223.08**	**130.92**	**1,340.96**	**793.38**	**0.13**	**0.07**	247.52	170.20	1,364.31	964.02	0.32	0.07	**234.74**	**138.74**	**1,405.17**	**857.20**	**0.11**	**0.04**
(E → $_{\mathrm{U}}$RU + $_{\mathrm{U}}$RU) + AR	2,950.38	1,927.05	23,175.39	21,293.64	9.89	0.71	**244.72**	**114.03**	**1,519.01**	**749.52**	**0.15**	**0.06**	269.38	100.36	1,605.27	672.70	0.12	0.06
(E → $_{\mathrm{U}}$RU) + AR	2,162.26	1,541.75	18,386.77	18,374.57	6.64	0.44	1,155.48	1,005.94	12,959.38	14,123.98	3.50	0.56	248.97	184.04	1,327.23	1,018.99	0.11	0.04
$_{\mathrm{U}}$RU	1,780.31	1,059.76	17,555.20	14,679.39	6.35	0.73	1,327.66	808.13	5,282.80	4,617.99	0.10	0.03	1,840.21	1,247.34	11,737.09	10,978.18	2.33	0.18
Long Short-Term Memory (LSTM)																		
$_{\mathrm{B}}$RU	1,124.29	522.77	13,528.89	10,943.31	3.30	0.21	2,267.79	1,935.99	14,917.69	15,686.46	3.56	0.45	1,328.14	647.15	5,485.95	4,212.59	0.12	0.04
E → $_{\mathrm{B}}$RU	1,339.27	904.16	7,096.00	4,733.68	0.30	0.15	3,029.54	3,088.49	17,588.19	23,093.53	0.71	0.24	2,013.99	1,002.01	8,203.49	6,971.06	0.27	0.03
(E → $_{\mathrm{B}}$RU + $_{\mathrm{B}}$RU) + AR	1,463.09	1,012.26	14,503.84	13,505.02	3.45	0.21	**229.85**	**135.67**	**1,304.71**	**746.95**	**0.26**	**0.15**	**221.61**	**110.78**	**1,317.11**	**712.53**	**0.12**	**0.03**
(E → $_{\mathrm{B}}$RU + $_{\mathrm{U}}$RU) + AR	1,114.13	875.69	12,659.32	12,044.33	3.32	0.30	253.15	154.09	1,506.05	1,005.71	0.25	0.10	250.02	115.48	1,536.47	733.91	0.12	0.03
(E → $_{\mathrm{B}}$RU) + AR	**230.72**	**66.84**	**1,430.87**	**375.47**	**0.11**	**0.04**	1,163.75	620.16	14,356.48	11,907.68	3.68	0.35	257.26	225.31	1,263.43	1,122.21	0.11	0.04
E → $_{\mathrm{U}}$RU	2,901.71	1,821.21	16,836.92	13,897.86	3.52	0.45	3,828.85	3,702.78	16,994.16	20,532.17	0.43	0.11	3,420.16	2,384.23	14,959.58	16,020.56	0.35	0.08
(E → $_{\mathrm{U}}$RU + $_{\mathrm{B}}$RU) + AR	1,030.73	841.67	10,553.23	11,022.68	3.16	0.22	251.33	93.95	1,592.97	672.39	0.19	0.09	1,016.56	532.07	12,012.26	9,602.11	3.16	0.26
(E → $_{\mathrm{U}}$RU + $_{\mathrm{U}}$RU) + AR	1,143.99	922.78	12,695.98	12,119.30	3.33	0.31	1,083.09	848.17	11,669.35	12,669.18	3.42	0.26	247.24	76.05	1,545.03	440.14	0.11	0.06
(E → $_{\mathrm{U}}$RU) + AR	305.72	199.80	2,147.26	1,840.07	0.12	0.07	243.91	138.62	1,573.45	874.97	0.33	0.30	242.51	106.62	1,505.87	750.02	0.11	0.06
$_{\mathrm{U}}$RU	4,313.01	2,677.42	24,016.09	22,526.52	10.75	1.40	3,370.24	2,602.69	16,814.62	16,794.95	6.01	0.87	3,074.70	2,227.76	16,504.10	15,659.24	4.31	0.44

Table 16: Ablation on the complete SARS-CoV-2 dataset.

Legend: Algorithms with the best performance are in **bold**; besides, we refer to the Transformer Encoder as E, recurrent unit as RU, Bidirectional as B, Unidirectional as U, and Autoregressive as AR.

	PyTorch's Hyperparameters						**Literature's Hyperparameters**						ReGENN's **Hyperparameters**					
	MAE	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**	**MAE**	**± STD**	**MSLE**	**± STD**	**RMSE**	**± STD**
Recurrent Neural Network (RNN)																		
${}_{\mathrm{B}}\mathrm{RU}$	434.18	250.54	2,678.49	2,046.82	0.09	0.05	1,895.13	995.98	7,184.26	5,885.75	0.08	0.03	424.32	200.68	2,819.11	2,021.66	0.07	0.04
$\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}$	3,680.89	2,404.90	25,773.58	22,768.94	7.47	0.80	2,510.11	1,324.54	9,521.55	6,720.47	0.19	0.05	4,161.35	3,028.85	28,755.33	28,712.61	11.08	1.31
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1,533.62	949.53	18,013.32	16,584.35	3.82	0.16	317.13	184.28	1,806.87	1,269.50	0.10	0.07	1,624.44	1,221.57	17,245.24	17,218.81	3.84	0.49
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	**245.06**	**65.03**	**1,359.34**	**462.61**	**0.09**	**0.05**	279.38	115.14	1,569.99	836.28	0.09	0.07	**257.15**	**149.23**	**1,438.72**	**823.36**	**0.09**	**0.06**
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1,386.30	859.67	16,848.20	15,758.33	3.72	0.45	304.52	141.03	2,029.00	1,595.46	0.10	0.04	1,471.49	952.84	17,050.08	15,672.77	3.74	0.34
$\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}$	3,171.99	1,445.00	21,117.23	16,502.10	3.92	0.10	2,692.19	2,133.65	8,934.01	8,459.59	0.29	0.18	1,949.65	1,188.93	16,925.69	16,149.04	3.84	0.41
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	350.75	202.78	1,985.39	1,228.78	0.11	0.03	280.09	142.50	1,669.51	1,096.35	0.10	0.03	372.80	190.41	2,155.36	1,322.94	0.11	0.07
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1,386.73	641.20	17,459.45	14,421.01	3.70	0.30	1,423.25	738.45	17,459.46	14,825.39	3.72	0.39	1,472.05	916.89	16,491.81	15,674.45	3.72	0.26
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1,375.60	988.31	15,987.09	16,378.54	3.69	0.34	**240.34**	**93.62**	**1,386.81**	**832.15**	**0.10**	**0.06**	1,350.35	817.74	16,673.11	15,280.76	3.69	0.33
${}_{\mathrm{U}}\mathrm{RU}$	1,277.24	1,053.33	7,948.96	7,430.65	2.89	0.96	1,954.90	1,154.82	7,310.14	6,297.24	0.21	0.13	1,175.99	768.83	6,716.42	5,337.00	2.08	0.73
Gated Recurrent Unit (GRU)																		
${}_{\mathrm{B}}\mathrm{RU}$	3,167.33	1,458.27	28,834.18	23,983.55	7.80	0.49	3,094.99	2,293.39	18,854.48	19,286.26	3.89	0.31	3,130.47	2,065.13	28,063.07	24,375.98	7.78	0.75
$\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}$	950.11	433.35	5,133.01	2,301.34	0.19	0.14	2,702.04	2,483.15	10,715.26	13,157.31	0.18	0.10	596.77	403.89	3,446.67	2,746.08	0.17	0.16
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	311.09	119.17	1,873.65	1,082.02	0.10	0.07	219.16	106.06	1,276.78	881.66	0.07	0.03	388.93	335.26	1,888.70	1,754.53	0.09	0.03
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	650.06	185.77	3,443.73	1,270.04	0.10	0.04	259.91	149.24	1,597.42	1,273.26	0.11	0.07	366.06	175.38	2,158.32	1,206.14	0.10	0.04
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1,377.91	517.07	17,145.02	14,067.11	3.66	0.10	505.81	582.37	3,916.95	5,297.45	0.11	0.08	1,502.20	1,005.35	16,799.96	14,836.87	3.67	0.39
$\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}$	1,360.35	737.84	7,611.46	4,036.40	0.27	0.11	2,380.63	1,475.81	8,555.18	6,771.74	0.23	0.12	968.83	673.75	5,982.35	5,430.49	0.22	0.07
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	**240.35**	**57.07**	**1,335.96**	**343.59**	**0.08**	**0.04**	245.62	159.01	1,367.82	966.48	0.07	0.03	**208.88**	**119.65**	**1,141.56**	**575.75**	**0.08**	**0.04**
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	3,377.44	1,741.87	25,335.28	22,336.83	7.44	1.07	**216.81**	**107.29**	**1,280.58**	**846.01**	**0.11**	**0.04**	4,052.71	2,625.95	31,384.80	28,694.03	11.30	1.27
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1,592.62	895.15	18,207.40	15,612.58	3.77	0.39	248.10	166.88	1,508.53	1,184.28	0.09	0.04	2,852.28	1,735.94	25,869.18	23,701.61	7.60	0.59
${}_{\mathrm{U}}\mathrm{RU}$	2,319.57	1,793.72	21,536.86	21,418.12	6.26	0.55	1,875.92	1,328.11	6,933.49	6,460.49	0.11	0.08	1,825.66	650.78	20,400.04	15,797.17	4.95	0.20
Long Short-Term Memory (LSTM)																		
${}_{\mathrm{B}}\mathrm{RU}$	3,000.14	1,954.01	25,989.65	24,313.81	7.65	0.78	3,042.70	1,987.48	18,539.83	17,591.06	3.80	0.41	1,800.23	928.80	20,264.27	17,544.01	3.95	0.15
$\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}$	619.31	356.13	3,094.25	1,878.42	0.17	0.11	2,309.35	1,269.98	7,953.65	6,114.91	0.23	0.15	704.80	343.17	4,088.05	2,893.91	0.20	0.10
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	**216.01**	**89.11**	**1,146.94**	**465.90**	**0.10**	**0.07**	256.32	59.69	1,412.55	487.27	0.12	0.07	1,598.03	884.02	19,056.28	16,854.23	3.89	0.17
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	2,863.94	1,345.36	27,905.87	22,477.61	7.69	0.62	**244.81**	**128.06**	**1,233.11**	**666.28**	**0.18**	**0.21**	4,064.52	2,702.07	31,196.10	28,670.34	11.30	1.01
$(\mathrm{E} \rightarrow {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	4,060.62	2,655.67	31,538.83	28,960.85	11.35	0.99	1,554.31	1,416.54	17,417.78	18,060.72	3.90	0.51	**211.93**	**83.35**	**1,181.31**	**570.10**	**0.08**	**0.06**
$\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}$	3,199.26	2,436.81	12,581.64	11,769.69	0.26	0.07	3,169.42	2,063.50	12,461.42	11,674.12	0.17	0.05	2,778.55	1,112.16	12,830.03	10,175.60	0.21	0.06
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{B}}\mathrm{RU}) + \mathrm{AR}$	1,351.42	644.34	17,065.24	14,155.37	3.67	0.30	1,375.17	1,053.14	15,406.60	15,929.81	3.69	0.34	1,380.99	864.20	16,370.41	14,977.34	3.66	0.32
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU} + {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	1,431.78	946.29	17,235.49	16,431.19	3.79	0.38	285.02	121.63	1,802.48	1,118.25	0.11	0.06	1,713.72	1,025.91	18,531.68	15,888.72	3.81	0.39
$(\mathrm{E} \rightarrow {}_{\mathrm{U}}\mathrm{RU}) + \mathrm{AR}$	222.06	85.96	1,265.74	642.42	0.08	0.03	265.80	112.59	1,437.64	686.42	0.14	0.09	260.66	215.01	1,513.92	1,430.67	0.10	0.03
${}_{\mathrm{U}}\mathrm{RU}$	4,943.62	2,892.09	28,696.47	24,921.66	7.59	0.30	4,942.10	3,425.34	25,691.11	25,415.68	7.38	0.97	3,723.88	2,536.94	21,022.60	20,012.26	2.83	0.31