CAIO FABRICIO DEBERALDINI NETTO

PHYSICS-INFORMED GRAPH REPRESENTATION LEARNING FOR OCEAN DYNAMICS

São Paulo
2023

CAIO FABRICIO DEBERALDINI NETTO

PHYSICS-INFORMED GRAPH REPRESENTATION LEARNING FOR OCEAN DYNAMICS

Corrected Version

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2023

CAIO FABRICIO DEBERALDINI NETTO

PHYSICS-INFORMED GRAPH REPRESENTATION LEARNING FOR OCEAN DYNAMICS

Corrected Version

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Fabio Gagliardi Cozman

São Paulo
2023

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 27 de Julho de 2023

Assinatura do autor: 

Assinatura do orientador: 

Catalogação-na-publicação

Netto, Caio Fabricio Deberaldini
Physics-Informed Graph Representation Learning for Ocean Dynamics / C. F. D. Netto -- versão corr. -- São Paulo, 2023.
73 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Inteligência Artificial 2.Redes Neurais de Grafos 3.Predição 4.Sistemas Dinâmicos I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

ACKNOWLEDGMENTS

Accomplishing this work was a challenging but rewarding task. And as I look back on this journey, I deeply recognize and express my immense gratitude to the many individuals who have been part of it. However, listing each individually would make this document excessively lengthy and the acknowledgments tedious for the reader. Therefore, I would like to dedicate a special recognition to those who have closely accompanied this evolution:

To my parents, Hilda and Nivaldo, my sincere thanks for teaching me the value of education and supporting me in another dream. You have been the best teachers in life that I could ever have. Your teachings about education as a tool for transforming realities, as well as the importance of hard and ethical work, and perseverance as a virtue, will never be forgotten by me. They are an integral part of what I aspire to transmit to others, whatever humble legacy I may leave. My priority will always be to try to repay you for all the dedication you have given me.

To my partner, Kissya, and my brother, Markus, I want to thank you for always being by my side, and believing in my dreams, even when they seemed unclear. Know that this feeling is mutual. I will support you in your dreams, cheer for you, and celebrate your achievements.

To my advisor, Prof. Dr. Fabio Cozman, my heartfelt gratitude for your guidance throughout these years. Under your mentorship, I have learned the true meaning of science and how to conduct high-level research. I feel prepared for the following academic challenges and dedicate a significant part of the knowledge I have gained during this journey to you. Thank you for your friendship, patience, and understanding, especially during the toughest moments we have faced in recent years.

To Itaú-Unibanco, I express my gratitude for the financial support provided through the *Programa de Bolsas Itaú* – PBI.

Lastly, I would like to extend my appreciation to all the faculty and staff members of the department who have directly or indirectly contributed to the completion of this work and my ethical and professional development.

"To achieve importance in a field of knowledge, talent must be disciplined. Understanding that talent is an enabler is accepting that this is your main struggle."

ABSTRACT

The perception of environmental elements surrounding us and how they affect our physical integrity leads us to design models of the real world, aiming to anticipate or mitigate the effects of disadvantageous situations for human life caused by events both in space and time. On that matter, we have had huge advances in physical sciences modeling, especially in environmental and engineering domains. One particular domain of interest is that of the oceans. Typically, researchers resort to numerical methods to understand and predict ocean dynamics in order to master its phenomena. Nonetheless, such methods reach limitations in a variety of scenarios. For instance, if the topographic map is complex (thus making the mesh generation also complex), the underlying processes are partially or completely unknown, or we are dealing with high-dimensional problems, numerical discretizations of the governing equations may not be suitable. Machine learning has been proven to be an alternative when oceanic data is available. The best current models, deep neural networks, require big data to perform well in prediction tasks, and given that ocean dynamics spans both space and time, there is the real challenge of being capable of relating information in different dimensions when modeling these systems with learning algorithms. Currently, the lack of big data in scientific problems is tackled by combining such neural networks with physical prior knowledge in the so-called physics-informed learning paradigm. On the other hand, graph neural networks from the field of graph representation learning have shown far excellent results when dealing with relational problems such as the prediction of spatiotemporal phenomena. However, the intersection between those two research fields is not well explored in general, and even less so in real-world problems. In this work we focus our efforts on bonding the fields of physics-informed machine learning and graph representation learning, devising a deep learning method for the task of forecasting oceanic variables, such as water sea surface height and current velocity. Experiments have been run within real-world data from two different locations in the southeast coastal region of Brazil, the Sepetiba/Ilha Grande Bay and the Santos-São Vicente-Bertioga Estuarine System. Our model can exploit both temporal and spatial inductive biases by joining state-of-the-art sequence and relational models and physical knowledge from a numerical model, with significant improvements in data efficiency and prediction accuracy.

Keywords – physical systems, ocean dynamics, physics-informed machine learning, graph representation learning, graph neural networks, forecasting.

RESUMO

A percepção dos elementos ambientais que nos cercam e como eles afetam nossa integridade física nos levam a projetar modelos do mundo real, visando antecipar ou mitigar os efeitos de situações desvantajosas para a vida humana, causadas por eventos tanto no espaço quanto no tempo. Nesse sentido, tivemos grandes avanços na modelagem das ciências físicas, principalmente nos domínios ambiental e de engenharia. Um domínio particular de interesse é o dos oceanos. Normalmente, os pesquisadores recorrem a métodos numéricos para entender e prever a dinâmica dos oceanos, a fim de dominar seus fenômenos. No entanto, tais métodos têm suas limitações em diversos cenários. Por exemplo, se o mapa topográfico é complexo (tornando a geração da malha também complexa), os processos fundamentais são parcial ou completamente desconhecidos, ou estamos lidando com problemas de alta dimensão, discretizações numéricas das equações governantes podem não ser adequadas. Aprendizado de máquina se mostra uma alternativa interessante quando dados oceânicos estão disponíveis. Mesmo assim, os melhores modelos, redes neurais profundas, exigem grandes massas de dados para desempenharem bem em tarefas de predição; dado que a dinâmica dos oceanos se estende tanto no espaço quanto no tempo, há um desafio real para conseguir relacionar informações em diferentes dimensões, ao modelar esses sistemas com algoritmos de aprendizado. Atualmente, a falta de grandes bases de dados em problemas científicos é abordada combinando redes neurais com conhecimento físico prévio, no chamado paradigma de aprendizado informado pela física. Por outro lado, as redes neurais de grafos, inseridas na área de aprendizado de representação de grafos, tem apresentado excelentes resultados ao lidar com problemas relacionais, como a previsão de fenômenos espaço-temporais. No entanto, a interseção entre esses dois campos de pesquisa não é bem explorada em geral, e muito menos em problemas do mundo real. Neste trabalho concentramos nossos esforços em unir os campos de aprendizado de máquina informado por física e aprendizado de representação gráfica, desenvolvendo um método de aprendizado profundo para a tarefa de predição de variáveis oceânicas, como altura da superfície do mar e velocidade da corrente. Experimentos foram conduzidos com dados reais de dois locais diferentes da região costeira do sudeste do Brasil, a Baía de Sepetiba/Ilha Grande e o Sistema Estuarino de Santos-São Vicente-Bertioga. Nosso modelo tem sido capaz de explorar vieses indutivos temporais e espaciais ao unir modelos sequenciais estado-da-arte, modelos relacionais e conhecimento físico de um modelo numérico, com melhorias significativas na eficiência do uso dos dados e acurácia de predição.

Palavras-Chave – sistemas físicos, dinâmica de oceanos, aprendizado de máquina informado por física, aprendizado de representação de grafos, redes neurais de grafos, predição.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The last ten years have witnessed major changes in the field of machine learning. There has been a significant advance in techniques that employ local optimization to fit large models from huge datasets. On the other hand, advances in processing capacity and data collection allowed such models and data to be effectively processed. Topics such as machine translation (BAHDANAU; CHO; BENGIO, 2014), computer vision (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; REDMON et al., 2016), and natural language processing (VASWANI et al., 2017) have been greatly affected.

Machine learning techniques, specifically deep learning ones, have seen an increase in interest in applying them to advance scientific discovery in various domains. For instance, researchers have made remarkable advances in fields of science as varied as physics (RAISSI; PERDIKARIS; KARNIADAKIS, 2019; CRANMER et al., 2020), biological sciences (STOKES et al., 2020; JUMPER et al., 2021), and material engineering (RACCUGLIA et al., 2016), applying these learning algorithms to real problems.

In particular, environmental and engineering domains, where first principle models dominated until recently, have benefited from those advances, especially in scenarios where domain knowledge is unknown or incomplete, problem complexity is enormous, and there is sufficient data available. One such field is that of physical systems (KUTZ, 2017; BERGEN et al., 2019; REICHSTEIN et al., 2019). A myriad of physical systems span both temporal and spatial dimensions, and much research focuses its efforts on approaching one of these dimensions with greater emphasis while modeling a specific problem with machine learning.

The parallel progress in the area of Graph Representation Learning (GRL) (HAMILTON; YING; LESKOVEC, 2017), particularly due to the development of practical and theoretical foundations for Graph Neural Networks (GNNs) (GORI; MONFARDINI; SCARSELLI, 2005; SCARSELLI et al., 2009; BRONSTEIN et al., 2017), has drawn attention to use them in scenarios where spatiotemporal relations are paramount. In physical systems, GNNs are used to model the domain of application in the form of

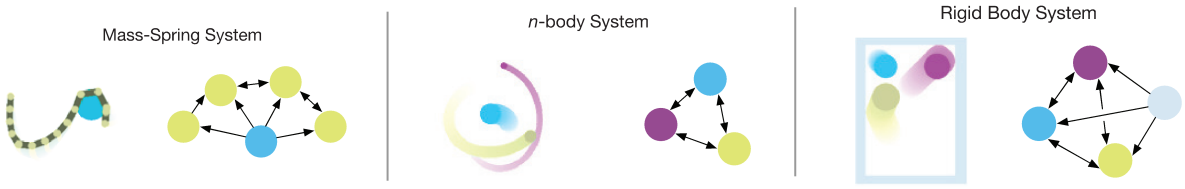


Figure 1: **Examples of physical systems modeled as a graph.** From left to right: (1) Mass-Spring System, (2) $n-$Body System, and (3) Rigid Body System. Adapted from (BATTAGLIA et al., 2018).

a graph, with the domain's entities/objects as nodes and their relations represented as edges between them. Also, the system's dynamics are encoded as temporal sequences inputs to the GNN. With that framework, GNNs can learn powerful representations of the physical systems and exploit the domain's spatiotemporal biases. Figure 1 shows some examples of those systems modeled in the form of a graph.

Despite those advances in data collection and the creation of new machine learning models and techniques, purely data-driven approaches highly depend on data availability. More crucially, they are prone to have weak extrapolation power of the knowledge learned to outside the training domain. In the context of this work, in physical systems, this poor generalization performance is easily seen due to physical inconsistencies and theoretical implausibility.

Researchers have, therefore, devoted efforts to integrating domain knowledge as prior information, in the form of explicit knowledge, hard constraints, or inductive biases, into learning models. One can imagine a range of scenarios for physical systems problems, with respect to those two variables, i.e., physical knowledge and data availability. Figure 2 presents how this relationship is organized. As a result, research on physical systems has experienced, recently, a fast-growing pace in the adoption of physics-informed machine learning[1] models (WILLARD et al., 2021; KARNIADAKIS et al., 2021) to solve diverse tasks (BRUNTON; PROCTOR; KUTZ, 2016; BATTAGLIA et al., 2016; RAISSI; PERDIKARIS; KARNIADAKIS, 2019; SANCHEZ-GONZALEZ et al., 2020).

In this work, physics-informed machine learning refers to algorithms enhanced by (human) prior knowledge derived from our empirical, physical, or mathematical conceptualization of the world (KARNIADAKIS et al., 2021). These algorithms are robust to missing or noisy data (most commonly in real-world scenarios), interpretable, given the

[1]It is worth mentioning that this field has been labeled with other names, such as physics-guided machine learning, or physics-induced machine learning. Throughout this work, these terms are used interchangeably.

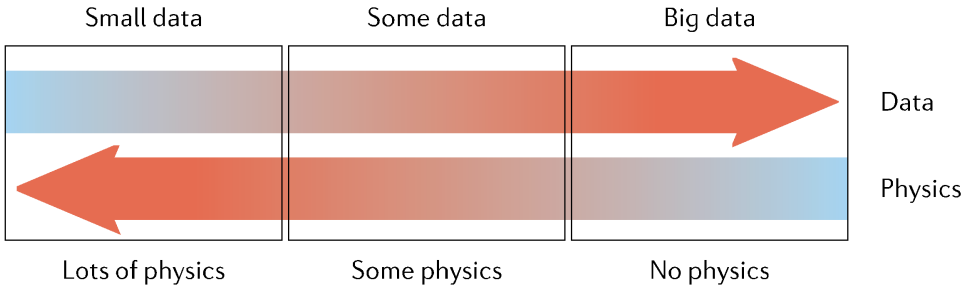


Figure 2: **Spectrum into which a physical problem can be framed.** On the leftmost side, the small data regime, we are dealing with the scenario where all the physics of the problem is known, and sufficient data is available, i.e., initial and boundary conditions, and the equations' coefficients. As expected, in the majority of the cases, researchers may know little about the physics of the problem, and some observational data is available, which is represented as the middle case scenario. Inversely, on the rightmost side, we have little to no knowledge about the physics, though there is plenty of data available — the big data regime. In this case, physics may be inferred from the data with techniques like machine learning ones. Extracted from (KARNIADAKIS et al., 2021).

embedded domain knowledge, and capable of providing accurate and consistent physics predictions, thus, more efficient to generalize to out-of-distribution data.

The goal of this work is to develop a physics-informed graph neural network to model ocean dynamics' variables, such as water sea surface height and current velocity. We aim to build a model capable of (1) dealing with temporally and spatially distributed phenomena, while (2) being robust to missing or noisy data, (3) learning representations grounded in physics, and (4) producing accurate and consistent predictions with the physical knowledge of the domain. More generally, we want to devise a model that can be extrapolated to any oceanic area in which data and physical prior knowledge are available, and any of those characteristics are present, with guarantees that it also works in real-world conditions.

To achieve that, we first reduced our scope to the southeast coastal region of Brazil, for which we have oceanic data available, and divided the project timeline into two moments. In the first one, we focused our efforts on gaining theoretical and practical experience with modeling and predicting oceanic variables with GNNs, in a purely data-driven setting, where we are only concerned about goal (1) above. We have built and compared our model with classical statistical learning and state-of-the-art models, with respect to the prediction of water current velocity in the region of Sepetiba/Ilha Grande Bay, located in the state of Rio de Janeiro.

In a second moment, we addressed a more complex problem: modeling the Port of

Santos Bay. Characteristics (1) and (2) explained above are present in this problem, and given that we do have a physical model of that region, we are able to enhance the knowledge learned by our GNN. As we had already made our proof of concept in the first part of the project, we focused on finding the best architecture by extensive empirical design in this stage.

So far, we have gathered robust evidence of the benefits of relational models, such as GNNs, in scenarios that require modeling of multiple dimensions (space and time, in the case of the oceanic domain). We have found that GRL models are data efficient (require fewer data), can handle noise and missing data, learn powerful representations and take advantage of information sharing, and, ultimately, perform excellently on real problems.

In short, this dissertation is an effort to contribute to the advance of physical systems modeling with both first principle and machine learning models, standing at the intersection between physics-informed machine learning and graph representation learning. This text is structured as follows: in Chapter 2, we discuss the necessary background information about Machine Learning (ML), Deep Learning (DL), priors for DL models, the Graph Neural Network (GNN) model, and how to leverage learning algorithms with physical priors. Following that, i.e. Chapter 3, we present the related works in which we developed our methods on top of and contributed to, putting our work in context. Chapters 4 and 5 present the real problems we seek to solve, the methods developed to address them, the reasoning behind our hypothesis, the experimental results we attained, and practical and theoretical limitations. Finally, in Chapter 6, we summarize our findings and discuss about future work that could improve our results.

2 BACKGROUND

This chapter covers the fundamental background information that is relevant to understand this work. It is worth emphasizing that the clarification of the concepts in here are by no means a definitive and/or complete exposition of them, but cover their essence to facilitate understanding of the remaining text.

2.1 Machine Learning

In a broad perspective, machine learning is an area of the AI research field in which researchers are concerned about developing systems capable of learning how to accomplish tasks using data, without being explicitly hard-coded. Mitchell (1997) states that an algorithm with learning capabilities is a computer program that is able to improve its **performance** in a class of **tasks** in light of **experience**. Thus, one can define, naively, a machine learning algorithm as data responsive, i.e., it improves its performance when more data is presented.

A machine learning algorithm can often be seen as a mapping from a (data)set of observations $X \in \mathbb{R}^{m \times n}$ to the task's domain $y \in \mathbb{R}^d$, i.e., a functional mapping f: $\mathbb{R}^{m \times n} \rightarrow \mathbb{R}^d$, where each one of the m observations is represented as a real-valued vector $\mathbf{x} \in \mathbb{R}^n$, composed of n characteristics (features) that may be important to describe and help in the task. Therefore, the challenge for a learning algorithm is to find out inductively the function that best represents the data-generating process, which is constrained by the task's objective. At first sight, it seems to be an ill-posed problem, given the infinite possibilities one can define f.

Yet, ML is now a successful paradigm to a wide range of important problems (POMERLEAU, 1988; LEWIS; RINGUETTE, 1994; CORTES; VAPNIK, 1995). The reason why machine learning works well on those problems is because inductive biases are imposed on the function being searched, thus limiting the search space. For example, one may know beforehand that the problem's outcome variable is linearly dependent on features,

so the function f must be linear, or that the learned function f is 1-Lipschitz continuous, i.e., little perturbations in the input won't affect the outcome. These hypotheses (or a combination of them) are what define a model, which together with sufficient data, is capable of solving a myriad of problems.

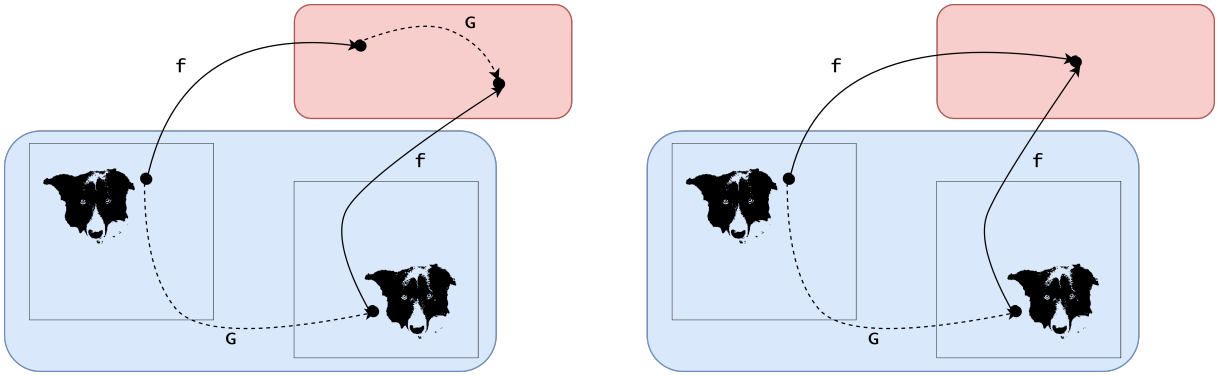
However, machine learning is not only concerned about performing well on observed data but also with generalizing to distributions outside the training data. If one is tackling a narrow task, simple models like linear regression will serve just fine. But, when confronted with human-relevant problems, such as speech recognition, object detection, natural language understanding, or modeling environmental systems, those models fail.

Those problems (and many others) are high-dimensional ones, suffering from a phenomenon known as the 'curse of dimensionality'. In order to capture patterns in such high-dimensional data, it is necessary to use more complex models. Nonetheless, complex machine learning models are prone to have poor performance on data not seen before. So, in the urge to solve the weak generalization capacity of the model, one would seek more data. But, the problem is that the amount of training data needed grows exponentially with its dimension and we end up in a vicious cycle.

2.1.1 Deep Learning

The previous challenge was the main driver for researchers to seek more powerful models that could scale to high dimensions, and neural networks have stood over other classical machine learning models. A few reasons for that are their flexibility and the theoretical guarantees that (multilayer) neural networks are able to approximate any continuous multivariate function to a desired degree of accuracy (CYBENKO, 1989; HORNIK, 1991). A paradigm shift – as defined by Kuhn (1962) – was the advance of computational geometry and theoretical efforts on neural networks through this lens. The search for and the injection of strong and regular geometric properties into neural network architectures, constructing powerful inductive biases in association with deeper models was the game changer that revolutionized the field of machine learning. It is worth citing the (nowadays) criticized work of Minsky and Papert (1969) for their introduction of those theories in the study of neural networks, especially theoretical progress with respect to group invariance properties that neural networks display.

A popular deep neural network architecture with those geometric properties is the Convolutional Neural Networks (CNNs). Authors of seminal papers, Fukushima (1980) and LeCun et al. (1989), respectively with the Neocognitron and LeNet model, established



$f(Gx) = Gf(x)$ $\qquad$ $f(Gx) = f(x)$

Figure 3: **Equivariant and invariant translation symmetries.** Given a translation operation G from a collection S ($G \in S$) of symmetry translation operations, a function f is (left) translation equivariant in S if $f(Gx) = Gf(x)$, and (right) translation invariant in S if $f(Gx) = f(x)$. In this example, a dog pattern is shifted in the image grid, and the corresponding function (equivariant/invariant) is applied to the image.

the foundations for recent deep neural network architectures, based on the work of Hubel and Wiesel (1959, 1962). The gist of their work resided on the concepts of compositionality and hierarchy, inspired by the biological evidence found by the latter researchers, and the introduction of geometric priors to the neural network's architecture.

Both Fukushima (1980) and LeCun et al. (1989) hypothesized that patterns in an image were defined by local information. However, this information can appear anywhere on the image. Therefore, the desired function f must be robust to translation transformations. That is, f must have **translation invariance**. In other words, the output of the model should not depend on the position of the aimed pattern in the image. Figure 3 explains mathematically and visually the geometric properties of equivariance and invariance to the group of translations.

To summarize, if one knows what kinds of regularities the function to be learned must observe, this knowledge can be used to constrain the search space.

2.1.2 Geometric Deep Learning

Many tasks of interest can be narrowed to prior regularities that come from low-dimensional structures of the physical world, such as exposed through CNNs. Inspired by that, researchers have recently contributed to the emergence of a new field known as Geometric Deep Learning (GDL), which focuses on unifying these regularities into a framework of geometric principles.

Bronstein et al. (2021) show how state-of-the-art models, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Transformers, can be devised through the lens of symmetry groups. Another model that has had significant advances in recent years due to theoretical advances in the GDL field, and which is of particular interest for this work, are Graph Neural Networks.

2.2 Graph Neural Networks

Graphs are structures that model systems of relations and interactions, used in many fields of science, from biology and physics to sociology.

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is defined as set of *nodes* $\mathcal{V}$ and *edges* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ between pairs of nodes. We assume that each node $v \in \mathcal{V}$ is embedded with a $d-$dimensional node feature $\mathbf{x}_v \in \mathbb{R}^d$. One common example is the modeling of social networks as graphs, where users and their interactions are represented, respectively, as nodes and edges of the graph, and user information, such as name, age, birth place and profile picture, are modeled as node features.

A key structural property of graphs is that the order of the nodes $v \in \mathcal{V}$ is not assumed to be in any particular configuration. In other words, if one performs any operation on graphs, it is desirable that it does not depend on the ordering of the nodes. Thus, functions applied on graphs should respect the property of **permutation invariance**, so any two *isomorphic* graphs have the same outcome from that function.

To formalize that notion, assume a generic graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with n nodes endowed with node features X, and $\mathcal{E} \neq \varnothing$. Node connectivity is represented with a $n \times n$ *adjacency* matrix A defined as follows:

$$a_{i,j} = \begin{cases} 1, & (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, following the definitions of invariant and equivariant functions described previously, given a permutation operation P, a function F applied[1] to this graph is *permutation invariant* if it satisfies:

$$F(PX, PAP^T) = F(X, A),$$

[1]Given that F is applied to a graph, it depends on both the node features X, and the adjacency matrix A.

and *permutation equivariant*, if it satisfies:

$$F(PX, PAP^T) = PF(X, A).$$

Due to the invariance constraint, it is reasonable to define the functional mapping in terms of node-wise operations, given that (1) ordering is not relevant, and that (2) each node depends only on a local subset of nodes in $\mathcal{V}$, named as the node's *neighbourhood*. A neighbourhood of a node v is defined as:

$$\mathcal{N}_v = \{u : (u, v) \text{ or } (v, u) \in \mathcal{E}\}.$$

Moreover, the neighbourhood also defines the *neighbourhood features*[2] of node v:

$$X_{\mathcal{N}_v} = \{\{\mathbf{x}_u : u \in \mathcal{N}_v\}\}.$$

Equipped with that knowledge, such a function F is constructed applying a permutation invariant ϕ function[3] to each node locally, i.e., ϕ is applied taking into consideration each node feature vector and its neighbourhood features:

$$F(X, A) = \begin{bmatrix} — & \phi(\mathbf{x}_1, X_{\mathcal{N}_1}) & — \\ — & \phi(\mathbf{x}_2, X_{\mathcal{N}_2}) & — \\ & \vdots & \\ — & \phi(\mathbf{x}_n, X_{\mathcal{N}_n}) & — \end{bmatrix}.$$

Thus, a Graph Neural Network (GNN) can be specified as a function F devised above, with stacked layers computing, one after another, a latent representation of each entity of the graph. Figures 4 and 5 represent how each node feature is updated through the permutation invariant local function ϕ, and the blueprint of a GNN, respectively. In the field jargon, ϕ is also known as the *diffusion* or *message-passing* function and F is defined as the *GNN Layer*. Each GNN model has a particular F and ϕ functions. Those are parameterized differentiable functions, which enable gradient-descent algorithms for optimization.

We have used the definition of a graph considering nodes as the only entities with potential features. However, in the most general definition, each one of the graph's entities, e.g. its edges, or even the graph itself, may possibly have attributes. So, the characterization of the functions explained above will vary depending on which entity has features

[2]The *neighbourhood features* is a *multiset* $\{\{...\}\}$ because the features of any two different nodes can have the same value.

[3]Bronstein et al. (2021) prove that if ϕ is permutation *invariant*, then F is permutation *equivariant*.

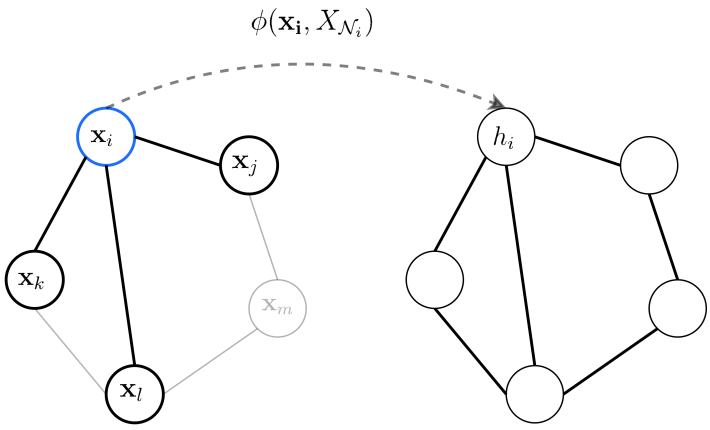


Figure 4: **Example of graph and diffusion function applied to it.** (Left) Local function ϕ applied to node n_i (blue) and its neighbourhood $\mathcal{N}_i$. (Right) Outcome of the message-passing applied to node n_i, which results in the new embedding h_i.

in it.

In this work, we use two complementary definitions to characterize GNNs' functions. The most general of them, inspired by (BATTAGLIA et al., 2018), a *GNN Layer* has functions associated with all its entities, i.e., nodes, edges, and global attributes:

$$
\begin{aligned}
e'_k &= \phi^e(e_k, \mathbf{x}_{r_k}, \mathbf{x}_{s_k}, u), & \bar{e}'_i &= \rho^{e\to v}(E'_i),\\
\mathbf{x}'_i &= \phi^v(\bar{e}'_i, \mathbf{x}_i, u), & \bar{e}' &= \rho^{e\to u}(E'),\\
u' &= \phi^u(\bar{e}', \bar{\mathbf{x}}', u), & \bar{\mathbf{x}}' &= \rho^{v\to u}(V'),
\end{aligned}
$$

where updated features are primed in the above definitions. Nodes, edges, and global attributes, respectively $\mathbf{x}_i$, e_k, and u, are updated by learnable $\phi^{\square}$ functions. Moreover, edges are aggregated by permutation invariant $\rho^{\square}$ functions. Instances of such permutation invariant functions are the *mean*, *median*, or *sum*. The same (permutation invariant) aggregation is made to update nodes that are pointed by edges or by the global attributes. This is also true when one aggregates node attributes for global features' updating. That is the scheme we employ in Chapter 4.

Nonetheless, when dealing with a graph with only node features, as it is presented in Chapter 5, the above equations can be compressed following the message-passing mechanism defined in (GILMER et al., 2017). This mechanism can be summarized by the following equation:

$$
\mathbf{x}'_i = \phi\left(\mathbf{x}_i, X_{\mathcal{N}_i}\right) = \phi^v\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} \phi^e\left(\mathbf{x}_i, \mathbf{x}_j\right)\right), \tag{2.1}
$$

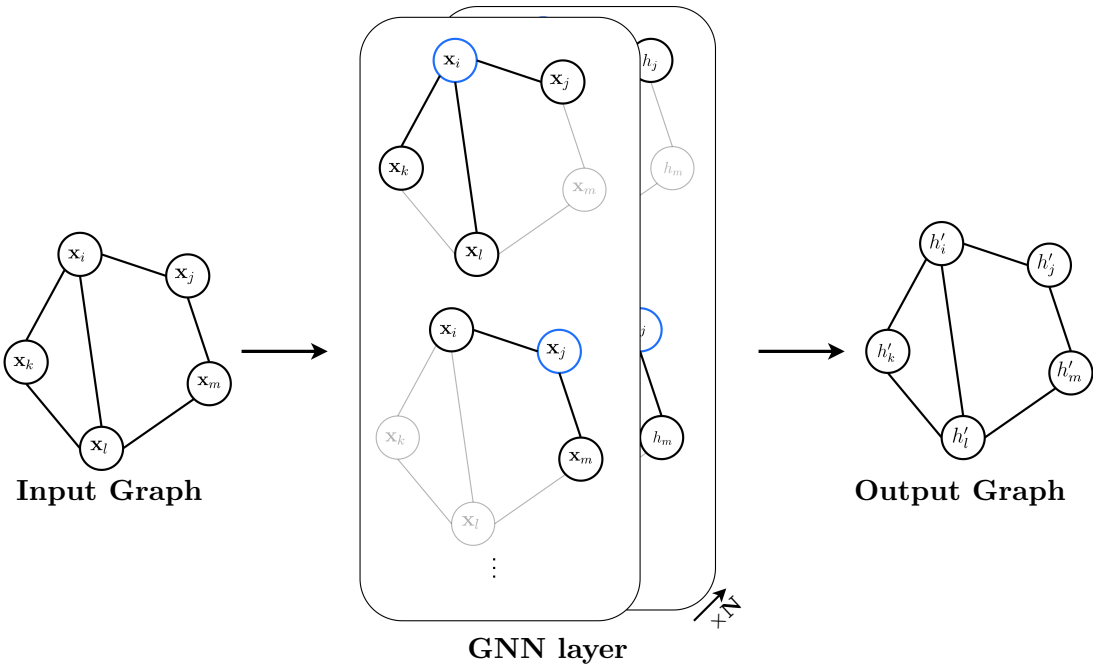


Figure 5: **Graph Neural Network blueprint.** Example of how a GNN learns new node representations $h'_{\square}$. Here, the diffusion function is being applied only on the nodes of the graph. However, it is noteworthy that an embedding layer can be applied in every graph's entities.

where $\mathbf{x}_i$ is the hidden representation (or node features) of a node i, ϕ^e is a differentiable *message* function that constructs a message to be propagated based on both sender ($\mathbf{x}_j$) and receiver ($\mathbf{x}_i$) nodes, linked through an edge, $\bigoplus$ denotes a non-parametric, permutation invariant *aggregation* function that gathers all messages from neighbour nodes, and ϕ^v is a differentiable *update function*, that updates the node's hidden representation. Each $\phi^{\square}$ is parameterized by a set of learnable parameters that are iteratively updated during the training process, as shown in Figure 5. Examples of such functions would be MLPs.

2.3 Physics-Informed Learning

As previously exposed, a physics-informed machine learning algorithm is an algorithm enhanced by empirical, physical, or mathematical models of the world. Concretely, however, that improvement in learning algorithms informed through physics is accomplished introducing any combination of relevant observational, inductive, or learning biases into the learned model. This process must be able to shift the learning algorithm towards physically plausible solutions, compared to free-prior solutions.

Observational biases are introduced into the model via data that captures the

underlying physics of the addressed problem, either through a natural data generating process or augmentation procedures. Given that the model will learn a functional mapping accordingly to the data, it will reflect the physical structures embodied in the data.

On the other hand, **inductive biases** are associated with the insertion of prior assumptions into the model's architecture, such that the learned function is guaranteed to satisfy certain physical laws. One successful example is the convolution operation formerly detailed in the discussion of CNNs. The physical information mostly accounts for mathematical constraints through symmetry groups, s.a. translation, permutation, and rotation.

Lastly, **learning biases** are introduced in learning algorithms by adjusting the loss function to attain some behaviour. An example of learning biases are regularization methods, s.a. $L1$ and $L2$ regularization, where a model is penalized when its solutions diverge from those allowed by the imposed constraint. An example of physics-informed learning algorithm that fits in this concept is the *Physics-Informed Neural Network* model (RAISSI; PERDIKARIS; KARNIADAKIS, 2019).

Again, it is noteworthy that those different ways of biasing a learning algorithm towards physically consistent results are not mutually exclusive. In fact, many recent breakthroughs in AI were only possible through a clever and engineered combination of those approaches, producing powerful hybrid machine learning models physically-informed.

3 RELATED WORK

This work spans multiple fields and builds on similar previous works. However, as it will be explained in the following sections, few works in the literature to date, to the best of our knowledge, have investigated the relationship between graph learning models and models guided by the domain's physical knowledge. In ocean-related domains, a few studies we have found differ from ours in the use of simulated data, resulting in applications not compatible with real-world scenarios.

We first lay out, in Section 3.1, related work that uses learning algorithms to tackle physical systems' prediction tasks. The efforts presented in this section ranges from classical machine learning methods for different domains, narrowing down to the modeling of physical systems through graphs processed by GNNs. After that, Section 3.2 gives an overview of the literature on learning algorithms guided through physics, with applications to graph-structured tasks. Special focus is given to works on domains that are mostly related to the oceanic domain. Lastly, in Section 3.3 some key research gaps of those proposals, which our work seeks to solve, are discussed.

3.1 Machine Learning on Physical Systems

As we previously discussed, machine learning has been adopted recently in various fields of science, or even replacing traditional first-principle models. Even though one would take a long time describing the space of ML applications, it is worth citing relevant disciplines encompassed in the field of physical systems that are important to this work. For instance, ML has had success in physical systems problems from fluid dynamics prediction (XIAO et al., 2019; BRUNTON; PROCTOR; KUTZ, 2016), earth systems and climate science simulation (KRASNOPOLSKY; FOX-RABINOVITZ, 2006; O'GORMAN; DWYER, 2018), and particle interactions (GLIELMO; SOLLICH; VITA, 2017).

In the context of this work, classical ML methods are used as surrogate models to

predict physical systems' dynamics. Models such as Support-Vector Machines (SVMs), Gaussian Processes (GPs), and even shallow neural networks replace appropriate modules of numerical simulators to speed up simulations, while preserving accurate results. For example, in (XIAO et al., 2019) the authors make use of GPs as a Reduced Order Model (ROM) to predict turbulence flows, given data from large scale turbulence fluctuation simulations. In another work, O'Gorman and Dwyer (2018) use random forests as a parameterization of convection terms, which are informative to predict extreme events, however, difficult to model.

Deep learning has helped accelerate the adoption of learning algorithms for system dynamics prediction. As similar problems to those described above tend to be high-dimensional, DL can outperform both classical ML and numerical methods, learning powerful representations, while keeping its flexibility. For deep learning approaches, researchers resort to different inputs and architectures in order to model system dynamics accurately. For instance, Ravuri et al. (2021) propose a generative model – specifically, Generative Adversarial Networks (GANs) – for the task of precipitation nowcasting. The deep learning model is fed with high-resolution radar data and learns to forecast the amount, timing, and location of rainfall at short time period. Numerical weather prediction systems, for instance, have excellent performance in forecasting several days of world-scale weather, some argue. Nonetheless, though powerful, these numerical methods struggle when generating high-resolution predictions for short lead time intervals (TOTH; KALNAY, 1997), i.e., nowcasting.

When dealing with physical systems where interactions and relations are paramount, graph-based learning models excel. For instance, in (BATTAGLIA et al., 2016) the authors present their Interaction Network (IN), which is a graph-based model that takes graphs representing complex object-interaction systems as inputs, diffuse object-level and relation information through the graph with a similar mechanism as message-passing GNNs (GILMER et al., 2017), and predicts the system dynamics. Relation systems such as n-body and rigid-body problems have had their dynamics predicted accurately within this framework, even in settings with distinct number of objects or relations, and time-steps to predict.

Also in the context of modeling physical systems through GRL (HAMILTON; YING; LESKOVEC, 2017) models, Sanchez-Gonzalez et al. (2020) developed a learning framework that can simulate various physical systems, such as fluid, rigid and non-rigid solid materials, interacting (or not) with each other. Their Graph Network-based Simulator (GNS) framework models the physical domain of interest with (discrete) particles as nodes

in a graph, each one of them connected with a finite neighbourhood of nodes. The nodes have feature vectors containing, e.g., the particles properties (mass, stiffness) or their dynamics (velocity, acceleration), as well as the edges. Graph information is propagated through message-passing mechanism (GILMER et al., 2017; BATTAGLIA et al., 2018). GNS is capable of predicting system dynamics for flexible time windows, generalizing for diverse domains, and speeding up simulations, while keeping accuracy and physical fidelity.

3.2 Physics-Informed Machine Learning

Another key topic here is Physics-Informed Machine Learning (PIML). As previously detailed, the main objective of PIML is to enhance learning algorithms' capabilities injecting prior (physical) knowledge into it, where priors are devised through empirical or mathematical formulations. Specifically, this is possible by introducing any combination of observational, inductive, or learning biases into the model.

Xu and Valocchi (2015) proposed an ML framework to address a problem in a domain similar to ours. The authors aim to enhance the predictive capabilities of a physics-based model of the groundwater flow of a specific river region, where they already have a numerical model. Given the predictions performed by the physical model, they compare it with real (observational) data and compute the residual between simulated and observed data points. After that, a machine learning model is fed with observed environmental variables that affect the water flow and predicts the residual error previously computed. Finally, the predicted bias from the data-driven model is then used as an uncertainty bias interval. So, with that bias interval, they are able to compute a confidence interval, using the prediction of the physical model as the center of this interval. In short, they proposed a bias-correcting model (DEMISSIE et al., 2009) guided via physical data of a numerical model.

On the other hand, regarding deep learning approaches, and, more specifically, those that make use of GNNs, it is worth diving into a few proposals. For example, in (HU et al., 2020) the authors deal with power flow prediction in electric systems. For that, they built a model with both inductive and learning biases, as described before. Based on the concept of (multi-task) supervised autoencoders and their (proved) regularization attribute (LE; PATTERSON; WHITE, 2018), they developed an autoencoder that solves the direct and inverse problem of interest. While the encoder is a vanilla Multi-Layer Perceptron (MLP), which maps inputs to the target variables, the decoder is designed as

a neural network inspired on fundamental circuit laws. Thus, the model is constrained both in the learning process, through the optimization of both the direct and inverse problem solving, and the model's architecture, via the introduction of physical knowledge into its design.

Similarly to (HU et al., 2020), the work by Lemos et al. (2022) is another one that falls into those two categories on how to bias a learning algorithm, i.e., both inductive and learning biases are present in their proposal. In that work, the authors' objective is to demonstrate a new approach to discover both physical laws and unobserved properties of complex physical domains with data and well-established scientific frameworks. Specifically, they use observations of the orbital trajectories of the Sun, planets, and moons in order to rediscover Newton's law of gravitation and the masses of these bodies. To do so, they defined a two-step approach to address both learning the law of gravitation and bodies' masses. First, they fit a GNN-based model using the observed trajectories of the celestial bodies. After that, symbolic regression is used to fit an analytical formula with internal components of the GNN. Precisely, the celestial bodies are positioned as the nodes of the graph, and the (physical) forces between bodies are embedded into the edges. The learned edge function is enforced to mimic Newton's law of gravitation, which was devised by an external symbolic regression module (CRANMER et al., 2020). Therefore, their model is able to predict the system dynamics adhering to the underlying physical equations and also to derive properties from it.

In (PARK; PARK, 2019), the authors focused their efforts to deal with wind-farm power estimation. Wind turbines positioned in wind farms handle interactions due to wake generation in the process of energy production. Thus, power production is affected by these wake interactions, reducing the wind farm's full power-generating capacity. In this sense, the authors proposed a physics-guided model, titled PGNN (Physics-induced Graph Neural Network), to estimate the power outputs of wind turbines. They modeled that problem by representing the wind farm and the environmental conditions in the form of a graph, processed by a GNN. With that, they are able to represent both the wind turbines as nodes and those interactions within the edge features of the link between turbines. Also, physics information is injected into their architecture with the help of a weak model, proposed in (PARK; LAW, 2015), that captures the intensity of interaction between wind turbines, weighting the relation between nodes.

3.3 Research Gap

The works here presented have clever and principled solutions to hard and important problems. However, as one can infer, they reside in specific niches. Therefore, there is not a one-fits-all solution to all of them. Even so, it is interesting to note the recent trend of approaching physical problems with learning algorithms. Not only that, works in the intersection $PIML \cap GRL$ seems to produce better results in predicting (physical) system dynamics. That points to a research direction not well explored, with promising results. We believe that only a handful of efforts have been interested in addressing real-world problems; most of them focus on problems with reduced scope and/or making use of limited analytical models with toy data. That suggests a stimulating gap of research that drew our attention and that is the subject of this work.

Therefore, the main goal of this research is to build a graph neural network that is able to incorporate physics principles to model oceanic variables, such as sea surface height and water current velocity. The specific objectives are the development of a model that can handle spatially and temporally distributed phenomena, tolerate missing or noisy data, generate physics-grounded representations, and provide accurate and consistent predictions based on domain-specific knowledge. In summary, we seek to design a flexible model that can be applied to any oceanic region where data and prior physical knowledge are available while also performing effectively in real-world conditions. In essence, this dissertation pursues to contribute to the advancement of physical systems modeling by bridging the gap between physics-informed machine learning and graph representation learning.

4 EXPLOITING RELATIONAL BIASES WITH GRAPH NEURAL NETWORKS

In this chapter, we explore a practical problem of both technological and economic significance that involves managing a large volume of temporal data within a known spatial framework. Specifically, our focus is on determining the water current speed at a specific location within a major port area in South America. To accomplish this, we use measurements collected by a network of sensors attached to nearby sea buoys. However, due to technical glitches, a significant number of data points are missing from these sensors. The prediction of sea conditions in the port area is crucial for port authorities who need to coordinate the movement of numerous heavy ships, such as those transporting oil or ore. Current physical models for this task are expensive to develop and support, as they require high-quality environmental measurements, accurate boundary conditions, and a detailed 3D representation of the location, including topological and bathymetry studies.

Abstractly, our problem falls within the realm of sequential data collection through a network of sensors with a fixed spatial structure, wherein the sensors often suffer from faults. Consequently, our specific application represents a broad category of significant forecasting challenges encountered by various industries that could benefit significantly from the properties of graph neural networks.

To address this problem, we developed a GNN-based model that grasps the spatial relationships within the domain. By leveraging historical real-world data collected from the Sepetiba/Ilha Grande Bay in Brazil, we are able to train and optimize the parameters of our model. In this chapter, we provide a detailed description of the GNN design we adopted and explore various nontrivial assumptions regarding its structural aspects, analyzing their impact. Our results demonstrate that the GNN achieves outstanding performance when compared to classical and state-of-the-art auto-regressive models. More importantly, it benefits from the spatial structure imposed upon it, which is the initial hypothesis that motivated our implementation.

Figure 6: **Locations of Sepetiba/Ilha Grande Bay's main port terminals.** Locations in order of appearance: 1 - Angra Terminal (Oil) – TEBIG; 2 - Guaiba Island Terminal (Ore) – TIG; 3 - Port of Sepetiba; 4 - CSN Terminal.

4.1 Background

In the following sections, we detailed our problem and its characteristics (Section 4.1.1), and recap definitions previously discussed, in order to understand the hypothesis adopted and how we approached the problem via GNN modeling (Section 4.1.2).

4.1.1 The Sepetiba/Ilha Grande Bay Forecasting Problem

The Sepetiba/Ilha Grande Bay is situated in Brazil's southeastern region, specifically in the State of Rio de Janeiro. This bay is a protected area in close proximity to the city of Rio de Janeiro, characterized by its diverse port infrastructure and significant maritime traffic. To provide visual context, Figure 6 illustrates the region of interest within South America (left) and highlights the regions' main port terminals: TEBIG, TIG, Port of Sepetiba, and CSN (right).

The operational efficiency and safety of commercial sea vessels are directly influenced by sea conditions and weather phenomena, which play a crucial role in navigation. Extreme events like low visibility or strong currents can disrupt ship traffic, necessitating the interruption of operations. Port authorities are responsible for accurately predicting environmental parameters, particularly within the short-term range of 24 to 48 hours. This task becomes even more critical in shared channels accommodating vessels of varying sizes, as in the present case. That is the case for the present problem, as we are interested in predictions within the 24-hour time window.

Typically, the prediction of environmental parameters relies on physical models of atmospheric and hydrodynamic circulation. These models incorporate global-scale model and low-resolution satellite data, measurements of local wind, and tidal variations boundary conditions (PIANC MarCom, 2012). Additionally, accurate 3D grid models of the

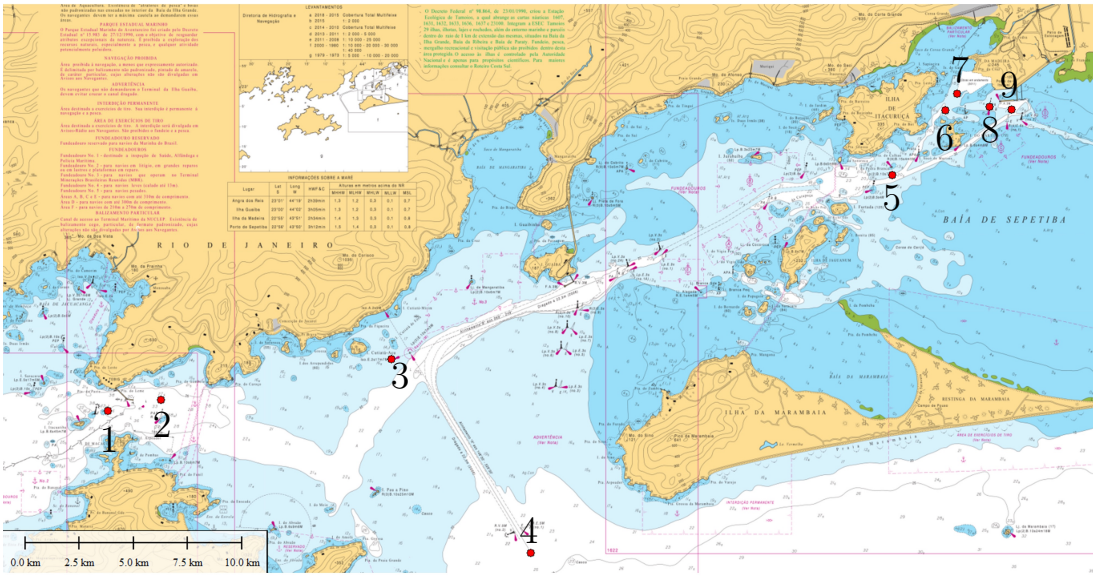


Figure 7: **Sensor buoys' locations. Oceanic phenomena, such as current and wind velocity, and tides are measured in each site.** Buoys' names: 1) B18 2) BEV 3) TIG 4) TIG 1 5) Bifurcação 6) Evolução 7) Pier 8) BPA 9) B22.

area, encompassing coastline and bathymetry information, are employed.

However, with advancements in sensing, data transmission, and storage technologies, there has been increased availability of measured data. This, in turn, has fueled the demand for more intricate data-driven techniques for forecasting time series (WU et al., 2019). These data-driven models operate independently of physics-based models and domain-knowledge inputs, automatically updating as new measurements become available. Our work aligns with this trend toward data-driven modeling.

In this study, we propose a novel modeling approach that leverages a spatially distributed network of sensors present along the Sepetiba/Ilha Grande Bay to forecast the speed of water current at a specific buoy, hereby defined as *Bifurcação*. The network consists of nine navigational buoys, indicated in Figure 7 (with *Bifurcação* labeled as number 6). Each buoy measures a set of oceanic variables, including tide elevation, current and wind speed, and visibility. The speed of the water current in the bay is significantly influenced by tide elevation, with a weaker dependence on wind speed through surface drag, and meteorological effects (indirectly indicated by visibility). A reliable forecasting system must capture these dependencies and their temporal evolution.

Every buoy within the network has a sampling frequency of ten minutes for data measurement. Considering the fairly slow dynamics of the system, we made the decision to sub-sample the data and focus on twenty-minute intervals between measurements. It is

important to highlight that the data collection process is prone to faults, with numerous missing features in many measurement rounds. Additionally, there are instances where buoys fail to report one or more features for extended periods, stretching to months. These challenges highlight the advantages of employing a Graph Neural Network (GNN) approach that can effectively leverage spatial information to address such issues.

4.1.2 Graph Neural Networks

In various scenarios that require the recognition and utilization of hierarchical patterns, deep neural networks have proven to deliver remarkable and occasionally unexpected performance. Nonetheless, fully connected neural networks encounter challenges when it comes to capturing constant relationships between entities (BATTAGLIA et al., 2018). While certain neural networks, such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs), incorporate weight sharing among multiple units, these connections primarily remain local within the model, either in a spatial or temporal sense.

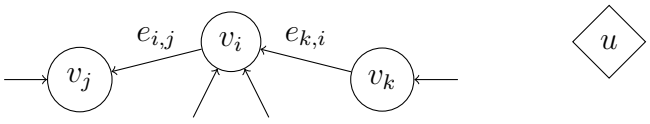


Figure 8: **Schematic of a graph with entities' features.** Here nodes, edges, and global attributes are presented.

To recap, GNNs have been developed to model a domain's structure, expressed through relations between entities. Specifically, a GNN uses nodes (representative of multiple classes), edges, and global attributes to codify the domain. Figure 8 illustrates a piece of a graph with such objects.

In order to comprehend, concretely, the components of a GNN, take into consideration a simpler example weakly related to the application later described. Imagine a mass-spring system. Our goal is to accurately predict the mass position of such a system. One can model it using a graph abstraction, where nodes represent the system's masses, while edges represent the physical relations between nodes, i.e. interactions between masses due to the springs. The node features v_i are its position, velocity, and mass value, for instance. The edge features, on the other hand, e_k are the stiffness and the natural length of the spring connecting the respective masses. Lastly, the global attribute u, which is visible and shared within all entities, is the gravity force. To predict the model's features, a prediction function is applied to entities. For example, one may wish to predict one mass (node) position after n-time steps in that hypothetical mass-spring system.

In this chapter, we employ the comprehensive definition of Graph Neural Networks (GNNs) as outlined by Battaglia et al. (2018), which was introduced in Chapter 2. To recapitulate, within this general framework, a GNN encompasses functions that are associated with nodes, edges, and global attributes:

$$
\begin{aligned}
e'_k &= \phi^e(e_k, \mathbf{x}_{r_k}, \mathbf{x}_{s_k}, u), & \bar{e}'_i &= \rho^{e\to v}(E'_i),\\
\mathbf{x}'_i &= \phi^v(\bar{e}'_i, \mathbf{x}_i, u), & \bar{e}' &= \rho^{e\to u}(E'),\\
u' &= \phi^u(\bar{e}', \bar{\mathbf{x}}', u), & \bar{\mathbf{x}}' &= \rho^{v\to u}(V'),
\end{aligned}
$$

where primed attributes represent updated features. Nodes, edges, and global attributes, respectively $\mathbf{x}_i$, e_k, and u, are obtained by learnable functions $\phi^{\square}$. Additionally, these entities' attributes are aggregated by permutation invariant $\rho^{\square}$ functions.

4.2 Proposed Model

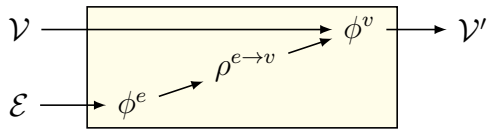
In accordance with Battaglia et al. (2018), we consider a graph to be represented by a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ denotes a set of nodes. Each node v_i is associated with a real-valued feature vector $\mathbf{x_i}$. The set of triples $\mathcal{E} = \{(\mathbf{e}_k, r_k, s_k) \mid k = 1, \ldots, N\}$ defines the edges, where each triple contains an edge attribute $\mathbf{e}_k$ represented as a real-valued vector associated with the edge (r_k, s_k)[1].

The structure of our underlying graph is defined as follows. Each node in the graph corresponds to a buoy, associated with a set of features such as current velocity in the x-axis and y-axis, wind velocity in the x-axis and y-axis, local sea level, temperature, and visibility. To capture the relationships between nodes, domain experts have selected specific node attributes that influence other nodes' features. These relationships are represented by the edges in the graph. The edge attributes are derived from the adjacent node attributes, including the current velocity components in the x-axis and y-axis, as well as the local sea level.

We consider two graph topologies: a fully connected (coined as **non-local** model) and a fully disconnected version (called **local** model). The first one captures effects between sensors, given that they share a geographic location and, thus, are spatially related. The second does not use the full strength of GNNs and was built only for comparison purposes. Illustrating these two schemes we have:

[1]It is worth noting that in our work, we do not utilize the global attribute $\mathbf{u}$ defined by Battaglia et al. (2018).

• The **non-local** graph neural network updates node features taking all relational information into account, inspired by (WANG et al., 2018). A representation of this architecture can be seen below:



The function $\phi^e(\mathbf{e}_k) = \mathbf{e}'_k$ is responsible for updating each edge attribute $\mathbf{e}_k$. Next, these updated attributes are aggregated using a simple *sum* operation, which results in $\bar{\mathbf{e}}_i = \rho^{e\to v}(\{\mathbf{e}'_k\}) = \sum_{k:(i,k)\in\mathcal{E}} \mathbf{e}'_k$. Furthermore, the function $\phi^v(\bar{\mathbf{e}}_i, \mathbf{x}_i) = \mathbf{x}'_i$ updates each node v_i based on the aggregated edges' attributes $\bar{\mathbf{e}}_i$. As discussed in Section 2.2, in the context of GNNs, the update functions applied to the entities of a graph are differentiable. In this particular case, the update functions ϕ^e and ϕ^v correspond to basic neural networks, specifically Multi-Layer Perceptrons (MLPs), which take the entity's attributes as input. In other words, given the features observed by the sensors at a given time step, which are stored in the nodes and edges of the graph, the values of these features are predicted in the next time step by updating the edge attributes using $\mathbf{e}'_k = \text{MLP}^e(\mathbf{e}_k)$, and subsequently updating the node attributes using $\mathbf{x}'_i = \text{MLP}^v(\bar{\mathbf{e}}_i, \mathbf{x}_i)$.

• The **local** graph neural network, on the other hand, looks only at node-level attributes, i.e., it only updates $\mathbf{x}'_i = \phi^v(\mathbf{x}_i)$.

It is worth emphasizing that while information does not flow between nodes in the second scheme, it still benefits from the graph structure, given that the nodes share the same update function $\phi^v = \text{MLP}^v(\mathbf{x}_i)$.

To facilitate the GNNs in capturing the temporal changes in signals, we construct node and edge attributes by concatenating measurements from the previous 48 hours, which are obtained at 20-minute intervals. This concatenation results in node feature-vectors $\mathbf{x}_i \in \mathbb{R}^{145\times 7}$ and edge feature-vectors $\mathbf{e}_k \in \mathbb{R}^{145\times 3}$, encompassing 144 observed data points along with the inference time from which a prediction is made.

4.3 Experiments

In the following sections, we make an in-depth discussion of topics related to the experimental part of the work. In Section 4.3.1 it is detailed the dataset, its characteristics, difficulties dealing with it, and how we overcome them. Next, the model configuration

timestamp	bcx_mean	bcy_mean	...	B18v
$01-01-2018\ 00h00$	0.789	0.567	...	NaN
$01-01-2018\ 00h10$	0.743	0.598	...	1.0
...	...	...	...	...
$31-12-2019\ 23h50$	0.722	NaN	...	1.0

Table 1: **Dataset structure.**

is presented in Section 4.3.2. Lastly, in Section 4.3.3, experimental results and their implications are discussed in detail.

4.3.1 Datasets

The set of observations consists of a large database from 01/01/2018 00:00 to 12/31/2019 23:50 (2 years of measurements), sampled every 10 minutes, from the four sites in Sepetiba Bay/Ilha Grande (TEBIG, TIG, CSN, and Porto de Sepetiba), which provides approximately 105 thousand points.

The data consist of measurements of maritime and climatic phenomena through a network of sensors placed on buoys, totaling 9 measurement points. Seven are the observed phenomena: components of water current velocity on the x and y axes, wind speed components on the x and y axes, sea level and local temperature, and visibility. Table 1 illustrates the arrangement of the obtained data.

Despite the great progress in recent years in terms of capturing, processing, and distributing data in different areas, and considering that the referred network of sensors is vulnerable to all sorts of inclement weather, it would be naive to think that the data would not be faulty. Thus, there was a considerable deficiency, to the point that it was necessary to implement algorithms capable of completing our base with some fidelity: roughly 3.2 million cells are absent in the dataset, corresponding to approximately 43% of the cells.

As an example, Figure 9 presents the distribution for current speed on the x-axis observed in the *Bifurcação* buoy. It is possible to notice that, for this specific region, there isn't faulty data. However, as it's presented afterward, there are regions that have huge bands of missingness due to sensor damage.

As stated above, in a completely opposite way, Figure 10, which presents the distribution for the same phenomenon, yet observed by the buoy *B22*, shows that some of these sensors failed to measure these phenomena for a large part of the period (if not, in some

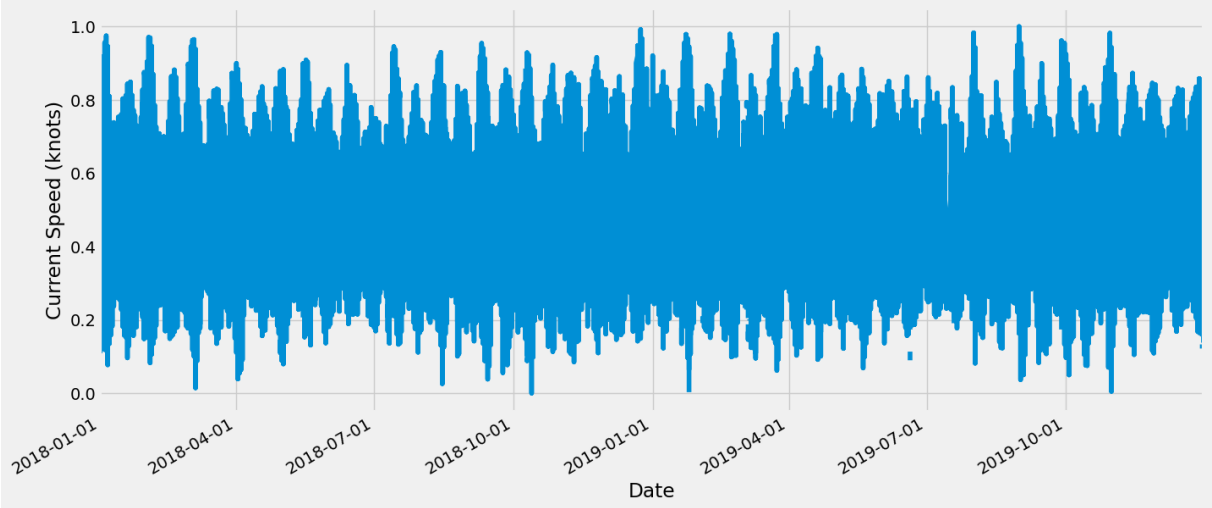


Figure 9: **Current speed on the x-axis observed on Bifurcação buoy.** For this specific region, the sensors are more robust and have periodic maintenance. Therefore, the data is complete.

cases, as illustrated in the figure, in most of it.)

After understanding the problem related to the database, two different methods were applied to perform data imputation:

1. **MICE (*Multiple Imputation by Chained Equations)*** (RAGHUNATHAN et al., 2001; BUUREN, 2007; AZUR et al., 2011)**:** this method works from the *a posteriori* probability maximization scheme (MAP). The central idea is to use the information from the *a priori* distributions of the missing variables together with the relationship between them. Initially, the base is completed from the distributions of the variables themselves. Next, one of these is defined as the dependent variable and a regression is performed having the other variables as independent variables. This process is repeated for all other variables, updating their values in each regression round, until the values of the variables cease to change, that is, when the values are optimized.

2. **Imputation via spatio-temporal distribution:** suppose we consider a buoy and an attribute is absent at some point. If this attribute is present for most buoys, we impute their mean when the variance is smaller than one *threshold*; otherwise, we impute the median. However, if most of the buoys also have this attribute absent, we take the average of this attribute over time for the buoy measurements. That is, in the first scheme, the spatial distribution of the attribute is taken into account, while in the second, the temporal distribution. The value of *threshold* was

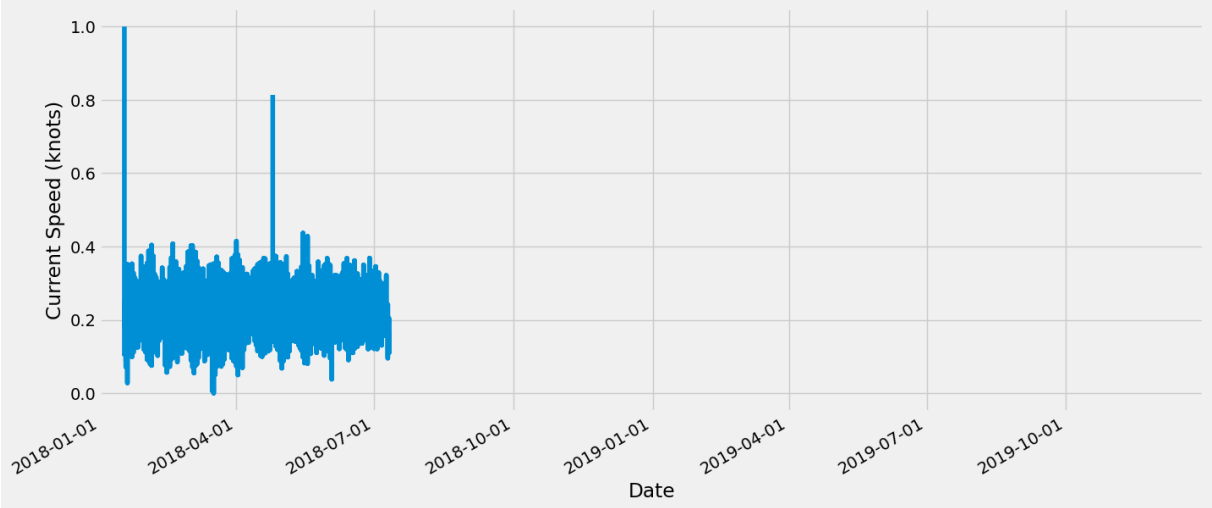


Figure 10: **Current speed on the x-axis observed on B22 buoy.** Like most of the other buoys, this one has a missingness problem. Almost the entirety of the time, the sensor is faulty.

obtained through several attempts, followed by sampling intervals to qualitatively define whether the imputation had a good result or not — the value found was 0.25.

We empirically noticed slightly better performance using the average-based imputation, so the following reported results were obtained with an average-completed dataset.

4.3.2 Model Configuration

Our GNN implementation utilized the Graph Nets library developed by DeepMind[2]. We made necessary adaptations to the library to suit our specific model requirements. During training, we used a batch size of 5,000 data points and employed a sliding window approach with 6 data points for training and 3 data points for testing, resulting in a 2:1 split between the training and test sets. For optimization, we employed the Adam optimizer with a learning rate of 1e-4. The models underwent training for 5 iterations. The neural networks employed in the model had 5 layers with 256 units per layer. We determined these hyperparameters, including the learning rate, number of layers, and number of units per layer, through grid search with cross-validation.

[2]Available at ⟨https://github.com/deepmind/graph_nets⟩.

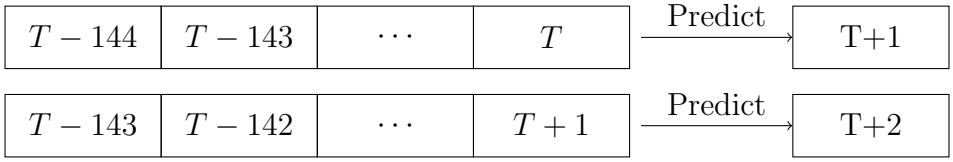


Figure 11: **Forecasting process.** A visual representation of the process to forecast target variables, taking into account the 48h past time window.

4.3.3 Discussion

We employed the developed model to make sequential predictions of the water current at *Bifurcação* for the next 24 hours. In this sequential approach, we utilized data from the previous 48 hours to forecast the subsequent measurement. For that, we used the same procedure used on auto-regressive tasks. After predicting a data point, we shift the time window, so that it incorporates the predicted value as an actual observation. This process is continued for the following 20-minute intervals, until all 72 measurements, corresponding to a 24-hour period with 3 measurements per hour, were predicted. Figure 11 provides an illustrative example of a 2-step prediction. As previously mentioned, since our objective was to forecast the water current for the subsequent 24 hours, we needed to perform 72 similar steps, as depicted in Figure 11.

Empirical evaluations demonstrated that average imputation yielded the most favorable outcomes, thus we retained it for further analysis. Moreover, assessments were conducted for both local and non-local neural network approaches (Section 4.1.2), affirming that the learned models successfully captured the dynamic nature of the signals. Specifically, the predictions accurately reflected the number of peaks within the forecasted period, which is a nontrivial characteristic given the variable nature of peak occurrences, as depicted in Figures 12 and 13. Nevertheless, the models encountered challenges in precisely estimating the peak values of the signals. The overall mean squared error for the non-local model amounted to a Root-Mean-Square error of 0.02 $knots^2$, while the local model yielded a slightly higher value of 0.10 $knots^2$. The squared error was computed by calculating the difference between the observed and predicted values across time.

Figure 12 illustrates a comparative evaluation of our model against widely used baseline models, namely ARIMA (BROCKWELL; DAVIS, 1987) and LSTM (HOCHREITER; SCHMIDHUBER, 1997), in the context of time series prediction. Both baselines were trained with a dataset of 20,000 data points over 50 iterations. Our model outperforms ARIMA significantly and shows a slight improvement over LSTM while demonstrating notable data efficiency. Specifically, our model required fewer data points and iterations

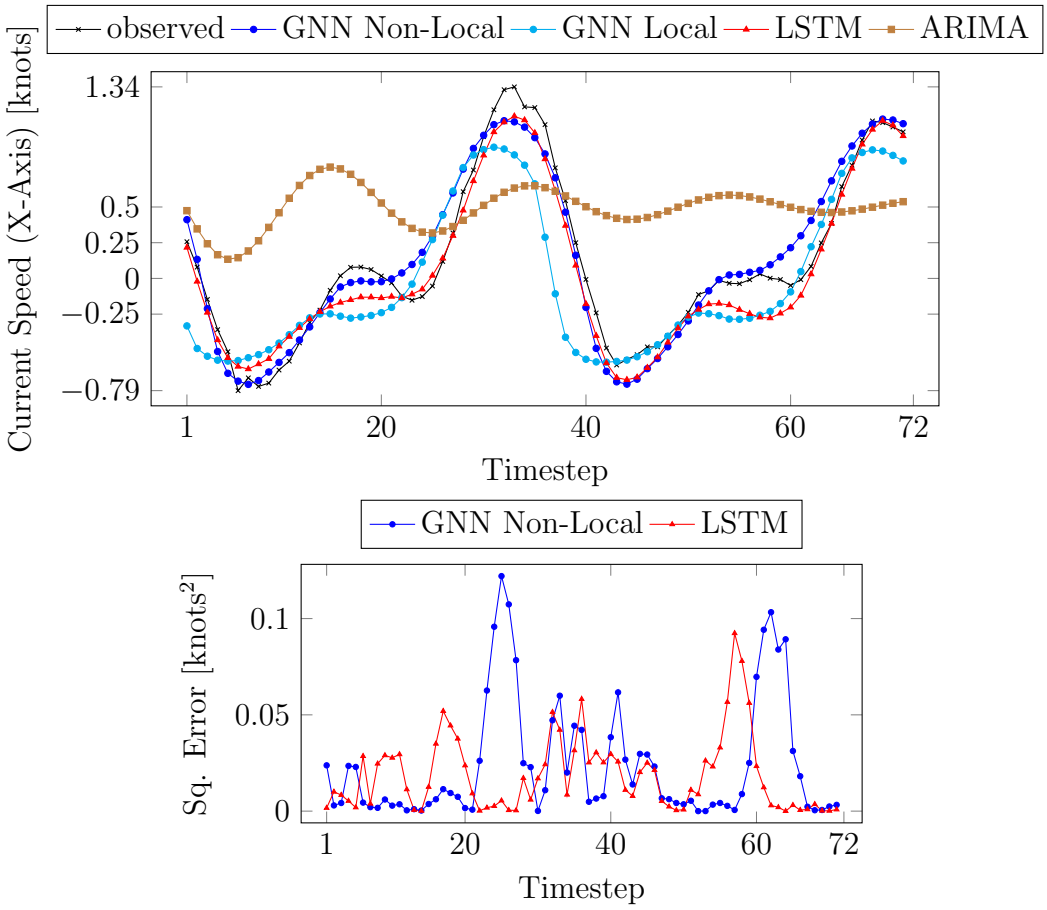


Figure 12: **Results on specific time window.** Proposed models and baseline comparisons. On the top, the forecasting of the different models against the observed values. On the bottom, the mean-squared error (MSE) comparison between the best GNN model and the LSTM baseline.

for training, approximately one-fourth of the baseline models' requirements. Comparing the two graph-based models, we observed that the non-local model achieved the best results, verifying our hypothesis that leveraging information sharing and combination among entities would yield benefits. Additionally, Figure 12 presents the squared error of both the LSTM baseline and the GNN non-local model across the time window. It is worth noting that at the *Bifurcação* buoy, the dominant current component is along the x-axis. A positive current velocity indicates an eastward direction, whereas a negative speed indicates a westward direction.

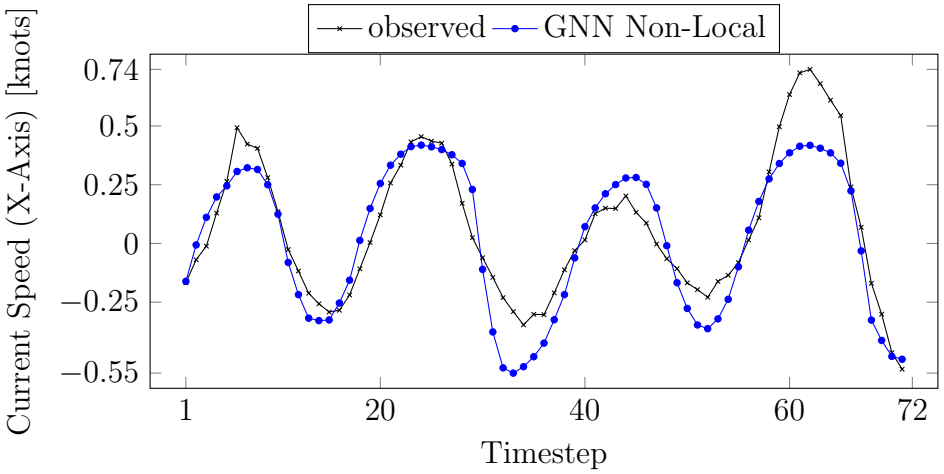


Figure 13: **Results on specific time window.** Forecasting of the best GNN model against the observed values on a different time window.

5 PHYSICS-INDUCED DYNAMIC GRAPH NEURAL NETWORKS: LEARNING MORE EXPRESSIVE REPRESENTATIONS

In the last chapter, we presented our first graph-based approach to forecasting ocean dynamics, yet in a less complex scenario, and used data imputation techniques to handle data missingness. Our goal with that was to establish a proof-of-concept and to understand whether our relational modeling hypothesis in that domain is feasible.

In this chapter, we are not only interested in exploring a more challenging problem in a more important region but also in distilling physical knowledge into our model as Karniadakis et al. (2021) define.

This chapter revolves around tackling the challenging task of forecasting the ocean dynamics within the Santos-São Vicente-Bertioga Estuarine system. This particular region has an intricate topography and multiple driving forces that wield a more significant influence on the system's dynamics compared to the region in the previous chapter. Furthermore, the area encompasses the Port of Santos, the largest port in Latin America.

A key element here is the existence of a physics-based numerical model implemented in the estuarine system. That numerical model, which forms an essential part of our approach, provides valuable outputs that guide the modeling process. By leveraging the outputs of the physics-grounded model, we are able to bias our spatiotemporal Graph Neural Network (GNN) architecture to align it with the underlying physical laws governing the dynamics of the region. This integration empowers our model to effectively capture and respect the fundamental principles determining the prediction of current velocity and sea surface height within the region. Lastly, inspired by the literature, we incorporated a module into our model that automatically learns an optimal graph topology. This is important as it eliminates the need to assume a predefined graph structure. As a result, we achieve improved performance while simultaneously reducing the time and space complexity of our model.

Extensive experimentation has been performed. A detailed ablation study was per-

formed to understand the limitations and advantages of each module within our proposed model. Furthermore, we conducted a comparative analysis, pitting classical and state-of-the-art baselines against our model. The rigorous analysis of our modeling approach, coupled with the careful consideration of alternative methods, serves as a robust foundation for our findings and conclusions.

5.1 Background

At the risk of redundancy, throughout this section we revisit concepts previously discussed and present and deepen new ideas included in our final model. We first introduce the Santos-São Vicente-Bertioga Estuarine system (SSVBES), detailing its characteristics, the gears of the region's numerical model, and the location of the sensors responsible for capturing the data (Section 5.1.1). Then we present the definition of dynamic graphs, which is necessary to subsequently understand how we are dealing with missing data and how that differs from the previous implementation, as well as a recap about the most common GNNs' framework for graphs with only node features, i.e., the message-passing mechanism (Section 5.1.2). Finally, we show one of the limitations of current neural graph-based models, the graph structure hypothesis, and how to circumvent that issue by incorporating the task of inferring the graph's topology into the model (Section 5.1.3).

5.1.1 Forecasting the SSVBES dynamics

The Santos-São Vicente-Bertioga Estuarine System (SSVBES) is situated along the southeastern coast of Brazil within the South Brazil Bight region. Similar to estuarine systems worldwide, the hydrodynamics of SSVBES is primarily influenced by three processes: astronomical tide, meteorological tide, and river discharge. The meteorological tide is determined by the synoptic winds blowing across the adjacent continental shelf, following simple Ekman dynamics. Consequently, it manifests as a gravity wave entering the Santos Bay. The sea surface height (SSH) within the bay experiences a decrease when winds blow from the North-Northeast direction, while an increase is observed when winds blow from the South-Southwest. Figure 14 illustrates the geographical representation of this region, including the three main channels comprising the estuarine system: São Vicente Channel, Santos Port Channel, and Bertioga Channel.

To understand the dynamics of the SSVBES as influenced by these driving forces, a dedicated system known as the *Santos Operational Forecasting System* (SOFS) (COSTA

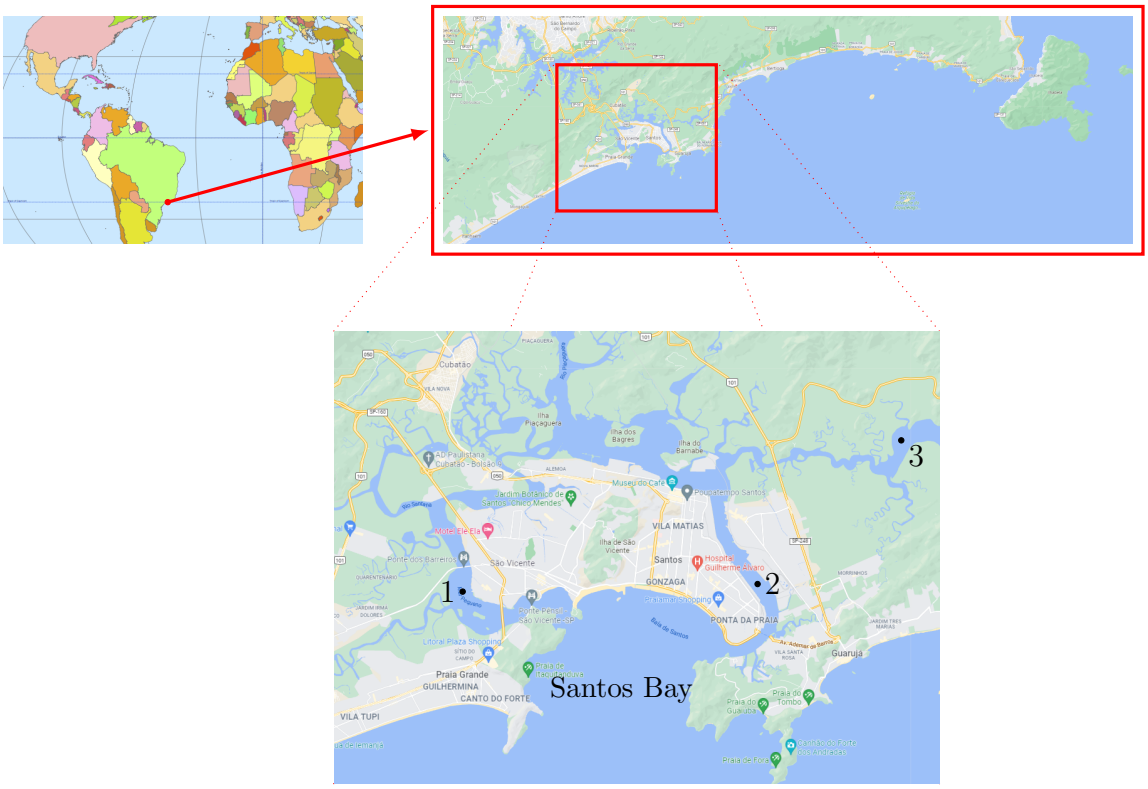


Figure 14: **The Santos-São Vicente-Bertioga Estuarine System.** At the bottom, the main locations: 1 - São Vicente Channel; 2 - Santos Port Channel; 3 - Bertioga Channel.

et al., 2020) has been developed by a large group of researchers. That system aims to offer daily forecasts for the region, employing a finite difference model that applies the *Navier-Stokes* equation with sigma vertical coordinates. It takes into account the effects of winds, tides, density gradients, and river discharge. The model exhibits remarkable performance in accurately predicting both sea surface height (SSH) and current velocity within the system.

Nonetheless, the precision of the model outputs is constrained by the limited availability of river discharge data, as it is not readily obtainable in near real-time. This constraint predominantly affects the accuracy of current predictions as river discharge in estuaries directly impacts flow patterns and induces changes in vertical density stratification, both of which can alter currents.

In this context, and analogously to the Sepetiba/Ilha Grande problem, we have explored the presence of a network of sensors in that region, as shown in Figure 15, to build a data-driven model for the forecasting task, but exploiting the implemented physics-based model (SOFS) in order to guide our model's optimization.

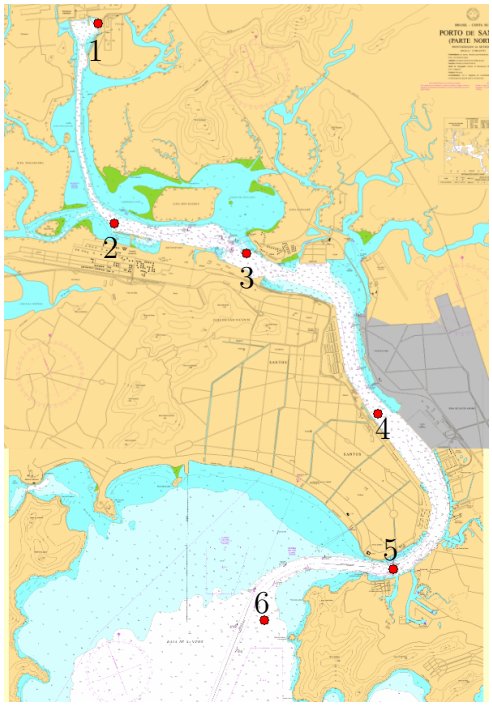


Figure 15: **Observation sites at Santos-São Vicente-Bertioga Estuarine System.** Location and name of all six observation sites used in the experiments: 1) TIPLAM, 2) Alemoa, 3) Ilha Barnabé, 4) CPSP, 5) Praticagem, 6) Palmas.

5.1.2 Dynamic Graphs and Message-Passing

A dynamic graph is defined as a graph whose structure changes over time. More broadly, that change can occur both in the graph's **topology** and at the node and edge **feature level**. Figure 16 shows how dynamic graphs behave over time.

In our scenario, for example, given the tendency to have missing data in the observation window used to train the model, a graph representation would be dynamic. In the Sepetiba/Ilha Grande Bay, we decided to impute the missing data and to have a fixed graph topology. However, that approach leads to biases for the model, as imputations start from hypotheses generally unrelated to the data-generating process. We adopted the strategy of first generating fixed representations (embeddings) for each node in our graph from the observation window, thus working with a dynamic graph (because not all variables will be present, depending on the time window).

Finally, recapping the definition of a *Graph Neural Network* through the perspective of message-passing mechanism (GILMER et al., 2017), described in Chapter 2, we have

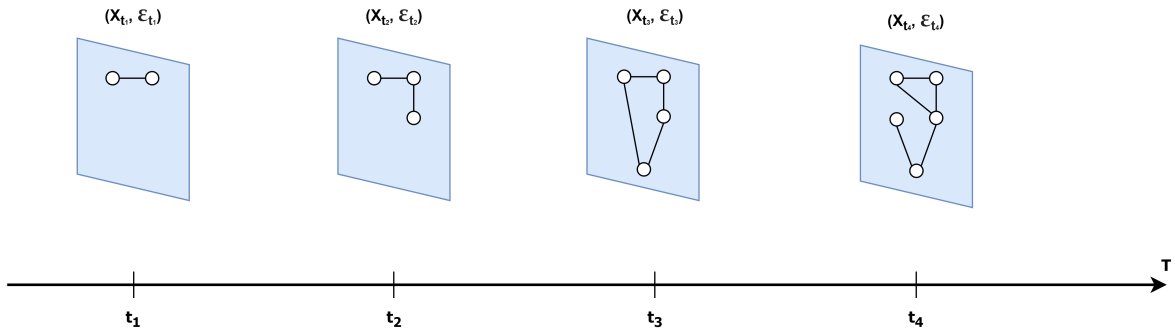


Figure 16: **Dynamic graph time flow.** The structure of the graph, locally, on node and edge feature level, and globally, in its topology, can change over time.

the following equation:

$$\mathbf{x}'_i = \phi\left(\mathbf{x}_i, X_{\mathcal{N}_i}\right) = \phi^v\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \phi^e\left(\mathbf{x}_i, \mathbf{x}_j\right)\right), \tag{5.1}$$

where $\mathbf{x}_i$ is the hidden representation of a node i, ϕ^e and ϕ^v are, respectively, the *message* and *update* differentiable functions, and $\bigoplus$ is a non-parametric, permutation invariant *aggregation* function. Here, we used the message-passing approach of GATv2 (BRODY; ALON; YAHAV, 2021).

5.1.3 Latent Graph Inference

Since its introduction (SCARSELLI et al., 2009), Graph Neural Networks have seen growing research on devising new architectures, following the same trend other neural network models have displayed in recent years. However, one constant throughout every work in that direction, from simple ones like those inspired by signal processing and spectral theory (BRUNA et al., 2013), passing by CNN-inspired ones (DEFFERRARD; BRESSON; VANDERGHEYNST, 2016; KIPF; WELLING, 2017), to complex architectures incorporating attention mechanisms (VELIČKOVIĆ et al., 2018; BRODY; ALON; YAHAV, 2021), is the fact that the underlying graph is always "given" or easily inferred depending on the task domain and the graph's topology is assumed to be fixed.

This is a restrictive and, sometimes, even prohibitive hypothesis. In scenarios where the graph structure is dynamic or the test set has graph structures not seen during training, it is a hard constraint.

A few researchers have developed techniques to learn not only a specific task but also the graph itself. Some efforts have explored the graph's geometric properties as

encompassed within its Laplacian matrix (LI et al., 2018) to infer the topology, while others have proposed methods that take advantage of the latent representations that machine learning models are able to leverage (WANG et al., 2019), also known as latent graph inference.

Graph topology inference is also interesting due to its ability to reduce GNNs' complexity, given that these models are sensitive to the number of edges it has to traverse ($\mathcal{O}_{GNN}(|\mathcal{E}|)$, where $|\mathcal{E}|$ is the number of edges.)

Therefore, we implemented a latent graph inference module as part of our final architecture, using the *Differentiable Graph Module* (DGM) (KAZI et al., 2022) implementation as inspiration, and making important compatibility adjustments with our modeling. The details of this module and the experimental results will be discussed in the following sections.

5.2 Proposed Architecture

We propose a modular architecture, composed of two elements: the Differentiable Graph Module (DGM) and the Encoder-Decoder. This section, hence, provides a wide description of each module, along with the associated concepts and definitions.

To abstract and model the Santos-São Vicente-Bertioga Estuarine System as a dynamic graph, we adopt a graph representation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes $v_i = (\rho, \ell)$, with each node v_i consisting of a node type ρ at a corresponding location ℓ, and $\mathcal{E}$ is the graph initial *adjacency* matrix. The node types correspond to measured and predicted ocean-related variables, such as sea surface height (SSH), water current, astronomical tide, and SOFS predictions, for instance. The node locations represent the six measuring stations, previously depicted in Figure 15.

Thus, every data point corresponds to an event associated with a specific node v_i. For instance, a measurement from an SSH sensor at the Alemoa station (site 2) is considered an event associated with the node $(\texttt{SSH}, \texttt{Alemoa})$.

Events are represented as pairs $(\mathbf{x}_t^i, t)$, where $\mathbf{x}_t^i \in \mathbb{R}^{d_\rho}$ and $t \in \mathbb{R}$. These pairs denote either measurements or forecast values, such as astronomical tide and SOFS forecast data, which can serve as input to the model. The length d_ρ of the feature vector depends on the node type. For instance, water current measurements consist of two features, namely speed and direction, while SSH measurements lead to a single feature.

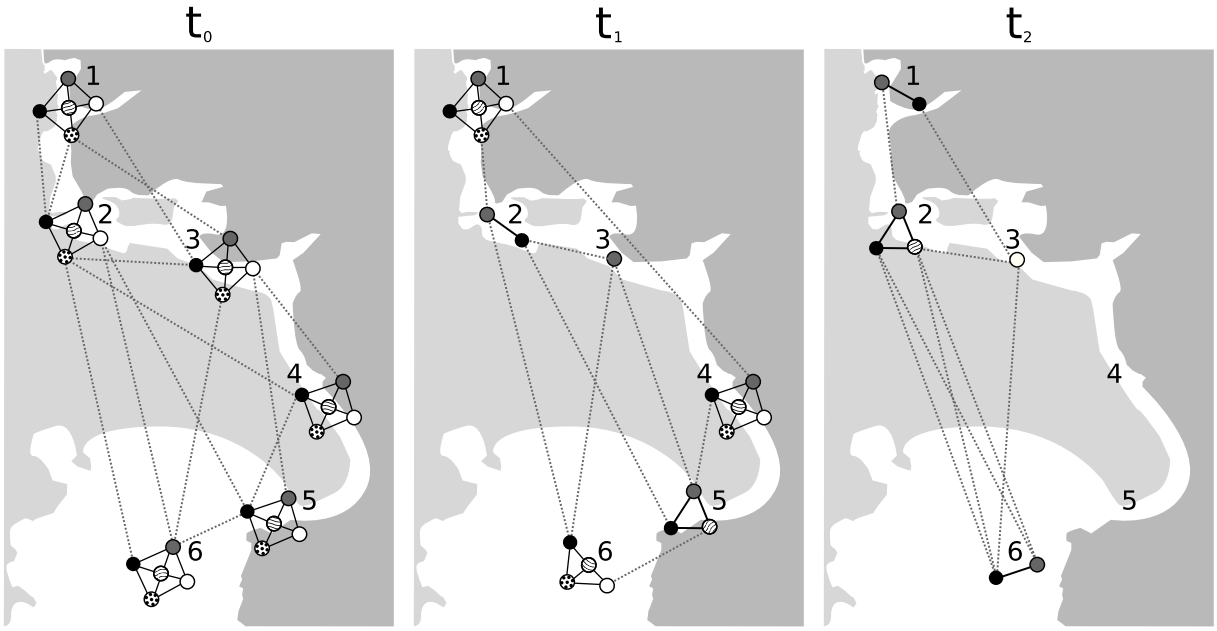


Figure 17: **Representation of the SSVBES as a graph.** Since the sensors are present in the wild, for each inference time $t_{\square}$ certain nodes can be present or not. Therefore, the graph structure will dynamically change within the inference time.

In order to build the graph that abstracts the SSVBES dynamics at a specific **inference time** t_0, we first take **past** and **future** time windows in days, with sizes s_r and s_f, respectively. After that, for each pair (ρ, ℓ) of node type and location, we check if there is at least one measurement in both time windows. With that, we are able to define the set of nodes $\mathcal{V}_0$ and edges $\mathcal{E}_0$ for the underlying graph in that inference time. Figure 17 shows how the graph changes dynamically depending on the inference time it is related to, due to the presence/absence of data measured by the different sensors in the network.

However, we still need to handle missing data inside time windows. For that, we implemented a temporal encoder that takes the sequence of observed and simulated time-series data, aggregates it with the timestamp encoding, and send it into a Recurrent Neural Network (RNN), producing a fixed-size embedding vector that represents the time-series for that specific time window, node type, and location.

The timestamp encoder is implemented following the description provided by Time2Vec (KAZEMI et al., 2019). It is defined as `Time2Vec`: $\mathbb{R} \to \mathbb{R}^T$ and is responsible for embedding a scalar representation of time, derived from event timestamps, into vectors of size T.

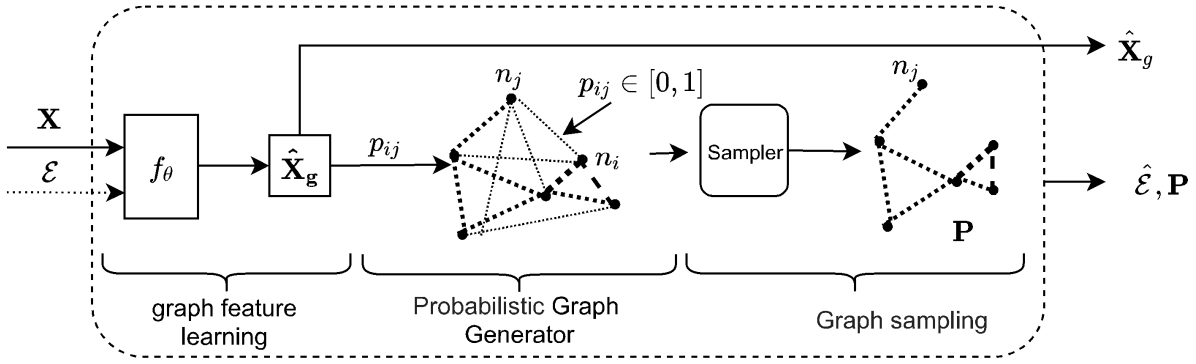


Figure 18: **Differentiable Graph Module (DGM) components and details.** The latent graph inference DGM module can be divided into 3 parts: (1) the graph feature learning, responsible for encoding the node features into a latent space where the probability (distance) between nodes is computed; (2) the probabilistic graph generator, which is where the matrix of unnormalized probabilities are calculated *de facto*; (3) the graph sampling, where the *Gumbel-Top-k Trick* (JANG; GU; POOLE, 2017) is applied for sampling without the need for normalizing the probability distributions. Extracted from (KAZI et al., 2022).

The Temporal Encoder is devised for both the past and future window of each node type, i.e., $\mathcal{F}^r_{\mathcal{T},\rho}$ and $\mathcal{F}^f_{\mathcal{T},\rho}$, respectively. For our work, $\mathcal{F}^{\square}_{\mathcal{T},\rho}$ is implemented as separate LSTMs (HOCHREITER; SCHMIDHUBER, 1997) in both cases. Mathematically, the temporal encoder is a function $\mathcal{F}_{\mathcal{T},\rho}$: $(\mathbb{R}^{s_r \times (d_\rho + T)}, \mathbb{R}^{s_f \times (d_\rho * C + T)}) \rightarrow \mathbb{R}^H$. It is worth pointing out that C indicates whether this is a *simulation* variable (C is set to 1), or an *observed* one (C is set to 0).

5.2.1 Differentiable Graph Module (DGM)

The DGM module is composed of three parts, as shown in Figure 18. Each one of them is responsible for important operations to infer the latent graph topology. These components are described below.

5.2.1.1 Graph Feature Learning

This part of the module consists of a learnable function $f_\theta(\cdot)$, used for graph representation in a latent space. This function can be any non-linear function, for instance an MLP, or a graph convolution operator, in case we have an input graph, i.e., $\mathcal{E} \neq \varnothing$. In our experiments with the DGM module, we used a 1-layer MLP, i.e. a linear layer followed by a nonlinearity.

5.2.1.2 Probabilistic Graph Generator

The probabilistic generator assumes the hypothesis of a fully-connected graph and computes the probability for each $(i, j) \in \mathcal{E}$ based on the following expression:

$$p_{i,j} = e^{-t||\hat{x_i} - \hat{x_j}||_2^2},$$

with t being a learnable temperature parameter. In this work, we embraced Kazi et al. (2022) suggestion and fixed the temperature parameter with a value of 4.0.

Additionally, we used the Euclidean distance to define the probability of two nodes being linked in the same way Kazi et al. (2022) did. Yet, other geometric spaces can be used.

5.2.1.3 Graph Sampling

After estimating the edge probability matrix P, a fixed $k-$degree graph is sampled To do that, it is used the *Gumbel-Top-k Trick* (JANG; GU; POOLE, 2017) method for sampling from the unnormalized probability distribution defined for each p_i. From another perspective, this is similar to imposing a stochastic relaxation of the *k-NN* rule for sampling. This method uses:

- The unnormalized probability distribution $p_i = [p_{i,j} : j = 1, ..., N]$ of a receiver node i;
- The k edges extracted given the first k elements of the result of the operation $\text{argsort}(log(p_i) - log(-log(q)))$, where $q \in \mathbb{R}^N$ is uniform i.i.d. in the interval $[0, 1]$;
- The new set of edges formed by k elements, denoted by $\hat{\mathcal{E}}$.

Finally, the DGM outputs the graph $\hat{\mathcal{G}} = \{\hat{\mathcal{V}}, \hat{\mathcal{E}}\}$. For the experiments with DGM module, we used a $k = 6$ neighborhood.

A key distinction between our implementation and the approach described in (KAZI et al., 2022) lies in our ability to handle graphs with varying numbers of nodes within a batch. While we employ the same optimization algorithm (mini-batch gradient descent) as Kazi et al. (2022) to optimize our model, our problem necessitates the inclusion of different node types within the same batch, resulting in varying node counts, and the author's implementation assumes graphs with a fixed number of nodes within the batch.

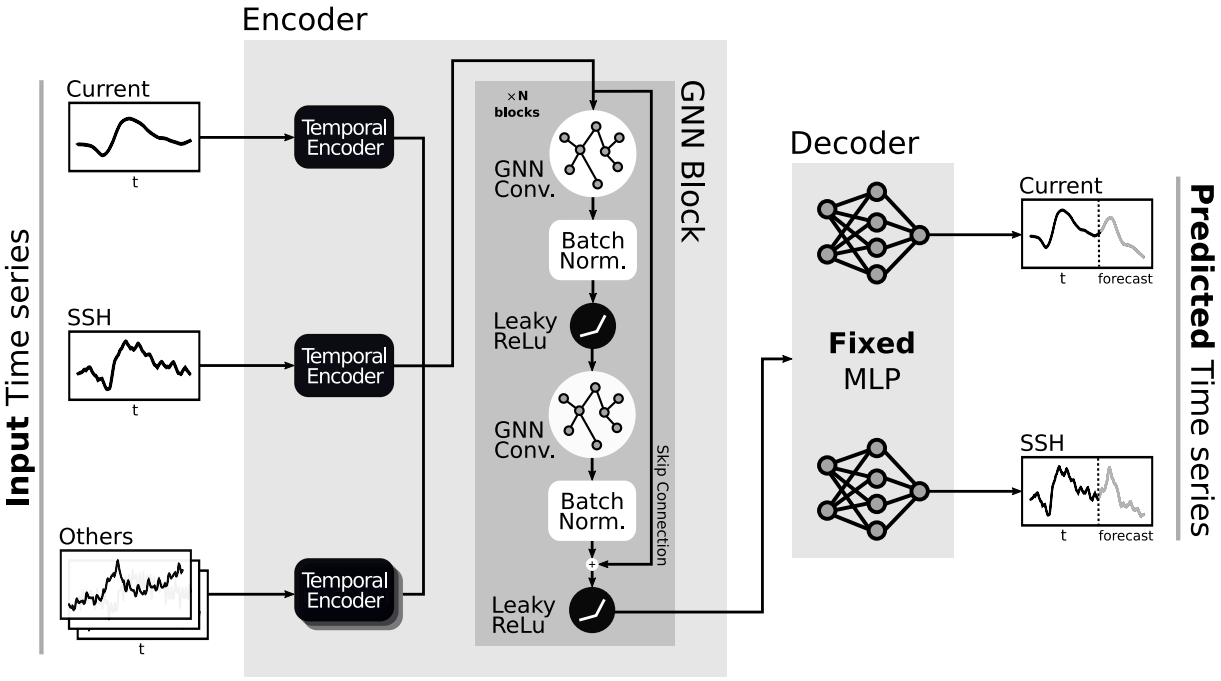


Figure 19: **Encoder-Decoder architecture.** Each node type is related to a *Temporal Encoder* and a *Decoder*, whereas all nodes share the same *GNN Block* module. Specifically, the Graph Neural Network is a stack of GNN Blocks, internally composed of a sequence of graph convolutions, normalization layers, and non-linear activation functions.

Consequently, we had to adapt their algorithm to accommodate this particular characteristic and ensure compatibility with our problem setting.

5.2.2 Graph Learning

The graph learning module is the element inside the Encoder-Decoder that is responsible for propagating/diffusing the information encompassed into each node's feature vector between them. It takes the fixed-size embedding produced by the temporal encoder discussed above and transforms it into new representations in a more expressive latent space. For that, it makes use of a local *GNN Convolution* operation, defined as $\mathcal{F}_{\mathcal{G}}\colon \mathbb{R}^{k_i \times H} \rightarrow \mathbb{R}^{h_i{}'}$, where k_i is the set of neighborhood nodes of the node i, followed by normalization and nonlinearity. We also apply a skip connection (HE et al., 2016).

That set of operations defines the message-passing process of our architecture, which we called *GNN Block* because we can stack them as layers. For the graph convolution, we used the Graph Attention proposed in (VELIČKOVIĆ et al., 2018), followed up with improvements in (BRODY; ALON; YAHAV, 2021).

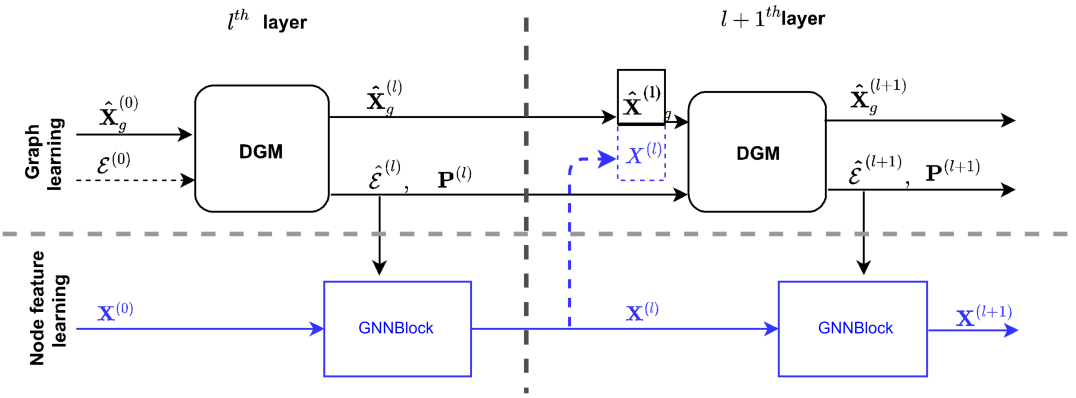


Figure 20: **DGM+GNN composition.** Inspired by Kazi et al. (2022), we compose the DGM module and the GNN, where they mutually feed one another. DGM receives the transformed graph feature matrix $\hat{X}_g^{(l)}$ and the graph adjacency matrix $\mathcal{E}^{(l)}$, and returns a new representation of the features $\hat{X}_g^{(l+1)}$ and a new adjacency matrix $\hat{\mathcal{E}}^{(l)}$. The new adjacency matrix is then used as the new graph structure in the GNN Block. Adapted from (KAZI et al., 2022).

5.2.3 Decoders

Lastly, we get back to the original time-series space through decoders, in order to forecast the signal of interest. The decoder is defined as $\mathcal{F}_{\mathcal{D},\rho}\colon \mathbb{R}^{h_i{}'} \to \mathbb{R}^{s_f \times d_\rho}$, providing the forecast vector $\hat{Y}$. In our model, we used MLP as decoders.

The Encoder-Decoder module is illustrated in details in Figure 19. This module comprises the *Temporal Encoder*, the *GNN Block* and the *Decoder*.

5.2.4 Composing DGM and GNN

Figure 20 represents how the *GNN Block* and the DGM are coupled in a modular fashion. These modules' operation alternates as information propagate forward. DGM first computes the graph's topology, and then, the GNN uses that information as input together with the nodes' features.

5.3 Experiments

Here we describe our dataset and its characteristics (Section 5.3.1), the experimental design in detail (Section 5.3.2), the metrics used to train and assess our model (Section 5.3.3), the configurations and the stack of tools used to implement and run our experiments (Section 5.3.4), and discussions raised after empirically testing our model (5.3.5).

5.3.1 Datasets

Our dataset contains a diverse set of observed environmental data collected from the 6 SSVBES' sites illustrated above, and simulated data, such as forecasted astronomical tides and SOFS predictions. Each location measures oceanic variables, such as SSH and water current speed and direction, accounting for a total of 32 time series, or 32 node pairs (ρ, ℓ). For the labels, we have two signals of interest: the prediction of water current and SSH at *Praticagem* station, or, in our notation, the two pairs (`SSH`, *Praticagem*) and (water current, *Praticagem*).

We gathered, approximately, 2 years of data between the years 2020 and 2021, with a sampling frequency of 10 minutes. We split the data in the proportion 4:1 between the train and test sets. Specifically, the first 80% of data were used for training and validating the model, while the remaining 20% were for testing. To define each inference time, observed data was aggregated using steps of 30 minutes between windows, meaning there are 3 graphs for each hour. Moreover, data were normalized using the Z-score.

It is worth explaining that even though SOFS simulations and the sensors' measured data have different sample frequencies (the former is obtained every 3 hours, while the latter every 10 minutes), this is not a problem for our model, given that it can incorporate new variables that appear in the 7-days time window used for forecasting. However, we were careful when computing the metric for comparison between our model and SOFS, using the latter's frequency to calculate the metric.

5.3.2 Experimental Design

To evaluate our model performance, we made a series of thorough experiments taking into consideration three experiments. The following is a description of how our experimental scenarios were designed:

Water Current and SSH Forecasting Optimization. For the first experiment, we used the train set to determine which combination of node types optimizes our model for the task of forecasting, separately, the water current and the sea surface height (SSH). We run 5-fold cross-validation for each combination of input variables (node types).

Graph Connectivity Optimization Next, we experimented with different graph topologies. Using the best (input) node types combination previously obtained, we op-

timized our architecture for the most promising graph structure. We followed the same hyperparameter optimization process as the last experiments, i.e., we run 5-fold cross-validation.

Water Current and SSH Forecasting Test Lastly, we compared our model to baselines, using the best configuration we got from the two other experiments. In this setup, we trained the model using the whole training set and forecasting for the test set. As explained in the last section, we were aware of the sampling frequency caveat. Thus, the forecasted time series, for all models, is subsampled to a 3-hour frequency, following SOFS's behavior, and then we compute its metric.

These experiments were performed in two distinct phases. Initially, we worked without employing SOFS' outputs as a node of our graph. Additionally, we treated the water current as a multivariate time series, using the water current's velocity and direction as separate features. Nonetheless, though the results were intriguing, after a closer look at the water current data, we recognized that the dominant component of the water current velocity was along the x-axis (see Figure 21).

Afterward, we started using the water current speed and SOFS's predictions as part of our modeling. It is important to highlight that in this second moment, we also introduced the Differentiable Graph Module (DGM) to further enhance our experiments. However, it is worth noting that when utilizing the DGM, we bypassed the first two optimization experiments. This is due to the inherent nature of the DGM, which not only learns the optimal topology but also determines the most influential nodes within the graph.

5.3.3 Metrics

To train and, therefore, to optimize the parameters of our model, we defined a multi-objective loss function, combining the graph-objective and the task-objective.

Graph-Objective Loss We adapted the graph loss defined in (KAZI et al., 2022) to a forecasting task. Here, we defined a graph loss that takes into account the mean absolute error of the prediction, modulated by the influence of each edge preserved during the graph's topology inference. Mathematically, the graph loss is defined as follows:

$$\mathcal{L}_{\mathcal{G}} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i| \prod_{l=1}^{L} \prod_{j:(i,j)\ \in\ \hat{\mathcal{E}}^{(l)}} p_{i,j}^{(l)}. \tag{5.2}$$

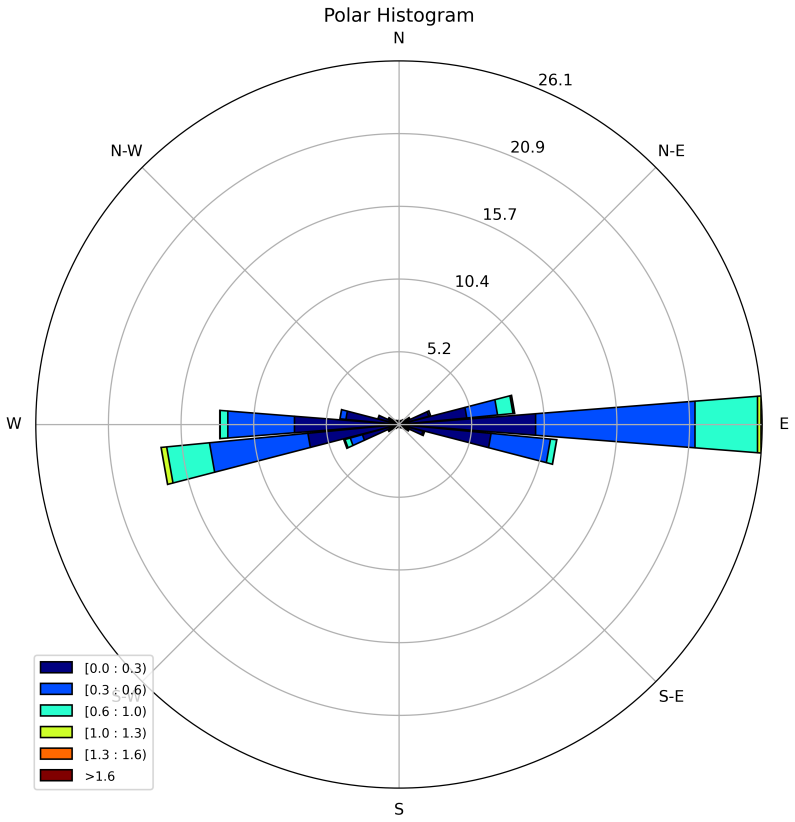


Figure 21: **Water current velocity distribution on *Praticagem* station as a polar histogram.** The water current velocity is almost entirely in the channel direction, i.e., the east-west direction.

Task-Objective Loss We evaluated our model's performance w.r.t. the forecasting task with the metric known as Index of Agreement (IoA) (WILLMOTT, 1981). This metric is defined as:

$$\mathcal{L}_{\mathcal{O}} = 1 - IoA = \frac{\sum_{n=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{n=1}^{N}(|\hat{y}_i - \bar{y}_i| + |y_i - \bar{y}_i|)^2}. \tag{5.3}$$

Finally, the model's total loss is the sum of both losses:

$$\mathcal{L}(\hat{Y}, Y) = \mathcal{L}_{\mathcal{G}} + \mathcal{L}_{\mathcal{O}}. \tag{5.4}$$

It is important to clarify that the complete loss function above is not used in all our experiments. In fact, we only use the multi-objective when the DGM is present. Otherwise, we just use the second term, i.e., task-objective loss. Besides, the IoA is the go-to metric for comparing the implemented models' results.

Additionally, we present results for Root-Mean-Square Error (RMSE) in meters (SSH)

Current optimization (Current w/o signal)				
	Velocity (m/s)		**Direction (degrees)**	
Scenarios	IoA ↑	Std. Dev. ↓	IoA ↑	Std. Dev. ↓
Current	0.706	0.005	0.818	0.009
Current+SSH	**0.726**	0.011	**0.842**	0.015

Table 2: **Water current optimization considering unsigned current.** Results obtained for water current velocity and direction forecasting at *Praticagem* station w.r.t. the input data. A fully-connected graph was imposed on that experiment. Presented values are the mean performance for 5-fold cross-validation. The best results are in bold.

and meters per second (water current) when showing the results from the last experiment.

5.3.4 Model Configuration

We employed well-known frameworks, namely *Pytorch* and *Pytorch Geometric* (PASZKE et al., 2019; FEY; LENSSEN, 2019), to implement the model, and *Weights&Biases* (BIEWALD, 2020) to track all experiments.

As stated above, we used one-layer LSTMs as temporal encoders with a hidden dimension (embedding size) of size 20, two GNN Blocks with GATv2 (BRODY; ALON; YAHAV, 2021) as GNN convolution, and MLPs as the DGM feature learning part and as the decoders in the Encoder-Decoder, as said earlier on. The model's training ran for 15 epochs with a learning rate of 5e-3.

5.3.5 Discussion

The results shown in the following tables point out three main findings. Even though our complete model benefits from sharing and propagating information between nodes/ocean variables, when optimizing for the input data, all experiments showed that the **temporal encoder** is a crucial part of the model. Sparser graphs have good performance compared to denser ones as can be seen in Tables 2, 3, 4, 6, 7 and 8.

However, that does not mean that the GNN's sharing capabilities are useless. On the contrary, they helped the model be more accurate, at the same time they are flexible enough to incorporate/eliminate input data, due to the modular aspect of the model. That is vital in real-world applications, especially those critical ones, such as the one we are dealing with. For example, the water current prediction greatly benefits when SSH or

Astronomical Tides' simulations are incorporated into the model, which is presented in the first experiment, regardless of the use of water current with or without signal (Tables 2 and 6).

It is also worth highlighting how the same behavior happens when we are forecasting SSH. Though less intense, sea surface height also has better performance when information between node types is shared, as presented in Tables 3 and 7. Nonetheless, it is clear that the task of predicting SSH displays narrow improvement, taking SOFS's performance into consideration. Finally, one interesting and common aspect is the fact that for both tasks, forecasting water current or SSH, the model performs better when simulated data is present, especially SOFS SSH and Astronomical Tide. This agrees with our position that physical knowledge, in the form of observational biases, can physically guide our model.

Graph connectivity optimization, our second experiment, also provides insights. In Tables 4 and 8, one can infer that specific (connected) graph structures can have worst performance than disconnected ones. Here, we raise the hypothesis that, because we are dealing with a heterogeneous graph, which means that different node types share information between them, and use the same ϕ and γ functions (eq. 5.1) to every node, our model might be suffering from a phenomenon known as *over-squashing* (ALON; YAHAV, 2021). But we should emphasize that, on average, connected graph schemes outperform disconnected ones.

The third experiment, which we designed for baseline comparison, follows similar conclusions to those discussed above. Tables 5 and 9 show that our model has better performance than SOFS, outperforming the physics-based model by 11% in forecasting water current speed, and keeping on par with the numerical model at the task of predicting the sea surface height.

When compared with statistical models, ours is still competitive and is at the forefront.

SSH optimization (Current w/o signal)		
Scenarios	IoA ↑	Std. Dev. ↓
SSH	**0.940**	0.008
SSH+Current	0.939	0.012

Table 3: **Sea surface height (SSH) optimization considering unsigned current.** Results obtained for SSH forecasting at *Praticagem* station w.r.t. the input data. A fully-connected graph was imposed on that experiment. Presented values are the mean performance for 5-fold cross-validation. The best result is in bold.

Graph Connectivity optimization (Current w/o signal)

	Velocity (m/s)		**Direction (degrees)**	
Scenarios	IoA ↑	Std. Dev.	IoA ↑	Std. Dev.
Disconnected	0.719	0.021	0.830	0.013
Same *type*	0.706	0.017	0.818	0.008
Fully connected	**0.726**	0.007	**0.842**	0.005

Table 4: **Graph connectivity optimization considering unsigned current.** Results obtained for graph connectivity optimization as a function of the model's performance on water current velocity and direction forecasting at *Praticagem* station, using the [Current+SSH] → [Velocity, Direction] scenario. Presented values are the mean performance for 5-fold cross-validation for different graph topology configurations. The best results are underscored in bold. As one can see, the connected model does not always produce the best result. Actually, there are node types that can hurt the prediction of others. This is evidence for the case of using a technique to infer the graph structure.

We implemented two ARIMA-like models for that experiment: (1) SARIMAX - fixed, and (2) SARIMAX - reoptimized. In the SARIMAX - fixed scheme, optimal parameters were determined for the initial training window and then utilized for all subsequent sliding windows. On the other hand, the SARIMAX - reoptimized scheme aimed to improve the modeling accuracy by dynamically searching for optimal parameters on a daily basis. This reoptimization process accounted for the seasonality characteristics inherent in the data. Our model outperforms both SARIMAX models by more than 5%.

Finally, the last two fascinating pieces of evidence empirically raised are associated with the use of the Differentiable Graph Module (DGM). First: when compared to a GNN without the DGM (Encoder-Decoder only), it is possible to see a significant drop in training time. Numerically, we had a 54% increase in the model training speed or a 35% reduction in training time. Moreover, since the DGM module enables the model to bypass the optimization processes (first and second experiments), we have a greater reduction in training time, given that the hyperparameter-tuning phase is part of a model's training. Second: that is remarkable due to the near real-time response of the DGM+GNN model, when operating in inference mode. As ARIMA-like models optimize their parameters during inference time, our complete model is also more advantageous. In summary, the DGM+GNN is better at generalizing to unseen data without the need to retrain the entire model, which is the case for those statistical models (see Table 10.)

Forecasting test (Current w/o signal)

	Velocity (m/s)		Direction (degrees)		SSH (m)	
Models	IoA ↑	RMSE ↓	IoA ↑	RMSE ↓	IoA ↑	RMSE ↓
SOFS	0.599	0.178	0.755	85.18	0.935	0.133
Ours	**0.718**	0.160	**0.842**	65.29	**0.938**	0.123

Table 5: **Forecasting test considering unsigned current.** Results obtained for water current and SSH forecasting at *Praticagem* station compared to SOFS performance. We used the configuration [Current+SSH] → [Velocity, Direction] for water current prediction, and the configuration [SSH] → [SSH] for sea surface height forecasting. A fully-connected graph was used given the results from graph connectivity optimization. Presented values are the mean performance for the entire test set. The best results are in bold.

Current optimization

Scenarios	IoA ↑	Std. Dev.
Current	0.859	0.003
Current+SSH	0.870	0.010
Current+Astronomical Tide	0.879	0.006
Current+SOFS Current	0.865	0.005
Current+SOFS SSH	0.863	0.015
Current+SSH+Astronomical Tide	0.887	0.006
Current+SSH+SOFS SSH+Astronomical Tide	**0.888**	0.008

Table 6: **Water current optimization considering signed current.** Results obtained for water current speed forecasting at *Praticagem* station w.r.t. the input data. At this moment, we had access to different simulation data, s.a. SOFS current and SSH prediction, and Astronomical Tide. A fully-connected graph was imposed on that experiment. Presented values are the mean performance for 5-fold cross-validation. The best result is in bold.

SSH optimization

Scenarios	IoA ↑	Std. Dev.
SSH	0.877	0.007
SSH+Current	0.871	0.013
SSH+Astronomical Tide	0.900	0.009
SSH+SOFS SSH	0.900	0.019
SSH+SOFS Current	0.853	0.008
SSH+SOFS SSH+Astronomical Tide	**0.920**	0.006

Table 7: **Sea surface height (SSH) optimization considering signed current.** Results obtained for SSH forecasting at *Praticagem* station w.r.t. the input data. At this moment, we had access to different simulation data, s.a. SOFS current and SSH prediction, and Astronomical Tide. A fully-connected graph was imposed on that experiment. Presented values are the mean performance for 5-fold cross-validation. The best result is in bold.

Graph Connectivity optimization

Scenarios	IoA ↑	Std. Dev.
Disconnected	0.867	0.016
Same *type*	0.873	0.010
Same *location*	0.893	0.006
Same *type* or *location*	**0.895**	0.006
Fully connected	0.885	0.005

Table 8: **Graph connectivity optimization considering signed current.** Results obtained for graph connectivity optimization as a function of the model's performance on water current speed forecasting at *Praticagem* station, using the [Current+SSH+SOFS SSH+Astronomical Tide] → [Speed] scenario. Presented values are the mean performance for 5-fold cross-validation for different graph topology configurations. The best result is in bold. Analogously, a specific connection can degrade the result. This is, again, evidence for the case of using a technique to infer the graph structure.

Forecasting test

	Speed (m/s)		**SSH (m)**	
Models	IoA ↑	RMSE ↓	IoA ↑	RMSE ↓
SOFS	0.812	0.220	0.935	0.133
SARIMAX (fixed)	0.856	0.202	0.900	0.164
SARIMAX (reoptimized)	0.859	0.193	0.915	0.146
GNN (Enc.-Dec. only)	0.891	0.173	0.938	0.107
DGM+GNN	**0.902**	0.167	**0.939**	0.105

Table 9: **Forecasting test considering signed current.** Results obtained for water current and SSH forecasting at *Praticagem* station compared to SOFS and ARIMA-like models' performance. We used the configuration [Current+SSH+SOFS SSH+Astronomical Tide] → [Speed] for water current speed prediction, and the configuration [SSH+SOFS SSH+Astronomical Tide] → [SSH] for sea surface height forecasting. For the GNN without the DGM module, i.e. the Encoder-Decoder structure only, we used the *same type or location* graph connectivity. For the complete model, i.e. DGM+GNN, the model learned the optimal structure together with the model's parameters. Presented values are the mean performance for the entire test set. The best results are in bold.

Time complexity		
Models	Training (s)	Inference (s)
SARIMAX (fixed)	-	18.4
SARIMAX (reoptimized)	-	739.2
GNN (Enc.-Dec. only)	13,980	0.1
DGM+GNN	**9,180**	**0.1**

Table 10: **Models' time complexity considering signed current.** Related training and inference time for each model. ARIMA-like models optimize their parameters during inference. Therefore, it does not make sense to consider the time it takes to train these models. On the other hand, by analyzing graph-based neural models' time complexity, we can see a significant drop in training time when we incorporate the DGM module. More specifically, we had a 54% increase in the model's training speed, i.e., a 35% reduction in training time. Also, coupling the DGM module enables us to bypass the optimization stages, reducing even more the training time because the hyperparameter optimization phase is part of a model's training. That is remarkable because, differently from classical statistical models, s.a. ARIMA-like, ML ones have near instantaneous time-response in inference mode. Furthermore, machine learning models are able to better generalize to unseen data without the need to retrain the model, which is the case for those statistical models.

6 CONCLUSION

This dissertation carries a significant contribution to the field of physical systems modeling, specifically in the ocean domain, by bridging gaps between physics-informed machine learning and graph representation learning. By developing a physics-induced graph neural network and by leveraging graph representation learning techniques, we have successfully addressed the challenges of modeling temporally and spatially distributed phenomena while incorporating physical knowledge into the modeling process.

In our study, we investigated two specific problems in the context of predicting ocean dynamics in the southeast coastal region of Brazil. Through comprehensive empirical evaluations, we have gained valuable insights and made important findings.

In the Sepetiba/Ilha Grande Bay problem, where our focus was on forecasting water current velocity, we demonstrated the effectiveness of our GNN model in capturing the dynamic nature of the signals. Our model exhibited robust performance, outperforming traditional statistical learning and state-of-the-art models, such as ARIMA and LSTM. By leveraging information sharing and combination among entities, the GNN model showed superior performance, showcasing the benefits of relational modeling in the oceanic domain.

In the SSVBES problem, we further enhanced our model by incorporating physical knowledge and automatic graph structure learning. The results highlighted three main findings. Firstly, the temporal encoder was identified as a crucial component of the model, significantly influencing its performance. Secondly, while sparser graphs showed good performance compared to denser ones, the sharing capabilities of the GNN proved to be valuable. The model's accuracy was improved by incorporating additional input data, especially those coming from simulations, such as SOFS' sea surface height and astronomical tides. This showed how our model benefited from physical knowledge in the form of observational biases. Lastly, automatically learning the graph's topology proved to be competitive, providing significant advantages. The inclusion of the Differentiable Graph Module (DGM) gave a better and faster model when compared with both the

physics-based (SOFS) and statistical models (SARIMAX).

In summary, the findings from our study provide valuable insights for future research and have practical implications for improving the understanding and management of oceanic systems. Future research could focus on further optimizing the GNN architecture and on developing methods to include additional physical knowledge into the modeling process. Additionally, exploring the generalizability of our model to other domains and expanding its application to other coastal regions would contribute to cross-fertilize these two broader fields: physics-informed machine learning and graph representation learning.

REFERENCES

ALON, U.; YAHAV, E. On the Bottleneck of Graph Neural Networks and its Practical Implications. In: *International Conference on Learning Representations (ICML)*. [s.n.], 2021. Available in: ⟨https://openreview.net/forum?id=i80OPhOCVH2⟩. cit. on p. 59.

AZUR, M. et al. Multiple imputation by chained equations: What is it and how does it work? *International Journal of Methods in Psychiatric Research*, John Wiley and Sons Ltd, v. 20, n. 1, p. 40–49, mar. 2011. ISSN 1049-8931. cit. on p. 39.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*, 2014. cit. on p. 15.

BATTAGLIA, P. et al. Interaction Networks for Learning about Objects, Relations and Physics. *Advances in neural information processing systems*, v. 29, 2016. cit. on pp. 16 and 28.

BATTAGLIA, P. W. et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. cit. on pp. 16, 24, 29, 35, and 36.

BERGEN, K. J. et al. Machine learning for data-driven discovery in solid Earth geoscience. *Science*, v. 363, n. 6433, 2019. Available in: ⟨https://www.science.org/doi/abs/10.1126/science.aau0323⟩. cit. on p. 15.

BIEWALD, L. *Experiment Tracking with Weights and Biases*. 2020. Software available from wandb.com. Available in: ⟨https://www.wandb.com/⟩. cit. on p. 58.

BROCKWELL, P. J.; DAVIS, R. A. *Time Series: Theory and Methods*. [S.l.]: Springer, 1987. cit. on p. 41.

BRODY, S.; ALON, U.; YAHAV, E. How Attentive are Graph Attention Networks? *arXiv preprint arXiv:2105.14491*, 2021. cit. on pp. 48, 53, and 58.

BRONSTEIN, M. M. et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv, 2021. Available in: ⟨https://arxiv.org/abs/2104.13478⟩. cit. on pp. 22 and 23.

BRONSTEIN, M. M. et al. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, IEEE, v. 34, n. 4, p. 18–42, 2017. cit. on p. 15.

BRUNA, J. et al. Spectral Networks and Locally Connected Networks on Graphs. *arXiv preprint arXiv:1312.6203*, 2013. cit. on p. 48.

BRUNTON, S. L.; PROCTOR, J. L.; KUTZ, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, v. 113, n. 15, p. 3932–3937, 2016. Available in: ⟨https://www.pnas.org/doi/abs/10.1073/pnas.1517384113⟩. cit. on pp. 16 and 27.

BUUREN, S. V. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, v. 16, p. 219–242, 2007. cit. on p. 39.

CORTES, C.; VAPNIK, V. Support-Vector Networks. *Machine Learning*, v. 20, n. 3, p. 273–297, Sep 1995. ISSN 1573-0565. Available in: ⟨https://doi.org/10.1007/BF00994018⟩. cit. on p. 19.

COSTA, C. G. et al. An operational forecasting system for physical processes in the Santos-Sao Vicente-Bertioga Estuarine System, Southeast Brazil. *Ocean Dynamics*, Springer, v. 70, n. 2, p. 257–271, 2020. cit. on p. 46.

CRANMER, M. et al. Discovering Symbolic Models from Deep Learning with Inductive Biases. *Advances in Neural Information Processing Systems*, v. 33, p. 17429–17442, 2020. cit. on pp. 15 and 30.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, Springer London, v. 2, n. 4, p. 303–314, dez. 1989. ISSN 0932-4194. Available in: ⟨http://dx.doi.org/10.1007/BF02551274⟩. cit. on p. 20.

DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *Advances in neural information processing systems*, v. 29, 2016. cit. on p. 48.

DEMISSIE, Y. K. et al. Integrating a calibrated groundwater flow model with error-correcting data-driven models to improve predictions. *Journal of Hydrology*, v. 364, n. 3, p. 257–271, 2009. ISSN 0022-1694. Available in: ⟨https://www.sciencedirect.com/science/article/pii/S002216940800543X⟩. cit. on p. 29.

FEY, M.; LENSSEN, J. E. Fast Graph Representation Learning with PyTorch Geometric. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. [S.l.: s.n.], 2019. cit. on p. 58.

FUKUSHIMA, K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, v. 36, p. 193–202, 1980. cit. on pp. 20 and 21.

GILMER, J. et al. Neural Message Passing for Quantum Chemistry. In: PMLR. *International conference on machine learning*. [S.l.], 2017. p. 1263–1272. cit. on pp. 24, 28, 29, and 47.

GLIELMO, A.; SOLLICH, P.; VITA, A. D. Accurate interatomic force fields via machine learning with covariant kernels. *Physical Review B*, APS, v. 95, n. 21, p. 214302, 2017. cit. on p. 27.

GORI, M.; MONFARDINI, G.; SCARSELLI, F. A new model for learning in graph domains. In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* [S.l.: s.n.], 2005. v. 2, p. 729–734 vol. 2. cit. on p. 15.

HAMILTON, W.; YING, Z.; LESKOVEC, J. Inductive Representation Learning on Large Graphs. *Advances in neural information processing systems*, v. 30, 2017. cit. on pp. 15 and 28.

HE, K. et al. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, 2016. p. 770–778. ISBN 978-1-4673-8851-1. cit. on p. 53.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. cit. on pp. 41 and 51.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, v. 4, p. 251–257, 1991. cit. on p. 20.

HU, X. et al. Physics-guided deep neural networks for power flow analysis. *IEEE Transactions on Power Systems*, IEEE, v. 36, n. 3, p. 2082–2092, 2020. cit. on pp. 29 and 30.

HUBEL, D.; WIESEL, T. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology*, v. 160, p. 106–154, 1962. cit. on p. 21.

HUBEL, D. H.; WIESEL, T. N. Receptive Fields of Single Neurons in the Cat's Striate Cortex. *Journal of Physiology*, v. 148, p. 574–591, 1959. cit. on p. 21.

JANG, E.; GU, S.; POOLE, B. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations (ICLR)*, 2017. cit. on pp. 51 and 52.

JUMPER, J. et al. Highly accurate protein structure prediction with Alphafold. *Nature*, v. 596, n. 7873, p. 583–589, Aug 2021. ISSN 1476-4687. Available in: ⟨https://doi.org/10.1038/s41586-021-03819-2⟩. cit. on p. 15.

KARNIADAKIS, G. E. et al. Physics-informed machine learning. *Nature Reviews Physics*, v. 3, n. 6, p. 422–440, Jun 2021. ISSN 2522-5820. Available in: ⟨https://doi.org/10.1038/s42254-021-00314-5⟩. cit. on pp. 16, 17, and 44.

KAZEMI, S. M. et al. Time2Vec: Learning a Vector Representation of Time. *arXiv preprint arXiv:1907.05321*, 2019. cit. on p. 50.

KAZI, A. et al. Differentiable Graph Module (DGM) for Graph Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2022. cit. on pp. 49, 51, 52, 54, and 56.

KIPF, T. N.; WELLING, M. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations (ICLR)*, 2017. cit. on p. 48.

KRASNOPOLSKY, V. M.; FOX-RABINOVITZ, M. S. Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction. *Neural Networks*, v. 19, n. 2, p. 122–134, 2006. ISSN 0893-6080. Earth Sciences and Environmental Applications of Computational Intelligence. Available in: ⟨https://www.sciencedirect.com/science/article/pii/S0893608006000050⟩. cit. on p. 27.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. Red Hook, NY, USA: Curran Associates Inc., 2012. (NIPS'12), p. 1097–1105. cit. on p. 15.

KUHN, T. S. *The Structure of Scientific Revolutions*. Chicago: University of Chicago Press, 1962. cit. on p. 20.

KUTZ, J. N. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, Cambridge University Press, v. 814, p. 1–4, 2017. cit. on p. 15.

LE, L.; PATTERSON, A.; WHITE, M. Supervised Autoencoders: Improving Generalization Performance with Unsupervised Regularizers. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2018. (NIPS'18), p. 107–117. cit. on p. 29.

LECUN, Y. et al. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, v. 1, p. 541–551, 1989. cit. on pp. 20 and 21.

LEMOS, P. et al. Rediscovering orbital mechanics with machine learning. *arXiv preprint arXiv:2202.02306*, 2022. cit. on p. 30.

LEWIS, D. D.; RINGUETTE, M. A Comparison of Two Learning Algorithms for Text Categorization. In: *In Third Annual Symposium on Document Analysis and Information Retrieval*. [S.l.: s.n.], 1994. p. 81–93. cit. on p. 19.

LI, R. et al. Adaptive Graph Convolutional Neural Networks. In: *Proceedings of the AAAI conference on artificial intelligence*. [S.l.: s.n.], 2018. v. 32, n. 1. cit. on p. 49.

MINSKY, M.; PAPERT, S. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969. cit. on p. 20.

MITCHELL, T. *Machine Learning*. McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673. Available in: ⟨https://books.google.com.br/books?id=EoYBngEACAAJ⟩. cit. on p. 19.

O'GORMAN, P. A.; DWYER, J. G. Using Machine Learning to Parameterize Moist Convection: Potential for Modeling of Climate, Climate Change, and Extreme Events. *Journal of Advances in Modeling Earth Systems*, v. 10, p. 2548 – 2563, 2018. cit. on pp. 27 and 28.

PARK, J.; LAW, K. H. Layout optimization for maximizing wind farm power production using sequential convex programming. *Applied Energy*, v. 151, p. 320–334, 2015. ISSN 0306-2619. Available in: ⟨https://www.sciencedirect.com/science/article/pii/S0306261915004560⟩. cit. on p. 30.

PARK, J.; PARK, J. Physics-induced graph neural network: An application to wind-farm power estimation. *Energy*, v. 187, p. 115883, 2019. ISSN 0360-5442. Available in: ⟨https://www.sciencedirect.com/science/article/pii/S0360544219315555⟩. cit. on p. 30.

PASZKE, A. et al. Pytorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in neural information processing systems*, v. 32, 2019. cit. on p. 58.

PIANC MarCom. *Use of Hydro/Meteo Information for Port Access and operations*. [S.l.], 2012. cit. on p. 33.

POMERLEAU, D. ALVINN: An Autonomous Land Vehicle in a Neural Network. In: *NIPS*. [S.l.: s.n.], 1988. cit. on p. 19.

RACCUGLIA, P. et al. Machine-learning-assisted materials discovery using failed experiments. *Nature*, v. 533, n. 7601, p. 73–76, May 2016. ISSN 1476-4687. Available in: ⟨https://doi.org/10.1038/nature17439⟩. cit. on p. 15.

RAGHUNATHAN, T. et al. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, v. 27, p. 85–95, 2001. cit. on p. 39.

RAISSI, M.; PERDIKARIS, P.; KARNIADAKIS, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, v. 378, p. 686–707, 2019. ISSN 0021-9991. Available in: ⟨https://www.sciencedirect.com/science/article/pii/S0021999118307125⟩. cit. on pp. 15, 16, and 26.

RAVURI, S. et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, Nature Publishing Group, v. 597, n. 7878, p. 672–677, 2021. cit. on p. 28.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788. cit. on p. 15.

REICHSTEIN, M. et al. Deep learning and process understanding for data-driven Earth system science. *Nature*, v. 566, n. 7743, p. 195–204, Feb 2019. ISSN 1476-4687. Available in: ⟨https://doi.org/10.1038/s41586-019-0912-1⟩. cit. on p. 15.

SANCHEZ-GONZALEZ, A. et al. Learning to Simulate Complex Physics with Graph Networks. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2020. p. 8459–8468. cit. on pp. 16 and 28.

SCARSELLI, F. et al. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, v. 20, n. 1, p. 61–80, jan. 2009. ISSN 1045-9227, 1941-0093. cit. on pp. 15 and 48.

STOKES, J. M. et al. A Deep Learning Approach to Antibiotic Discovery. *Cell*, v. 180, n. 4, p. 688–702.e13, fev. 2020. cit. on p. 15.

TOTH, Z.; KALNAY, E. Ensemble forecasting at NCEP and the breeding method. *Mon. Wea. Rev*, p. 3297–3319, 1997. cit. on p. 28.

VASWANI, A. et al. Attention is All you Need. *Advances in neural information processing systems*, v. 30, 2017. cit. on p. 15.

VELIČKOVIĆ, P. et al. Graph Attention Networks. *arXiv:1710.10903 [cs, stat]*, fev. 2018. cit. on pp. 48 and 53.

WANG, X. et al. Non-local neural networks. In: *IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2018. p. 7794–7803. cit. on p. 37.

WANG, Y. et al. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, ACM New York, NY, USA, v. 38, n. 5, p. 1–12, 2019. cit. on p. 49.

WILLARD, J. et al. Integrating scientific knowledge with machine learning for engineering and environmental systems. *ACM Computing Surveys (CSUR)*, ACM New York, NY, 2021. cit. on p. 16.

WILLMOTT, C. J. On the validation of models. *Physical Geography*, Taylor Francis, v. 2, n. 2, p. 184–194, 1981. Available in: ⟨https://doi.org/10.1080/02723646.1981.10642213⟩. cit. on p. 57.

WU, M. et al. Prediction of short-term wind and wave conditions for marine operations using a multi-step-ahead decomposition-ANFIS model and quantification of its uncertainty. *Ocean Engineering*, v. 188, p. 106300, 2019. cit. on p. 34.

XIAO, D. et al. A reduced order model for turbulent flows in the urban environment using machine learning. *Building and Environment*, v. 148, p. 323–337, 2019. ISSN 0360-1323. Available in: ⟨https://www.sciencedirect.com/science/article/pii/S0360132318306607⟩. cit. on pp. 27 and 28.

XU, T.; VALOCCHI, A. J. Data-driven methods to improve baseflow prediction of a regional groundwater model. *Computers & Geosciences*, v. 85, p. 124–136, 2015. ISSN 0098-3004. Statistical learning in geoscience modelling: Novel algorithms and challenging case studies. Available in: ⟨https://www.sciencedirect.com/science/article/pii/S0098300415001284⟩. cit. on p. 29.