Um estudo de geração de quadros intermediários usando redes generativas para auxiliar artistas de animações tradicionais

Gianluca Takara Ciccarelli

Dissertação apresentada ao
Instituto de Matemática e Estatística
da Universidade de São Paulo
para obtenção do título de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Roberto Hirata Jr.

São Paulo
2 de Fevereiro de 2024

Um estudo de geração de quadros intermediários usando redes generativas para auxiliar artistas de animações tradicionais

Gianluca Takara Ciccarelli

Esta é a versão original da dissertação
elaborada pelo candidato Gianluca
Takara Ciccarelli, tal como
submetida à Comissão Julgadora.

Agradecimentos

Muito mais do que ser apreciada, a arte se tornará indissociável da vida cotidiana[...]. Em outros termos, o Paraíso Terrestre será o Mundo da Arte.
— Meishu-Sama

Agradeço, primeiramente, a Deus, pela vida que recebi, pela permissão de realizar este trabalho, e por todas as graças que me foram concedidas.

A Meishu-Sama, por me mostrar o caminho durante todas as vezes em que tudo era incerto, e por ser, acima de tudo, meu salvador.

A meus pais, por me criarem, educarem, e por estarem ao meu lado em cada momento que precisei.

A Luan Haniel Ferreira Sanches Torres, por ter me motivado a dar início a este trabalho.

A Daniel Carvalho Mendonça e Renan Vinicius de Araújo, pelas conversas, apoio, e inspiração durante esta jornada acadêmica.

A José Eduardo Petruci, pela compreensão durante o tempo que me ausentei do trabalho para a realização desta pesquisa.

A Roberto Hirata Jr. e Nina Sumiko Tomita Hirata, pela paciência, direcionamento, aprendizado, e acima de tudo, por me mostrarem o valor de terminar tudo aquilo a que dou início.

E a todos que contribuíram, direta ou indiretamente, no desenvolvimento deste trabalho.

A todos, minha mais sincera gratidão.

Resumo

Gianluca Takara Ciccarelli. **Um estudo de geração de quadros intermediários usando redes generativas para auxiliar artistas de animações tradicionais**. Dissertação (Mestrado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2024.

A industria de animações tradicionais vem apresentando grande crescimento nos últimos anos. Dentre essas, destacou-se o estilo "anime", tipo de animação produzida primariamente no Japão. Animações tradicionais deste tipo são caracterizadas por uma grande quantidade de quadros desenhados manualmente, e também pela exageração dos traços e aspectos dos elementos presentes em uma cena, frequentemente deixando de representar certos objetos de uma forma mais realística para ressaltar seus principais aspectos. À medida que cresce a demanda por mais animações deste tipo, cresce também a carga de trabalho sobre os artistas encarregados por elas. Na animação tradicional, isso se reflete no aumento da quantidade de quadros que precisam ser desenhados, uma vez que o número de quadros apresentados em cada cena costuma influenciar diretamente na qualidade percebida por quem a vê. Este processo apresenta um desafio para estúdios de animação, uma vez que ele necessita, em grande parte, ser feito à mão, por depender muito da visão artística de quem o desenha. Há também a dificuldade em implementar automações em sua confecção, visto que as ferramentas disponíveis hoje produzem resultados notavelmente diferentes do esperado em animações de grande porte. Nesta dissertação estudamos a viabilidade da geração de quadros intermediários em animações tradicionais utilizando redes neurais. Apresentamos os desafios em treinar uma rede para este propósito, que são a escassez de imagens propícias para sua realização, bem como a impossibilidade de utilizar métodos de interpolação de vídeo já existentes por conta da notável diferença entre desenhos e imagens reais. Explicamos também a metodologia e experimentos realizados, que têm como objetivo analisar os resultados da inferência temporal das imagens, calculada para cada par de quadros na animação original. Para facilitar a aplicação e visualização dos modelos implementados, desenvolvemos um programa de código aberto que realiza a interpolação de quadros intermediários a partir de um conjunto de quadros reais. Esse programa pode inspirar futuros desenvolvedores a criar uma ferramenta profissional para artistas utilizarem em suas criações.

Palavras-chave: Animação tradicional. Rede neural. Interpolação de quadros.

Abstract

Gianluca Takara Ciccarelli. **A study of frame inbetweening using generative networks to assist traditional animation artists**. Thesis (Master's). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2024.

The industry of traditional animations has seen considerable growth in the last years. One type of animation that has made a great impression is the "anime" style, which is primarily produced in Japan. Traditional animations of this kind are characterized by a great amount of hand-drawn frames, and by an exaggerated use of strokes and perspectives in a scene, frequently opting to not present certain elements realistically in order to highlight some of their features. As the demand for this kind of animation grows, the workload of the artists in change follows suit. In traditional animation, this is reflected in an increased number of frames that must be drawn, since the quantity of frames in each scene has a direct correlation with the perceived quality of those who view it. This process presents a challenge for animation studios, as it is predominantly handmade due to its dependency on the artistic vision of those who draw it. It is also difficult to implement automation in its creation, as the tools available today produce results notably different from what is expected by high-quality animations. In this dissertation, we study the viability of using neural networks to generate inbetween frames in traditional animation. We present the challenges in training a network for this purpose, which are the scarcity of valid images for its training, as well as the impossibility of using well-known video interpolation methods due to the notable difference between drawings and real footage. We also explain the methodology and experiments performed, which were used to analyze the results of the temporal image inference, calculated for each pair of frames from the original animation. In order to facilitate the visualization of the implemented models, we have developed an open-source program that can interpolate inbetween frames by using a set of real frames, and that may inspire future developers to create professional tools for artists to use.

Keywords: Traditional animation. Neural network. Frame interpolation.

Lista de Abreviaturas

GAN	Generative Adversarial Network (*Rede Geradora Adversarial*)
cGAN	Conditional Generative Adversarial Network (*Rede Geradora Adversarial Condicional*)
IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo

Lista de Figuras

Lista de Tabelas

Lista de Programas

Sumário

Capítulo 1

Introdução

1.1 Contexto

A indústria de animações tradicionais japonesas, popularmente conhecidas como "anime", vem apresentando grande aumento na sua produção, e na sua demanda (Sherman, 2019). Dentre os motivos para isso, podemos citar as suas características visuais. Estas são distintas da grande maioria das outras animações existentes na indústria, que possuem grande foco no uso de computação gráfica. Apesar de tanto animações tradicionais quanto as de computação gráfica terem méritos e seus próprios desafios para serem produzidas, constata-se que os públicos alvo dos dois tipos são diferentes um do outro, e portanto, cada tipo possui uma demanda para si, mesmo com a existência do outro.

Uma das características da produção de animações tradicionais, que não está presente em animações de computação gráfica, é a criação de quadros desenhados à mão. A quantidade desses quadros por segundo possui grande influência sobre a qualidade de uma animação, o que leva ao desejo de se ter mais desses quadros. O motivo deles influenciarem tanto a qualidade da animação está na forma como cada cena é retratada. Com mais quadros por segundo, mais movimentos podem ser exibidos, e estes ficam mais fluidos para quem os assiste. Definir a quantidade de quadros que serão produzidos, assim como escolher o ângulo, posição, e até mesmo distorções nos objetos da cena, é um trabalho que requer, além de uma visão artística para animação, uma grande quantidade de horas de produção.

Como cada quadro é desenhado à mão, mesmo que utilizando dispositivos de *hardware* e ferramentas de edição de imagem no computador, os artistas realizam um trabalho árduo para desenhar a maior quantidade de quadros possíveis. Dentro de uma rotina de produção em um estúdio, isso também é restrito por prazos e entregas, aumentando ainda mais a pressão sobre seus desenhistas. Apesar de isso ser um problema conhecido desde os primórdios da animação tradicional, até mesmo no estilo ocidental popularizado por *Walt Disney*, os prazos para se completar o desenho eram maiores do que os atuais, por causa da menor demanda internacional.

Apesar do avanço tecnológico, que permitiu com que ferramentas de desenho gráfico fossem incorporadas no processo de produção de animações tradicionais, seus desenhos

continuam sendo feitos primariamente à mão.

Nos últimos anos, também foi observado o aumento de ferramentas que utilizam redes neurais generativas para aumentar o número de quadros por segundo de vídeos e animações, como o DAIN (Bao *et al.*, 2019) e AnimeInterp (Siyao *et al.*, 2021). Apesar dos ótimos resultados, e de sua popularidade para o nicho de edição de vídeo, o objetivo dessas redes é aumentar a resolução temporal destas animações prontas, o que não ajuda na produção destas, além de gerar um resultado visivelmente diferente dos de uma animação tradicional demandada pelo mercado. Isso pode ser visto na Fig. 1.1, que exibe um quadro intermediário criado pela rede DAIN, e possui diversos elementos borrados.

Figura 1.1: *Rede DAIN aplicada em uma animação tradicional para aumentar o número de quadros por segundo. Nesta figura, são exibidos o quadro inicial, o quadro intermediário, e o quadro seguinte. É possível notar diversos elementos borrados no lado direito do quadro intermediário. Figuras extraídas do vídeo "What if Akira was animated at 60 frames per second"(https://www.youtube.com/watch?v=j9-H8DPWTOc).*

A identidade visual criada por animações tradicionais permanece, até hoje, incapaz de ser replicada por processos automatizados.

1.2 Objetivos da dissertação

Os objetivos desta dissertação encontram-se abaixo, e serão detalhados nos capítulos seguintes.

- Estudar a viabilidade de se gerar quadros intermediários através de uma rede neural treinada especificamente para quadros ainda no estágio de produção.
- Criar uma rede que receba dois quadros de animação, e gere o quadro intermediário entre eles.
- Montar um *dataset* com quadros no estágio de produção, apesar da escassez de imagens deste tipo disponíveis publicamente.
- Avaliar o impacto na qualidade das imagens geradas ao utilizar imagens geradas artificialmente para o treinamento da rede.
- Desenvolver e disponibilizar[1] um programa *open source* que pode ser rodado em computadores *low-end* que simplifique a geração de quadros intermediários através da rede treinada nesta dissertação, e que dê ao artista a liberdade de gerar os quadros que ele achar relevante.

[1] https://github.com/JanValJanos/FrameGen

1.3 *Dataset*

Um dos grandes desafios enfrentados durante este trabalho foi a criação de um *dataset*. No campo de desenhos em estilo *anime* existem diversos *datasets* públicos disponíveis, como o *DANBOORU* (Anonymous *et al.*, 2021) e o *Anime Face Dataset* (Churchill e Chao, 2019). Isso facilita o treinamento de redes neurais que possuem o propósito de atuar sobre imagens de *anime*, como na edição de imagens para transferência de estilo (Zhang *et al.*, 2017) e colorização de desenhos (Ci *et al.*, 2018). No entanto, os *datasets* atuais de *anime* são usados apenas para treinar, validar e testar redes relacionadas a desenhos estáticos. Ao entrar no campo de desenhos animados, como a animação tradicional em estilo *anime*, somente este tipo de dados passa a não ser suficiente, visto que, além de ser necessário gerar corretamente os traços do desenho, também é necessário utilizar informações temporais da animação para definir como cada elemento da cena deve estar disposto no quadro gerado.

Apesar de conjuntos de quadros, ou *frames*, serem criados em grandes quantidades em estúdios de animações tradicionais, a realidade é que a grande maioria destes nunca é disponibilizada gratuitamente ao público. As que são, muitas vezes estão escaneadas de forma que as impossibilita de serem utilizadas para o treinamento de uma rede generativa para animações. Além disso, a etapa de produção da animação em que o artista precisa manualmente desenhar os quadros, chamada na indústria de anime de *douga*, possui somente contornos, sem preenchimento de cores ou adição de efeitos gráficos, o que os torna visivelmente diferentes de uma animação finalizada, criando outra barreira na obtenção de dados reais.

Em se tratando da estimação de quadros intermediários em animação, também se deve considerar o estilo de "animação limitada"(Eberle, 2015) empregado por animes, que utiliza quadros de impacto para simular movimentos rápidos, e que possuem pouca continuidade visual com os quadros vizinhos (como pode ser visto na Figura 1.2).

Cenas deste tipo seriam prejudiciais ao processo de treinamento pois a visão do artista foge da representação contínua e real do movimento para dar expressividade à sua arte, e portanto, se torna imprática para ser estimada por uma rede que conta com tal continuidade.

Figura 1.2: *Exemplo de um anime que usa de técnicas de animação limitada, espaçando os momentos da cena entre os quadros para causar uma impressão maior de impacto e ação. Imagens extraídas de SAKUGABOORU.*

Por conta de todos estes fatores, a montagem do *dataset* de *frames* de animação se tornou um dos principais desafios nesta dissertação. Para realizar esta montagem, foram pesquisados três métodos para gerar o *dataset*. O primeiro método utiliza uma rede neural auxiliar para converter quadros originados de animações finalizadas para a etapa *douga* da

produção, ou seja, possuindo apenas contornos. Isso permite a extração de dados de treino de animações disponíveis publicamente, que existem em abundância, e resolve o problema da escassez de *frames douga* de boa qualidade.

Com relação à descontinuidade visual frequentemente empregada nas cenas de animes, foi aplicado o segundo método na montagem do *dataset*, que é composto pela seleção à mão das sequências de quadros que possuam uma continuidade clara. Isso foi feito utilizando um conjunto de dados de animações finalizadas, e montando um *dataset* somente com aquelas que possuam um movimento que possa ser interpolado.

Finalmente, como o treinamento da rede foi feito utilizando quadros de animação somente com contornos, o terceiro método se resume a utilizar também um conjunto de imagens adicionais simuladas. Estas animações são compostas por formas geométricas simples, como retângulos e círculos, que possuem apenas contornos, e realizam movimentos definidos programaticamente. Com esse aumento do *dataset*, treinamos a rede para inferir mais tipos de movimento, sem necessariamente eles estarem presentes no conjunto de animações reais fornecidas inicialmente.

Unindo os métodos de conversão para *douga*, seleção manual de cenas, e geração de imagens simuladas, foi possível montar um conjunto de dados pertinente à proposta de treinar uma rede que possa auxiliar animadores a criarem quadros *douga*.

1.4 Rede de geração de quadros

A solução estudada nesta dissertação baseia-se no uso de uma rede neural que recebe dois quadros de uma sequência de animação e gera um quadro intermediário, posicionado entre os dois quadros de entrada.

A rede segue um modelo de rede geradora adversarial convolucional *CGAN* (Isola *et al.*, 2016). Este modelo permite a extração das características das duas imagens de entrada, que então são sintetizadas em uma nova imagem, conforme visto na Figura 1.3. Para fazer uso do máximo de informações possíveis dessas duas imagens, a rede geradora segue o modelo U-Net (Ronneberger *et al.*, 2015), que utiliza *skip connections* que evitam a perda de informações à medida que as imagens passam pelos *kernels* das camadas internas da rede.

A rede discriminadora possui três entradas, que são as duas imagens usadas na geradora, e mais uma, que é o quadro intermediário entre elas. Isso permite que o conjunto da rede seja treinado utilizando triplas de quadros da animação, com o primeiro e terceiro quadros sendo as entradas da rede geradora, e o segundo quadro sendo o *ground truth* com o qual o quadro gerado deve ser comparado. A saída da rede discriminadora devolverá o quão essa tripla se assemelha a uma sequência de quadros em uma animação, avaliando a similaridade estrutural entre os três quadros, e também a continuidade visual entre eles.

Para permitir que a rede criada nesta dissertação seja utilizada por artistas que não estão familiarizados com estes conceitos, foi desenvolvido um programa em Python que utiliza a rede neural para gerar os quadros intermediários que o artista deseja.

Este programa deve servir como uma ferramenta complementar para o animador, que

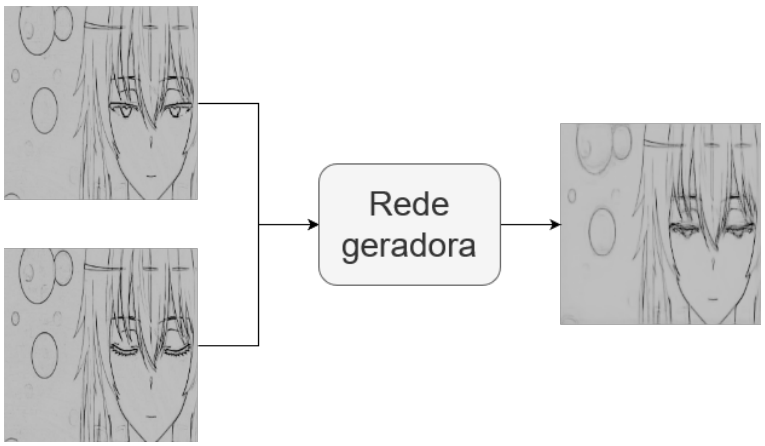


Figura 1.3: *Fluxograma simplificado do funcionamento da rede. Esta recebe dois desenhos como entrada, e gera um quadro intermediário a partir deles.*

poderá gerar quantos *frames* precisar, e somente nos intervalos que desejar. O programa conta com uma interface simples, que permite com que o usuário importe a sequência de quadros, navegue neles para o ponto onde deseja gerar novos quadros, e então gerar quantos achar necessário. Com os novos quadros, o usuário pode então prever a animação, e depois exportar estas imagens para arquivos novamente.

O programa também conta com um modo de comparação, que permite que o usuário compare, para cada tripla de *frames* da animação, o quadro intermediário real com o gerado pela rede.

1.5 Dissertação

Esta dissertação é composta da seguinte forma: No Capítulo 2, Fundamentos, serão revisados os principais conceitos utilizados, que tanto no domínio do problema (o desenho de quadros na produção de *animes*), e no da solução (a utilização e treinamento de uma rede neural). No Capítulo 3, são listados alguns estudos realizados no campo do aprendizado de máquina que estão relacionados à esta pesquisa. No Capítulo 4, será detalhada a pesquisa, o que foi realizado, e o raciocínio por trás das decisões feitas. No Capítulo 5, serão exibidos os resultados do método aplicado no capítulo anterior, assim como medidas de avaliação empregadas. Por fim, no Capítulo 6, considerações finais sobre o cumprimento dos objetivos propostos.

Capítulo 2

Fundamentos

Neste capítulo apresentaremos conceitos relevantes para esta dissertação: primeiramente, a definição e conceitos básicos de animação, para contextualizar a indústria sob a qual esta pesquisa atua; em seguida, as duas medidas utilizadas para avaliar as imagens geradas por este estudo; e por fim, conceitos relevantes sobre rede neurais, que são a base para a realização desta pesquisa.

2.1 Animação

Nesta seção, definimos o que é uma animação, algumas formas de produção, como isso tem sido utilizado na indústria, e conceitos relevantes para o entendimento do problema que será tratado nos capítulos seguintes.

2.1.1 Definição

Uma animação é definida como um conjunto ordenado de imagens, chamadas de "quadros", que quando exibidos em sequência criam uma ilusão de movimento ao espectador (Kerlow, 2004). Dentre as diversas formas de animação que hoje existem, destacam-se a *animação tradicional*, que é composta por quadros desenhados a mão por um artista, e a *animação computadorizada*, composta por quadros renderizados por computador a partir de um modelo matemático.

A produção de cada um dos tipos possui vantagens e desvantagens, porém a diferença mais notável entre as duas é o aspecto da animação final. Na Figura 2.1, temos dois exemplos de animação, uma tradicional 2D (esquerda), e uma outra computadorizada 3D (direita). Mesmo examinando apenas um quadro de cada uma, é possível notar que animações 2D e 3D resultam em aspectos muito diferentes um do outro, com a animação 2D notavelmente apresentando áreas de cores mais uniformes, bordas bem definidas, e um plano de fundo feito desenhado de maneira diferente para que o movimento do primeiro plano seja ressaltado. Já na animação 3D, suas cores são muito mais afetadas pela iluminação, reflexão e posição dos objetos, e cada elemento da cena é um modelo que pode ser animado separadamente. Estas e outras diferenças são comumente o fator principal na escolha de qual tipo de animação o artista deseja utilizar.

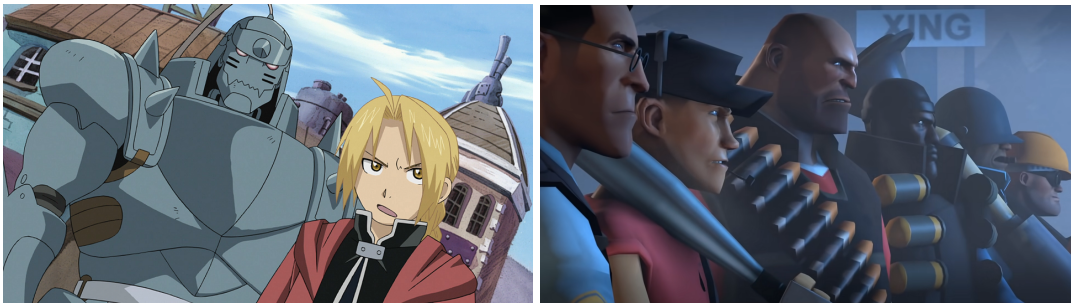


Figura 2.1: *Na esquerda, um exemplo de animação tradicional, e na direita, um exemplo de animação computadorizada. Figuras retiradas do anime "Fullmetal Alchemist: Brotherhood", e da animação "Mann vs. Machine"*

Artistas que criam animações computadorizadas utilizam programas para criar modelos (tanto 2D quanto 3D), e através dos programas, conseguem modificar e posicionar estes modelos com liberdade, além de ser possível alterar sua textura, luminosidade, e mais importante, seu movimento. Ao serem renderizados, os programas geram os quadros automaticamente conforme definido pelo artista. O ponto de vista do espectador é definido através de um ponto chamado de "câmera", e a perspectiva dos objetos em cada quadro é renderizada tomando-se em conta esta câmera, para que o espectador possa enxergá-los da forma que o animador pretendeu. Enquanto o animador possui a facilidade de interpolar o movimento destes objetos, ele também precisa definir o movimento de diversas partes de um mesmo objeto para criar a animação da forma que planejou.

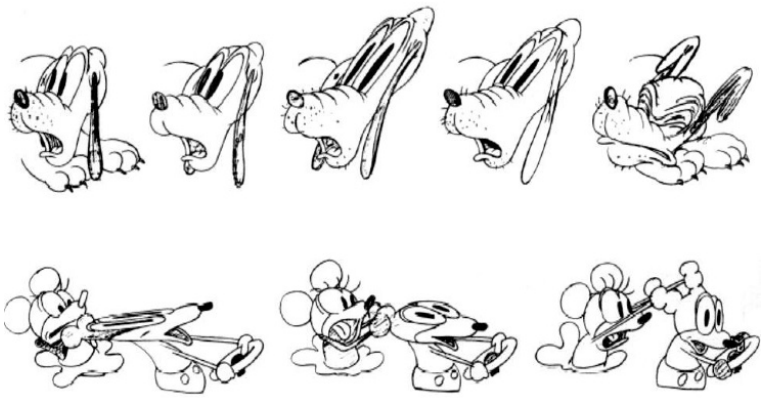


Figura 2.2: *O princípio de squash and stretch, um dos fundamentos da animação tradicional, é utilizado para expressar e exagerar movimentos em animações. Desde animações clássicas até animações modernas, este princípio é considerado um dos mais característicos deste tipo de animação, dando vida e dinamismo a seus quadros. Imagem adaptada do livro "The Illusion of Life".*

Artistas de criam animações tradicionais desenham cada quadro a mão, o que engloba tanto desenhos feitos em papel ou celuloide quanto desenhos feitos com auxílio de um programa de computador. O artista tem a liberdade para escolher como a perspectiva dos objetos de uma cena é exibida para o espectador, podendo inclusive deformá-los exagerando-se o movimento que está sendo realizado (Williams, 2009), o que seria difícil de realizar através de um modelo matemático. Como é possível visualizar na Fig. 2.2, muitas destas deformações exageradas não obedecem conceitos observados na vida real, sendo

caracteristicamente extrapolados para dar mais vida à cena. Em especial, na Fig. 2.2, é possível notar como as características faciais, principalmente os olhos, são esticadas além do normal, e como isso gera expressividade para o desenho.

A animação tradicional é difícil pois um grande número de quadros precisa ser desenhado à mão para criar uma animação de qualidade. Mesmo com os avanços tecnológicos desde sua concepção, a etapa de desenho dos quadros continua sendo uma árdua tarefa feita por artistas. Apesar desse trabalho servir como meio para os desenhistas se expressarem na sua obra, ele também tem forte impacto na dificuldade, tempo, e custo de produções maiores.

2.1.2 Produção

O foco deste estudo são animações japonesas, popularmente conhecidas como *anime*. Nesta seção serão introduzidos conceitos relevantes à produção de *animes* que servirão como base para o objetivo do trabalho desenvolvido.

O estilo de animação aplicado em animações japonesas possui uma divisão clara de papeis a serem cumpridos para alcançar o resultado desejado na animação. A Figura 2.3 contém um fluxo simplificado de produção de um quadro de animação neste estilo. Nela é possível ver o *storyboard*, a primeira etapa da produção do desenho, que contém apenas esboços representando parcialmente a posição e ângulo dos objetos na cenas. Esta serve apenas de referência para as etapas seguintes terem a base de onde começarem os desenhos reais. Eles não são usados para compor uma animação, mas sim visualizados lado-a-lado para que os artistas ganhem melhor noção de como cada cena irá acontecer. Na etapa seguinte vista na Figura 2.3 , a etapa de *layout*, alguns quadros chaves já são desenhados com mais definição, e estes são usados para definir os principais momentos do movimento e fluxo da animação.

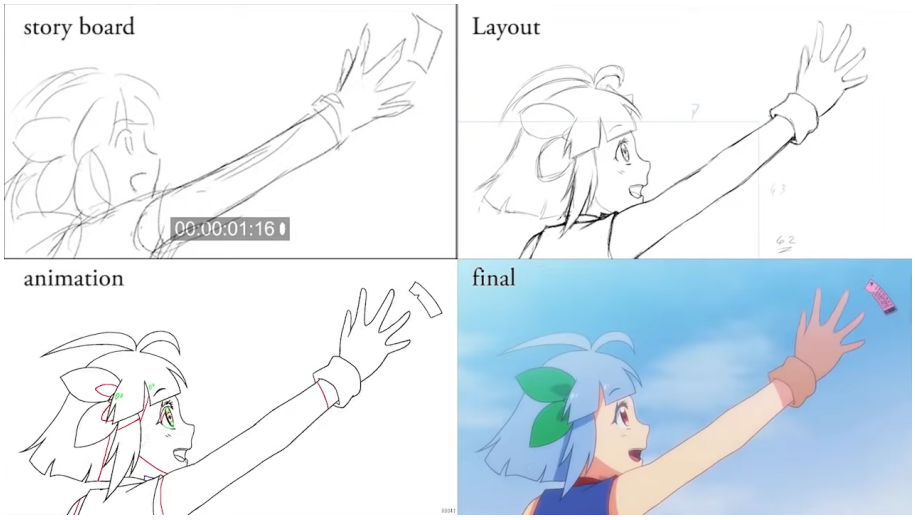


Figura 2.3: *Um mesmo quadro de anime, passando pelas diversas etapas de produção. É possível notar como na etapa de animation (douga) os contornos já estão da forma que aparecem no produto final. Figura extraída do vídeo "BokuraNoTabi making of"(https://www.youtube.com/watch?v=8hLPrpGtJx8)*

A etapa seguinte, *animation*, já possui os contornos reais que serão utilizados na ani-

mação final. Todos os quadros, inclusive os quadros intermediários (quadros que fazem as ligação entre os quadros chaves), já são desenhados desta forma. E por fim, o resultado final (indicado na figura pela *label final*), já possui colorização, efeitos de pós-processamento (como efeitos degradé nas cores) e o plano de fundo, desenhado separadamente.

Dentro da produção de animação, existem duas etapas que não são explicitamente visualizadas na figura que valem ser ressaltadas. São elas a produção de quadros chaves (*keyframes*), e de quadros intermediários (*inbetween frames*). Primeiramente, vê-se necessário esclarecer um ponto importante sobre estes termos. Na área de compressão de vídeos, estes termos assumem um significado diferente dos que são utilizados na indústria de animação. Os termos *keyframe* e *inbetween frame* (ou *interframe*), em métodos de compressão como o MPEG, se referem a formas diferentes de se armazenar os quadros em um arquivo de vídeo, distinguindo quadros que são guardados por completo, ou aqueles que apenas guardam as alterações nos píxeis com relação ao quadro anterior.

No universo das animações tradicionais, os quadros chave, ou *genga*, são quadros desenhados que são compostos apenas por contornos, e descrevem os momentos principais de cada cena. Eles são mais trabalhados do que o que foi desenhado no *storyboard*, e nos quais já é possível visualizar o aspecto da animação final, além de incluir muito mais detalhes do que as etapas anteriores. Através deles, animadores experientes estabelecem as bases para o fluxo da animação final, e utilizam de seu conhecimento para representar da melhor forma cada objeto na cena. Apesar do alto nível de detalhes, estes quadros aparecem em menor número, e sozinhos não são o bastante para passar uma ilusão de movimento convincente. Por isso, o passo seguinte é o desenho dos quadros *inbetweens*, ou *douga*, que têm como propósito completar o espaço temporal entre os *keyframes* e fazer a movimentação parecer mais natural. Nesta etapa, busca-se criar um grande número de *inbetweens* para criar uma animação de maior qualidade, e compara-se quadros consecutivos, como visto na Figura 2.4, para criar o quadro intermediário. Nesta figura, o animador utiliza um programa de computador que sobrepõem dois quadros consecutivos, para visualizar o movimento sendo realizado na animação, e então começa a desenhar um novo quadro para preencher o vazio neste movimento.

Na produção de animações tradicionais, o espaçamento e a quantidade de quadros são utilizados para retratar diversos aspectos da cena. Dentre os Doze Princípios Básicos da Animação (Thomas e Johnston, 1981), o princípio de *Timing* denota que as sensações de velocidade e impacto de uma ação são influenciadas pelo número de quadros utilizados para descrevê-la.

Para movimentos mais rápidos, a prática comum é expressar o movimento usando menos quadros, cada um representando posições com maior espaçamento. Já movimentos mais lentos são normalmente desenhados com mais quadros e menor espaçamento entre os momentos retratados.

Apesar dos animadores buscarem desenhar um número satisfatório de quadros por segundo, esta é uma tarefa muito custosa, por exigir que cada quadro seja desenhado à mão. Por isso, vê-se o interesse de estudar um método computacional que animadores possam utilizar para facilitar este processo, sendo uma ferramenta que toma proveito da similaridade entre cada par de quadros para gerar quadros intermediários.

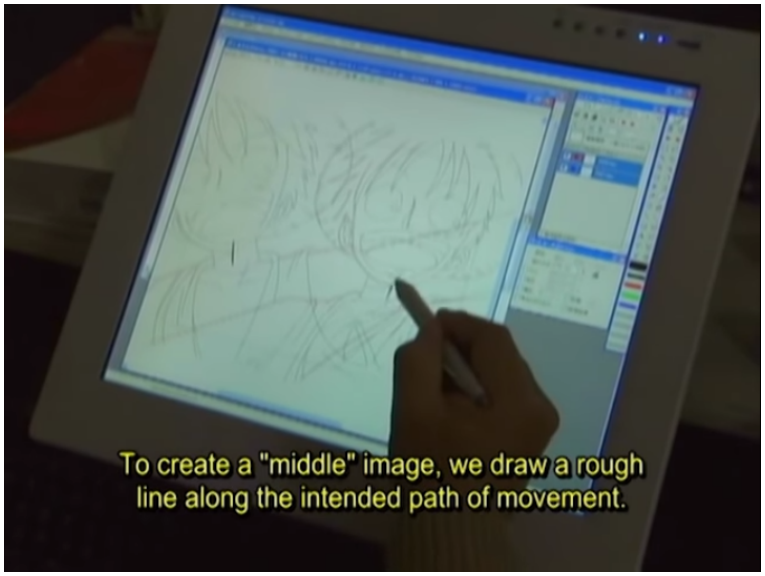


Figura 2.4: *Animador comparando dois quadros para fazer o desenho de um quadro intermediário. Imagem extraída do vídeo "Inside Toei Animation Part 1"(https://www.youtube.com/watch?v=Q4ByyRYgyZw)*

2.1.3 Indústria

Enquanto estúdios são responsáveis pela produção de animações, o mercado de *animes* envolve outros grupos. Sua exibição é feita através de canais de televisão, ou de serviços de *streaming*, ambos os quais possuem contratos com as empresas donas das animações. A maioria dos canais essão situados no próprio Japão, enquanto serviços de *streaming* como *Crunchyroll* e *Netflix* podem ser acessados mundialmente. Estes últimos começaram a fazer parte ativamente da indústria nos últimos 10 anos, em que a demanda de *animes* por espectadores fora do Japão impulsionou seu crescimento.

Graças ao aumento da popularidade e demanda, o mercado de *animes* no Japão está atingindo valores recordes históricos (Sherman, 2019). Esse crescimento permitiu com que diversos estúdios, como *MAPPA* e *ufotable*, crescessem, consequentemente aumentando também o número de *animes* lançados anualmente e o número de vagas para animadores, tanto para iniciantes quanto para profissionais.

Apesar do lucro recorde da indústria, o salário de animadores permanece muito baixo para os padrões do país, especialmente para animadores de *inbetweens* (Margolis, 2019). Além disso, devido à saturação do mercado fazendo com que muitas produções ocorram simultaneamente, animadores também enfrentam longas jornadas de trabalho e prazos apertados.

2.2 Medidas de comparação de imagens

Neste seção, serão apresentadas as definições das duas medidas de comparação entre imagens utilizadas neste trabalho: PSNR (Horé e Ziou, 2010) e SSIM (Z. Wang *et al.*, 2004).

A *PSNR* (*Peak Signal-to-Noise Ratio*) é uma medida utilizada para calcular a interferência por parte de ruídos em sinais. Ela é aplicada no contexto de visão computacional, sendo usada para comparar imagens e suas representações. Uma de suas aplicações é a análise de imagens após passar por algum algoritmo de compressão, sendo capaz de medir o quanto essa compressão afetou a imagem a nível de píxel. Ela é calculada com base na diferença pixel-a-pixel entre duas imagens de tamanho MxN através da seguinte fórmula:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} [I(i,j) - K(i,j)]^2} \right)$$

Onde MAX é o valor máximo que cada píxel pode assumir. Na representação RGB, este valor é 255 para cada canal. A identificação de distorções na imagem por conta de ruídos vem justamente da comparação pixel-a-pixel, deixando de lado outras análises estruturais da imagem.

A *SSIM* (*Structural Similarity Index*), por sua vez, faz uso de informações estruturais das imagens para fazer a comparação das imagens. Assumindo valores entre -1 e 1, sendo -1 a maior diferença possível entre as imagens, e 1 o valor que indica que as duas imagens são iguais, esta medida foi desenvolvida para tomar em conta a dependência entre píxeis de cada região das imagens. Fatores como a diferença no contraste e intensidade luminosa entre as imagens afetam o valor desta métrica.

A SSIM entre duas imagens x e y, de mesmo tamanho, é calculada utilizando a seguinte fórmula:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + (k_1L)^2)(2\sigma_{xy} + (k_2L)^2)}{(\mu_x^2 + \mu_y^2 + (k_1L)^2)(\sigma_x^2 + \sigma_y^2 + (k_2L)^2)}$$

Onde μ_x e μ_y são a média dos píxeis de suas respectivas imagens; σ_x e σ_y são o desvio padrão; σ_{xy} é a covariância entre as imagens; L é o valor máximo que cada pixel pode assumir (neste caso, 255); e k_1 and k_2 são duas constantes utilizadas para evitar valores muito altos quando o denominador é próximo de zero, e por padrão são utilizados $k_1 = 0.01$ e $k_2 = 0.03$.

2.3 Aprendizado de máquina

Nesta seção, definimos o que é aprendizado de máquina, suas origens, e os conceitos que são relevantes para a pesquisa desenvolvida nesta dissertação. Entraremos em detalhe na área de redes neurais, dentro da qual está presente o modelo de rede de análise de imagem que utilizamos.

2.3.1 Definição

Enquanto o desenvolvimento de programas tradicionais faz com que o computador execute os comandos definidos explicitamente pelo programador, a área de aprendizado de máquina busca desenvolver métodos para que o programa seja capaz de alterar seu

comportamento baseado nas informações que lhe são fornecidas. Isso permite com que ele "aprenda"com o conjunto de dados, e se torne capaz de tomar decisões que não foram definidas durante sua criação.

Apesar do campo ter ganhado reconhecimento geral apenas nos últimos anos, sua teorização e desenvolvimento se originaram muito antes. A técnica de *back propagation* (Plaut *et al.*, 1986), base para a utilização de redes neurais, teve seu artigo publicado em 1986.

Primeiras implementações de aprendizado de máquina em computadores modernos, como a *Support-Vector Machine* (SVM) (Cortes e Vapnik, 1995), já tiveram seus artigos publicados em 1995. Modelos ainda mais teóricos e com implementações feitas em computadores muito mais antigos, em especial o *perceptron*, teve sua primeira implementação feita em 1958.

Estes modelos foram base para o aprendizado de máquina moderno. Gradualmente, os usos de aprendizado de máquina para solucionar problemas que antes se diziam impossíveis se tornou o padrão. Apesar de modelos de aprendizado de máquina ainda demorarem para encontrar uso comercial, algumas conquistas notáveis começaram a surgir. Dentre elas, destaca-se a vitória do *DeepBlue*, desenvolvido desde 1986 pela IBM, contra o campeão mundial de xadrez Garry Kasparov em 1997 (Greenemeier, 2017), algo nunca feito antes por conta da da quantidade de jogadas possíveis a serem analisadas.

Com a revolução de *hardware* na década de 2000, que permitiu tanto o armazenamento de grandes quantidades de dados por conta de discos rígidos maiores e mais acessíveis, e também crescimento de processadores, em especial de placas de vídeos, foi crucial para a evolução dos métodos de aprendizado de máquina, como será visto no próximo capítulo.

2.3.2 Redes neurais

O modelo de redes neurais é uma aplicação de aprendizado de máquina que se baseia no uso de camadas com nós, cada um capaz de armazenar uma variável numérica, e que possuem conexões com os nós das camadas anterior e seguinte. Essas conexões entre nós criam uma rede que recebe uma entrada, que é atravessada pelas diversas camadas ocultas da rede, até chegar na última camada, que é onde é gerado o resultado do processamento.

O uso de múltiplas camadas tem como objetivo extrair diferentes níveis de características da entrada. À medida que a entrada original passa pelas camadas, características de nível cada vez mais detalhado são extraídas. Utilizando como exemplo uma imagem, é possível interpretar que as primeiras camadas de uma rede neural realizam operações similares às que já eram feitas a métodos já existentes de visão computacional, como a detecção de regiões e cantos, enquanto que as últimas camadas já estariam sintetizando informações de nível interpretativo, como reconhecer objetos na imagem. Na Figura 2.5, é possível visualizar uma rede deste tipo, em específico uma rede neural convolucional, feita especificamente para o processamento de imagens. Nesta figura, temos 1 camada de entrada, com 8 nós, 3 camadas ocultas, cada uma com 9 nós, e 1 camada de saída, com

4 nós. O nó de cada camada está ligado com todos os nós da anterior, e todos os nós da próxima.

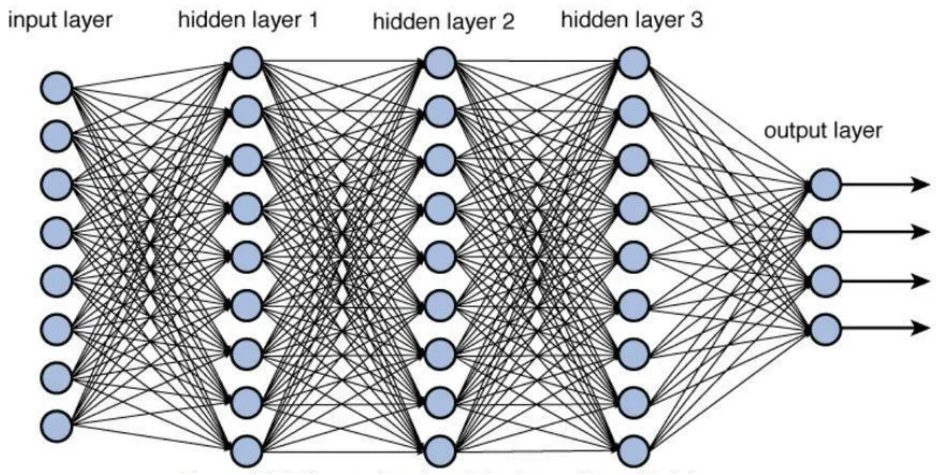


Figura 2.5: *Exemplo de uma rede neural com diversas camadas internas. É possível visualizar como cada nó utiliza todos os nós da camada anterior como entrada, e como isso se repete até as camadas de saída.*

O conceito de redes neurais, assim como muitos outros de aprendizado de máquina, teve suas origens muito antes do seu uso ser popularizado (Schmidhuber, 2022). Modelos de *perceptrons* de múltiplas camadas, que utilizam nós e treinamento iterativo, já podiam ser observados em trabalhos como os de Amari, 1967 e Ivakhnenko e Lapa, 1967. Foi somente com o desenvolvimento de GPUs no final da década de 2000 que redes neurais ganharam um uso prático, visto que GPUs mais potentes permitiam a implementação de redes neurais melhores.

Estas técnicas são amplamente utilizadas na atualidade, sendo capazes tanto de tomar decisões como classificação de imagens (Lecun *et al.*, 1998) e recomendações de conteúdo, assim como geração de imagens (Isola *et al.*, 2016) e vídeos com base em algum parâmetro, como mudança de estilo de pinturas (Zhang *et al.*, 2017) e alteração de traços faciais em fotos.

Para alcançar resultados como estes, são necessárias algumas etapas na confecção do mesmo. Primeiramente, ao se tratar de um programa que irá tomar decisões que não foram definidas pelo desenvolvedor, entra o conceito de "treinamento", ou seja, a repetição de execuções do programa que alteram seu comportamento para tomar melhores decisões ao longo do tempo.

A definição dos valores que cada ligação possui, e sua atualização à medida que várias iterações do programa são executadas, é o que é chamado de "treinamento", e é responsável por ajustar a rede neural até que os resultados vistos na última camada estejam de acordo com o propósito do seu desenvolvimento.

O valor de cada nó é resultado de uma operação numérica realizada com cada nó da camada anterior. Isso permite com que diferentes pesos sejam dados para a conexão com cada um desses nós, dando mais valor, tanto positivo quanto negativo, para cada ligação.

Como definir o valor de um nó através da soma e multiplicação dos nós anteriores levaria a resultados muito lineares, algo que nem sempre irá refletir o que precisa ser interpretado na entrada, são utilizadas também funções de ativação, que são funções aplicadas sobre o valor de cada multiplicação de nós para introduzir a não-linearidade dos resultados. Funções como a *rectifited linear unit* (ReLU) (Fukushima, 1969) ou a tangente hiperbólica são popularmente utilizadas por conta da sua não-linearidade, e também por restringirem os resultados com limite superior ou inferior. Este último permite com que valores numéricos muito grandes sejam evitados.

O treinamento é feito utilizando um conjunto de dados, ou *dataset*, com exemplos de casos reais sob os quais o programa fará previsões, e se ajustará conforme ele acerta ou erra essas previsões. Este conjunto é montado cuidadosamente, buscando conter casos relevantes para o programas, balancear seus dados para refletir o conjunto real, normalizar suas representações numéricas para que diferentes casos se tornem comparáveis e mensuráveis pelo programa, entre outras técnicas, todas com o propósito de adequar o conjunto de dados ao programa para que este extraia as características corretas.

Durante o treino, além da estrutura também são definidos hiperparâmetros, que são ajustados manualmente pelos desenvolvedores para alterar o comportamento do programa. Isso permite com que pesquisadores tenham maior controle sobre o que está ocorrendo na etapa de treino, visto que frequentemente o modelo de rede neural pode assumir um tamanho que se torna difícil de regular apenas através da arquitetura. Estes hiperparâmetros incluem fatores como a taxa de aprendizado, que define o quanto cada nó é alterado em cada etapa do treino, e a função de ativação de cada ligação, que permite normalizar as alterações feitas sobre cada valor.

Combinando estas etapas, é possível treinar redes neurais para os mais diversos usos, contanto que se tenha um conjunto de dados bem montado e uma estrutura bem definida. Com isso, aos poucos o treino irá atualizando os valores dos nó e conexões da rede neural, até que as suas saídas se tornem cada vez mais condizentes com os valores de entrada.

2.3.3 Rede Neural Convolucional

O modelo de rede neural convolucional, ou *CNN* (*Convolutional Neural Network*), toma como base os princípios estudados na área de redes neurais, com uma mudança fundamental na estrutura da rede. Os nós, que cada um antes possuía apenas 1 valor numérico, agora possuem um *kernel*, ou filtro.

O *kernel* é o mesmo utilizado na operação matemática de convolução, o que dá o nome para este modelo. Cada nó armazena um *kernel* de tamanho fixo, de valores que são usados na operação de convolução com a imagem dos nós anteriores. Os valores que compõem os *kernels* de cada camada são atualizados durante o treino, com o propósito de serem usados para extrair características locais e globais da imagem.

As redes neurais convolucionais também lidam com o conceito de canais. Imagens são representadas por três canais de cores *RGB* (vermelho, verde e azul). Ao trazer isso para o campo de redes neurais, cada camada da rede passa a ser representada por um conjunto de canais. O número de canais é equivalente ao número de *kernels* que a camada da rede possui. Assim, ao estruturar uma rede neural convolucional, é comum ver que a entrada

é composta por três canais (ou múltiplos disso, caso mais de uma imagem seja usada), e esse número de canais aumenta gradualmente à medida que as camadas ocultas da rede são atravessadas, até que esse número passa a diminuir ao se aproximar da camada de saída, a qual por sua vez possui novamente três camadas, compondo assim uma nova imagem.

Redes neurais convolucionais também utilizam técnicas para diminuir o tamanho (altura e largura) da imagem nas camadas internas, chamado de *downsampling*. A operação básica para isso é chamada de *pooling*, que compara conjuntos de píxeis nos canais da camada anterior, e retorna um valor para cada conjunto. Um exemplo disso é o *maxpooling*, que compara blocos de valores, e retorna o máximo entre eles, diminuindo então o tamanho da imagem por um fator de N. Isso é útil para diminuir o custo computacional das operações feitas na rede, assim como permitir a extração de características em níveis de abstração diferentes.

A própria operação de convolução já oferece uma forma de realizar o *downsample* da imagem, através do *stride*. Ao se realizar uma convolução, é possível escolher o espaçamento entre os píxeis de cada operação. Ao mover o *kernel* pela imagem original, é possível fazê-lo se mover pixel por pixel, ou pular um pixel e seguir para o próximo. O número de píxeis movidos é o *stride*, e *strides* com valor maior do que 1 diminuem o tamanho da imagem resultante pelo mesmo fator.

Após as camadas de *downpooling*, é comum ver camadas de *uppooling*. Seu processo é análogo aos do anterior, aumentando o tamanho da imagem ao invés de diminuir. Isso é usado para reconstituir uma imagem do mesmo tamanho da entrada, visto que o *downpooling* e convoluções diminuíram seu tamanho afim de extrair suas características.

A Figura 2.6 mostra uma rede neural convolucional (LeCun *et al.*, 1989) , específica para lidar com imagens, que utiliza várias camadas de *downpooling*, até retornas às camadas de *uppooling*. Nela é possível também visualizar os conceitos até aqui mencionados. O tamanho da imagem sendo diminuído e aumentado à medida que as camadas internas são atravessadas. O número de canais também variando, representado na imagem pela grossura de cada bloco, e também as conexões que são feitas entre camadas de diferentes níveis da rede.

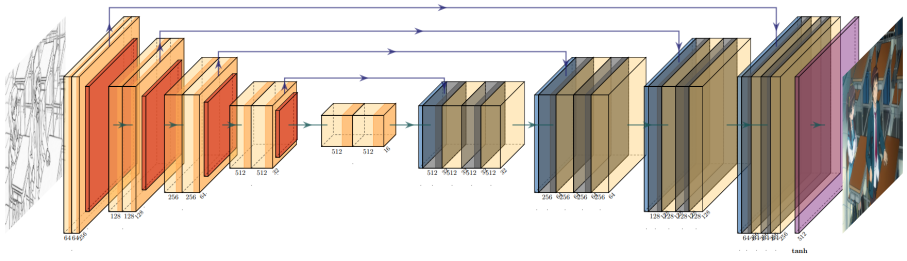


Figura 2.6: *Visualização de uma rede neural convolucional com diversas camadas internas. Neste caso, a visualização é de uma rede para processar imagens, e portanto suas camadas internas podem ser interpretadas como blocos de imagens de tamanho definido, e um certo número de canais.*

Redes neurais convolucionais também possuem camadas completamente ligadas, chamadas de *fully connected*. Estas se assemelham a uma rede neural normal, ou seja, os

valores das camadas anteriores passam por uma função de ativação para nós com valores únicos, que não são *kernels*. Isso permite com que o resultado final da rede possa ser um valor numérico, e não uma imagem. Seu uso mais notável é a classificação de imagens, onde é possível ter diversos nós no final da rede, cada um representando uma *label*, e após passar por camadas de convolução, a rede utiliza as características extraídas para tomar a decisão de qual *label* associar à imagem de entrada.

Um de seus primeiros usos foi para o reconhecimento de dígitos escritos à mão (LeCun *et al.*, 1989), já utilizando conceitos de *backpropagation* e utilização de *kernels* para fazer o reconhecimento de imagens. Os conceitos usados em redes neurais convolucionais de *kernels* e *downsampling* já podiam ser observados muito antes no "neocognitron"(Fukushima, 1980), considerado hoje como a inspiração para CNNs modernas. Hoje, o reconhecimento de escrita é amplamente realizado utilizando novos modelos de redes, como redes totalmente convolucionais (FCN) (Such *et al.*, 2019).

Atualmente, redes neurais convolucionais são amplamente usadas em campos da indústria que trabalham com *machine learning* e imagens. A classificação de objetos aplicada em carros autônomos (Balasubramaniam e Pasricha, 2022); o reconhecimento facial (Freitas Pereira *et al.*, 2022), comum em autenticação de usuários; a segmentação de objetos dentro da imagem (Long *et al.*, 2015), vista em programas de edição de imagem e de assistência médica; a descrição de imagens (Sultana *et al.*, 2018), usadas para acessibilidade; a geração e transformação de imagens baseada em parâmetros do usuário (Isola *et al.*, 2016), agora utilizadas para geração de pinturas; a interpolação de vídeos (Kang *et al.*, 2020), permitindo o *upsampling* dos quadros sem que eles precisem ser gerados pelos meios originais. Todos esses são alguns dos usos sobre os quais CNNs são aplicadas hoje.

Dentre os modelos de CNNs estudados, o modelo *GAN* será descrito no capítulo seguinte, dado a sua relevância para esta dissertação, e pela sua contribuição para a indústria de geração de imagens artificiais.

2.3.4 Rede Geradora Adversarial

O modelo de rede neural utilizado neste trabalho é conhecido como *Generative Adversarial Network*, ou *GAN* (Goodfellow *et al.*, 2014). Este modelo tem como base o treinamento paralelo de duas redes, chamadas de geradora (*G*) e discriminadora (*D*).

A rede discriminadora *D* é uma rede classificadora treinada para discernir imagens reais de imagens geradas artificialmente. Isso é feito utilizando um conjunto de treino com imagens reais, que a rede é treinada para reconhecer como sendo reais, e imagens artificialmente geradas pela rede geradora, que são classificados como tal. Esta rede recebe como entrada uma imagem, e após algumas camadas de convoluções, ela possui camadas completamente ligadas, que então levam até uma última camada com apenas um nó, indicando se a imagem de entrada é real ou artificial.

A rede geradora *G*, em seu desenho original, recebe um vetor latente e é treinada para gerar uma imagem a partir dele. O vetor latente, neste contexto, é um vetor composto por valores amostrados de uma distribuição probabilística, tendo como propósito induzir aleatoriedade nas imagens geradas. A qualidade dessa imagem é medida comparando-a

com as imagens do conjunto de imagens reais, e com a avaliação da rede discriminadora. Esta rede possui camadas completamente ligadas no começo, seguida por camadas de convolução que continuam até a última camada.

As imagens geradas por *G* são utilizadas no treino de *D*, sendo marcadas como imagens falsas. Enquanto isso, o resultado de *D* é utilizado na composição da medida de perda da rede *G*.

O objetivo é que *G* use o *feedback* de *D* para gerar imagens melhores, que sejam capazes de gerar falsos positivos na avaliação de *D*, enquanto *D* seja capaz de discernir as novas imagens geradas por *G* do conjunto real. Este treinamento paralelo é a base da funcionalidade de *GANs*, tendo como objetivo melhorar a performance de ambas as redes simultaneamente, de forma que uma forme resultados artificiais mais convincentes (e, portanto, mais parecidos com o conjunto de treino), e a outra seja capaz de reconhecer características que revelam imagens que foram geradas artificialmente. Isso resulta em duas redes de alta precisão, cujos bons resultados de uma só foram atingidos graças à crescente melhoria da outra.

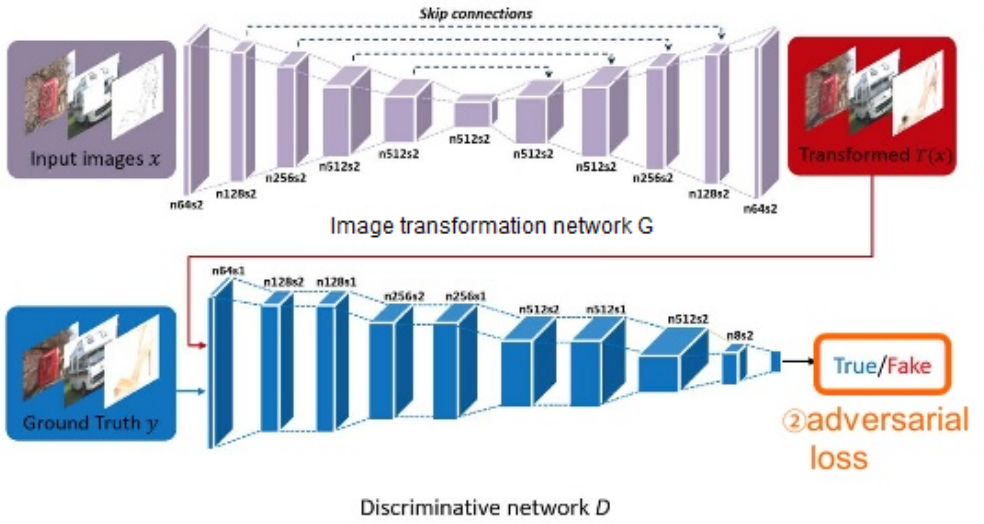


Figura 2.7: *Rede geradora generativa que utiliza uma imagem como condicionador, também conhecida como cGAN. Figura adaptada da página "Unpaired Image to Image Translation with CycleGAN"(https://blog.paperspace.com/unpaired-image-to-image-translation-with-cyclegan/)*

O uso de *GAN*s foi expandido por Isola *et al.*, 2016, que apresenta o conceito de também utilizar uma imagem como entrada em *G* e *D*. Esta imagem é utilizada como condicionadora pela rede para guiar a geração das imagens por *G*, permitindo a definição de certos atributos na imagem gerada. A rede *D* utiliza essa imagem para calcular se a imagem gerada por *G* condiz com a entrada. Isso permitiu com que o modelo *GAN* deixasse de gerar imagens convincentes a partir de dados aleatórios, e passasse a gerá-los utilizando um condicionador compreensível por seres humanos. Neste modelo, uma imagem real é usada para condicionar a rede, e esta a transforma de alguma forma baseada no treinamento que recebeu. Dado a ênfase na possibilidade de condicionar a rede para o resultado desejado, este modelo foi batizado de *cGAN* (Conditional Generative Adversarial Network). A rede

G então passa a ser composta inteiramente por camadas de convolução, e pode assumir diversas estruturas já conhecidas para tradução de imagem, como a U-Net (Ronneberger *et al.*, 2015), já que ambas a entrada e saída agora são imagens. Um exemplo disso pode ser visto na Figura 2.7, que utiliza um modelo U-Net para a sua rede geradora, recebendo e gerando uma imagem. O exemplo também utiliza diversas camadas de convolução na sua rede discriminadora, para somente na última camada possuir uma ligação completa.

Uma vez que as *GANs* agora conseguem receber imagens como condicionadores, seu uso pôde se expandir para edição de imagens, como remoção ou inclusão de objetos, preenchimento com texturas, ou até mesmo conversão de desenhos simples para imagens fotorrealistas.

Capítulo 3

Revisão bibliográfica

Neste capítulo, serão explorados alguns trabalhos da literatura que tratam de temas relevantes para esta dissertação. Alguns foram inspirações para os tópicos aqui estudados, enquanto outros apresentam soluções e análises diferentes para problemas parecidos.

Primeiramente, será falado sobre a área de interpolação de vídeos através de redes neurais. Após isso, trataremos especificamente sobre a criação de quadros para animações tradicionais. Apesar de muitas das técnicas aplicadas em estudos de interpolação de *frames* serem reaproveitáveis neste contexto, muitas deixam de ser efetivas por conta da diferença entre imagens de um vídeo real, e os de um desenho animado. Por se tratar de uma aplicação específica da geração de quadros, ele apresenta menos estudos especializados, e ainda não possui tantos usos práticos solidificados na indústria.

As pesquisas aqui mencionadas aplicam conhecimentos de *machine learning* em diversas etapas da criação de animações tradicionais, e todas buscam lidar com problemas que se estendem por qualquer pesquisa nessa área, como a alta variação entre quadros adjacentes, a impossibilidade da utilização de artefatos visuais para gerar resultados aproximados convincentes, e as grandes áreas com píxeis idênticos que dificultam na diferenciação dos elementos entre cada quadro.

Seguindo estes dois temas, cada pesquisa será abordada separadamente, e buscamos ressaltar a proposta, contribuições e os principais conceitos de cada uma.

3.1 Interpolação de vídeo

Depth-Aware Video Frame Interpolation (DAIN) (Bao *et al.*, 2019) é um modelo que utiliza uma robusta arquitetura que separa a extração de características dos quadros de entrada em diversos módulos. Cada um desses módulos é especializado numa tarefa, quais sejam: extração de contexto; estimativa do *optical flow*; estimativa da profundidade e estimativa do *kernel* de convolução. As partes são depois combinadas num novo módulo onde se sintetiza o quadro intermediário (Fig. 3.1).

O foco desta rede está na solução do problema da ocultação de objetos (*object occlusion*), ou seja, a interpolação dos quadros com objetos que estão totalmente ou parcialmente fora

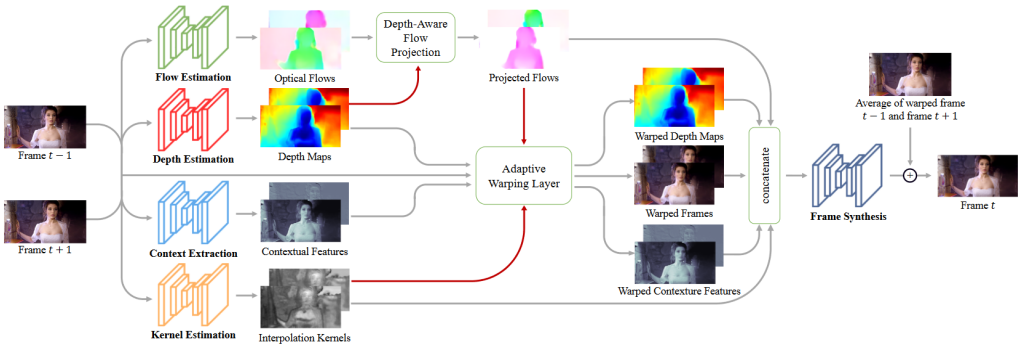


Figura 3.1: *Arquitetura da rede DAIN. Combinando múltiplos módulos de redes neurais, a rede final é capaz de fazer a interpolação de vídeos de gravações. Figura adaptada do artigo "Depth-aware video frame interpolation"por* Bao *et al., 2019.*

do campo de visão. Este problema não é trivial de ser solucionado, uma vez que métodos tradicionais, como treinar a rede para reconhecer objetos vistos em um conjunto de treino, seriam impraticáveis por conta do tamanho de variações e exemplos que poderiam existir. A rede DAIN busca solucionar o problema da ocultação através de uma arquitetura voltada especificamente para sintetizar objetos que estão parcialmente oclusos. A arquitetura dividida em módulos foi pensada justamente para tratar do problema da ocultação. O módulo de estimação do mapa de profundidade é uma das principais propostas da pesquisa, visto que, utilizando as profundidades aproximadas de diferentes elementos da cena, é possível estimar a posição intermediária de cada um, sem a interferência da ocultação por outros elementos. Este cálculo da posição intermediária é tradado pelo módulo de cálculo bi-direcional do *optical flow*, que toma proveito das informações de profundidade extraídas pelo outro módulo para estimar com fidelidade a posição intermediária de cada elemento. E não menos importante, um módulo de sintetização, que é capaz de criar imagens baseadas nos quadros originais, e utilizando as informações de profundidade e *optical flow*, colocar os objetos nas posições corretas enquanto respeita as proporções e limites dos originais.

A arquitetura demonstra resultados excelentes ao interpolar sequências de vídeos reais, e inclusive é capaz de ser usada em vídeos já finalizados para aumentar indefinidamente o número de quadros por segundo.

Porém, quando aplicada para o campo das animações tradicionais, ela acaba sacrificando a qualidade do desenho em cada quadro, por tentar aplicar efeitos como *blur* nos desenhos, algo que seria cabível em um vídeo real, porém não em uma animação tradicional.

A rede Intermediate Feature Refine Network for Efficient Frame Interpolation (IFRNet) (Kong *et al.*, 2022) propõe uma abordagem diferente da que é normalmente utilizada em redes de interpolação de quadros. Enquanto a maioria divide as tarefas de cálculo de *optical flow*, extração de características e sintetização de detalhes em múltiplos módulos, cada um com sua própria arquitetura e funções de perda, a proposta da IFRNet é realizar todas essas tarefas em uma única rede, treinada em sua totalidade com as mesmas funções de perda.

A ideia é que, ao se calcular o *optical flow* e somente usá-lo no módulo seguinte, muitas

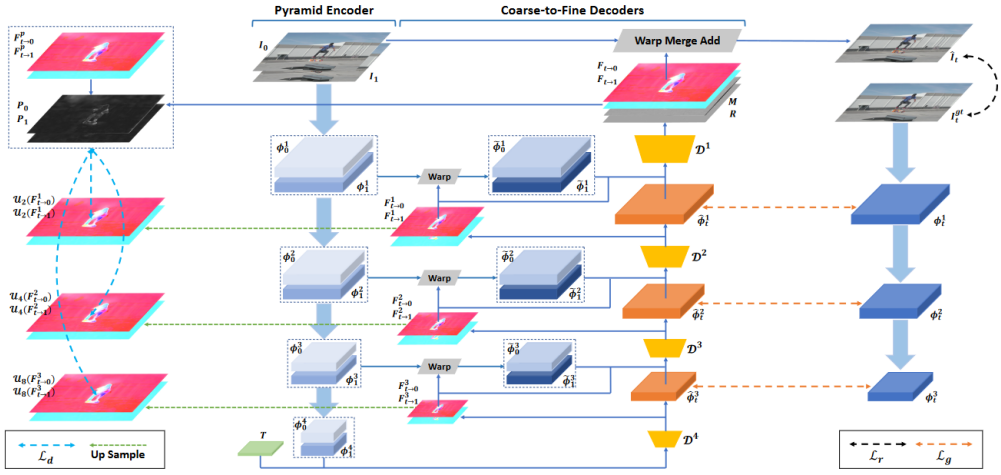


Figura 3.2: *Arquitetura da rede IFRNet. Utilizando uma estrutura única, baseada no conceito encoding-decoding, o optical flow é calculado e aproveitado em todas as etapas da sintetização. Figura adaptada do artigo "IFRNet: Intermediate Feature Refine Network for Efficient Frame Interpolation"por* KONG *et al., 2022.*

das conexões são desperdiçadas e muitas das informações são descartadas. A proposta para evitar esse desperdício é realizar a extração de características (*features*) em paralelo ao cálculo do *optical flow*. A extração das características é feita em uma hierarquia descrita como "piramidal", onde então são refinadas gradualmente até sintetizar os detalhes mais finos da imagem. O cálculo do *optical flow* é feito utilizando informações da extração de características, e simultaneamente fornecendo informações para a sintetização. A rede é treinada utilizando uma função de perda que usa informações extraídas dos quadros originais para restringir os objetos do quadro interpolado. Conforme visto na Figura 3.2, isso resulta em uma rede muito menor do que as utilizadas em outras abordagens, por conta da menor quantidade de camadas e parâmetros, e também resulta em um tempo de processamento muito menor, sem ferir a qualidade das imagens geradas.

3.2 Animação tradicional

Enquanto pesquisas sobre interpolação de vídeo têm avançado consideravelmente, estas são, em sua maior parte, voltadas para vídeos reais. Animações tradicionais diferem de vídeos reais por possuírem elementos bem-definidos com bordas, cada um colorido com apenas uma cor, e por possuírem sequências com movimentos muito mais espaçados. Esta diferença dificulta a aplicação de métodos popularmente utilizados na interpolação de vídeos reais, como o cálculo do *optical flow* (NARITA *et al.*, 2019).

A rede AnimeInterp (SIYAO *et al.*, 2021) possui uma metodologia construída especificamente para lidar com estas diferenças características de animações tradicionais. Primeiramente, sua pesquisa subdivide o problema da interpolação de quadros de desenho em dois: a falta de textura, que dificulta a correspondência entre objetos em quadros adjacentes, e a grande diferença na posição dos objetos da cena entre quadros, que podem se alterar tanto em posição quanto forma, além de seu movimento que pode ser não-linear.

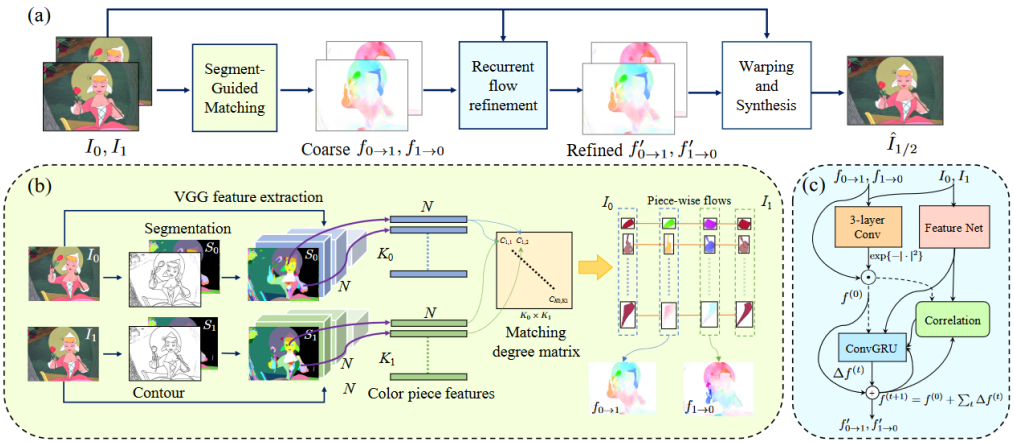


Figura 3.3: *Arquitetura da rede AnimeInterp. Múltiplas estruturas são utilizadas, com o fluxo (a) representando o modelo geral da rede, (b) representando o módulo de segmentação, e (c) o módulo de refinamento do optical flow. Figura adaptada do artigo "Deep animation video interpolation in the wild"por* Siyao et al., 2021.

Para tratar desses dois problemas, a rede AnimeInterp utiliza uma função de correspondência de cores entre quadros seguidos para identificar o movimento realizado pelos objetos da cena, além de um método de refinar o cálculo do *optical flow* para tomar em consideração o limite estabelecido pelas bordas. Ambos podem ser vistos na Figura 3.3, que mostra estes dois módulos, assim como a estrutura geral da rede.

A rede foi treinada utilizando-se um *dataset* denominado ATD-12K, que consiste de 12 mil triplas de imagens de animações de diversos estilos, além de anotações que classificam o tipo de movimento da tripla, da área de interesse onde a variação principal está ocorrendo, e a dificuldade da possível inferência do quadro intermediário. Utilizando o método da correspondência entre cores, o refinamento do *optical flow*, e o *dataset* ATD-12K, a rede AnimeInterp é capaz de gerar imagens convincentes, mesmo em animações de estilos diferentes dos usados no conjunto de treino.

Um fator importante é que a sua aplicação é voltada para animações prontas. Isso significa que os quadros já passaram pelas etapas de colorização, inclusão de plano de fundo, e inclusão de efeitos gráficos. Por conta disso, seu campo de atuação difere do trabalho de pesquisa aqui apresentado. Nosso objetivo é estudar um método para ser utilizado na etapa de definição dos contornos dos quadros. Na linha do tempo de produção de animações, esta etapa vem antes da etapa de colorização.

A pesquisa realizada pelo AnimeInterp foi levada um passo adiante com Chen e Zwicker, 2022, que propôs utilizar a arquitetura do AnimeInterp com melhorias com o objetivo de evitar efeitos borrados nas animações, algo comum em interpolações e que se torna indesejável em desenhos animados. Isso é feito através de uma nova rede, denominada *SoftSplat Lite*, que utiliza tecnologias mais recentes de cálculo de *optical flow*, enquanto o desenvolvimento feito pela pesquisa propõem alterações focadas em na extração de características e sintetização de detalhes, focando na otimização destes dois para criar uma rede mais efetiva, e também mais leve do que a original do AnimeInterp.

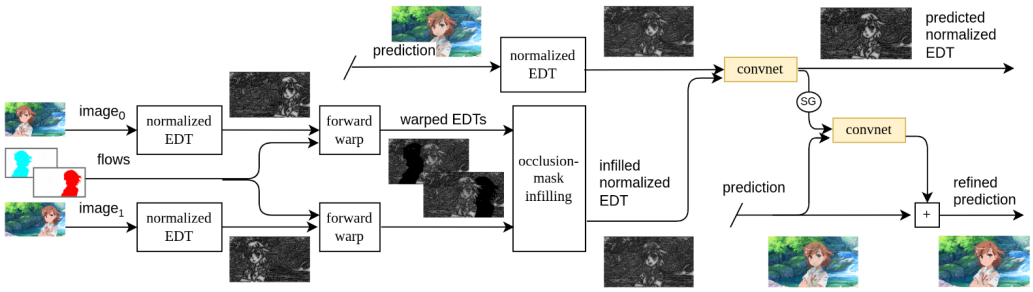


Figura 3.4: *Arquitetura da rede de transformada da distância utilizada na SoftSplat Lite, que fica separado da rede principal. Figura adaptada do artigo "Improving the Perceptual Quality of 2D Animation Interpolation"por* Chen e Zwicker, 2022.

A pesquisa também propõem a alteração da correspondência de objetos em quadros diferentes. O que antes era feito no AnimeInterp utilizando a correspondência de cores, no SoftSplat Lite passa a ser feito através de uma rede que utiliza a transformada da distância (Strutz, 2023) para aproveitar os limites criados pelas linhas, e melhor identificar as regiões que são delimitadas por elas. Esta rede pode ser vista na Figura 3.4, que fica separada da rede principal da SoftSplat Lite. Apesar da limitação do uso de quadros coloridos ainda estar presente, este método apresentou resultados melhores do que os do AnimeInterp para identificar regiões correspondentes entre quadros diferentes.

Li *et al.*, 2022 propõem uma abordagem diferente das anteriores, que apesar de requerer mais do que apenas os quadros já desenhados, apresenta resultados que podem ser melhor ajustados pelo artista que os desenhou. Ainda utilizando os quadros finalizados com cores, a sua pesquisa sugere o uso de um esboço, apenas com bordas, para fazer a geração dos quadros intermediários.

Apesar do desenho extra que se torna necessário por conta do esboço, a rede se demonstrou capaz de gerar quadros condizentes com os originais. O esboço também não precisa ser detalhado como um quadro da animação original, já que é aplicado um algoritmo de simplificação de traços no esboço, tornando-o mais funcional para extração de características, e mais fácil de ser desenhado. A pesquisa utiliza também uma rede temporal inspirada por pesquisas em animações 3D para fazer a estimação bidirecional dos movimentos, o que torna o posicionamento dos objetos mais crível.

Um ponto notável dessa pesquisa, que é similarmente tratado nesta dissertação, é a montagem do *dataset*. O artigo considera que, por conta da baixa taxa de quadros-por-segundo de animações tradicionais (por volta de 12), o treinamento da rede pode ser dificultado. Para isso, ele utiliza animações 3D, junto com as tradicionais, para realizar o treino. Animações 3D possuem o benefício de ter uma taxa de quadros muito maior do que as tradicionais, e mesmo com a diferença no domínio entre ambos, a suavidade das animações 3D permitiu com que a rede produzisse animações tradicionais similarmente suaves. A utilização de imagens de outros domínios por esta dissertação será explicada no capítulo 4.3.1.

No campo da assistência para animadores no nível de produção da animação, Even *et al.*, 2023 se destaca por tratar de uma etapa mais anterior do que a etapa *douga*, traba-

lhada nessa dissertação. Considerando a etapa de esboço, onde os traços são ainda mais simples, e frequentemente não- contíguos, esta rede permite com que artistas visualizem a movimentação e distorções desejadas em seus desenhos simplificados, focando mais na visualização e na assertividade do posicionamento e formato dos objetos, do que na precisão das linhas. Isso porque, por estar na etapa de *sketching*, a precisão destas importa menos, já que sua definição final vem somente na etapa seguinte (*douga*).

Estas movimentações e distorções dos *sketches* é feito através de uma rede de correspondência de regiões, que busca de ater menos aos traços do desenho, e mais às características geradas por conjuntos de traços. A baixa necessidade de comparar traços permite com que desenhos *sketches* possam ser feitos sem se preocupar com o número de traços que o compõem, algo que é crucial para redes que analisam *sketches* através da contagem traços.

Na questão da colorização, há pesquisas tanto na área de desenhos estáticos como na área de desenhos animados. Shi *et al.*, 2020 se aprofunda neste último, propondo uma rede que utiliza alguns exemplos de quadros coloridos para colorir outros quadros similares automaticamente. Isso é feito utilizando a correspondência entre elementos na imagem baseado na extração de bordas usando a transformada da distância destas bordas.

Neste último, também foi utilizado a rede *sketchKeras* (lllyasviel, 2017). Esta rede realiza a conversão de imagens de desenhos colorizados para uma versão apenas com seus contornos mais relevantes. Essa rede é útil, pois permitiu aos pesquisadores gerarem quadros *douga* mesmo sem possuir os desenhos originais. Esta ideia também será explorada no capítulo 4.3.2.

Indo além do campo de modelos de aprendizado de máquina, o trabalho DiLight (Carvalho *et al.*, 2017) lida com a questão da interpolação de quadros utilizando um modelo matemático para fazer a correspondência entre traços do quadro inicial e do quadro final, representando estes traços como curvas 2D. Esta representação permite com que os quadros intermediários sejam gerados através da interpolação do movimento destas curvas, possibilitando a geração de qualquer número de quadros intermediários.

Esta abordagem utilizando um modelo de curvas também necessita de linhas de apoio, ou *guidelines*. Estas linhas são desenhos adicionais que devem ser feitos pelo artista, mas muito mais simples do que os desenhos *douga*, que se assemelham ao formato do desenho final. Estas linhas são usadas para melhorar a correspondência de cada traço do desenho final, visto que as linhas de apoio possuem menos traços e são, portanto, mais úteis para fazer a correspondência, o que permite com que este modelo matemático seja aplicado para animações sem a necessidade do treinamento realizado com redes neurais.

Capítulo 4

Método

Neste capítulo descrevemos o método desenvolvido durante esta pesquisa. Serão detalhados as considerações feitas para a adaptação da rede neural Pix2Pix (Isola *et al.*, 2016) , e para a montagem do conjunto de imagens.

4.1 Definição

Tomando como contexto a criação de quadros para animações tradicionais, o escopo deste estudo foi limitado para considerar apenas sequências em que há pouca variação na posição dos objetos no quadro. Além disso, os quadros utilizados se encontram na etapa *douga* da criação da animação, e portanto são compostos apenas por bordas, sem preenchimento com cores.

Estudamos um método que utiliza uma rede geradora adversarial convolucional *CGAN* (Isola *et al.*, 2016), que recebe dois quadros de animação consecutivos constituídos apenas por bordas, e gera uma imagem que possui os mesmos objetos dos quadros de entrada, e que estes estejam em uma posição intermediária no movimento descrito pelos dois quadros.

Esta rede utiliza quadros no estágio *douga* por este se encontrar em uma das primeiras etapas de produção da animação. Isso permite com que o artista tenha a liberdade de gerar quadros, passá-los para revisão, e caso algo precise ser alterado, ele pode refazer o processo até possuir quadros que estejam de acordo com o esperado. Além disso, visto que há trechos de animações compostos por sequências de quadros iguais (ou seja, cenas estáticas), esta rede tem como finalidade ser utilizada em cenas com movimento, pois é em cenas assim que há a necessidade de mais quadros distintos.

Para se treinar uma rede que seja capaz de interpolar movimentos simples utilizando apenas desenhos com bordas, foi montado um *dataset* composto por quadros com apenas bordas, extraídos de animações finalizadas, e imagens simuladas, constituído por animações com formas geométricas simples realizando movimentos uniformes.

4.2 Estrutura da Rede

A estrutura das redes neurais foi baseada no modelo generativo apresentado por Isola *et al.*, 2016. A rede geradora G segue o modelo U-Net apresentado por Ronneberger *et al.*, 2015, que utiliza ligações residuais entre camadas de lados opostos da rede para processar simultaneamente informações de alto nível, extraídas pelas primeiras camadas, e informações de baixo nível, obtidas após várias camadas de convolução. A Fig. 4.1 apresenta a arquitetura da rede geradora G. A figura apresenta as diversas camadas colorizadas de acordo com a etapa que o fluxo se encontra e a tarefa que realiza.

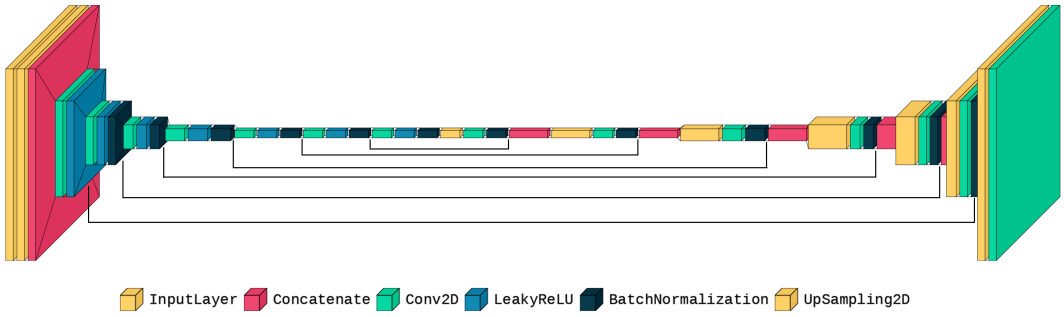


Figura 4.1: *Arquitetura da rede geradora G adaptada para este trabalho. É possível notar, além das ligações normais entre camadas internas, as skip-connections, que permitem a transferência de valores entre camadas em lados opostos. Figura gerada através do pacote visualkeras (Gavrikov, 2020).*

Alteramos a rede U-Net utilizada pela rede geradora original para receber duas imagens como entrada, ao invés de apenas uma. A rede G recebe dois quadros consecutivos da animação e gera o quadro intermediário, de forma que este complemente o movimento descrito pelos quadros de entrada. Ambas são concatenadas na primeira camada da rede, e então passam por diversas camadas de convoluções. Nestas camadas, são utilizados *kernels* de dimensão 4x4. Na primeira camada, são utilizados 64 *kernels*, e este número é dobrado após certos blocos, sendo que nas camadas no meio da rede, são utilizados 512 *kernels*. Cada camada de convolução utiliza *stride* de 2 píxeis, diminuindo pela metade a dimensão das imagens a cada vez que passam por uma dessas camadas.

Após as camadas de *downsampling* da rede, são introduzidas áreas de *upsampling*. Nestas, são utilizadas camadas do *keras* chamadas de *UpSampling2D*, que dobram o número de linhas e colunas da entrada que recebem, e preenchem as novas posições com os mesmos valores que já estavam na matriz. Após cada camada de *UpSampling2D*, é adicionada uma camada de convolução com *stride* de 1 pixel, a fim de manter o tamanho da imagem. O número de *kernels* também é diminuído pela metade em cada uma delas. Também são utilizadas *skip-connections*, ligando cada bloco de convolução da primeira metade da rede com o da segunda metade.

As camadas de *batch normalization* utilizam momento de 0.8, e a ativação da *LeakyReLU* utilizam *alpha* = 0.2. As convoluções são feitas com *stride* de 2 píxeis, e na última camada, ao invés de utilizar a ativação *LeakyReLU*, é utilizada a função de tangente hiperbólico. Esses hiperparâmetros foram baseados na implementação da rede Pix2Pix.

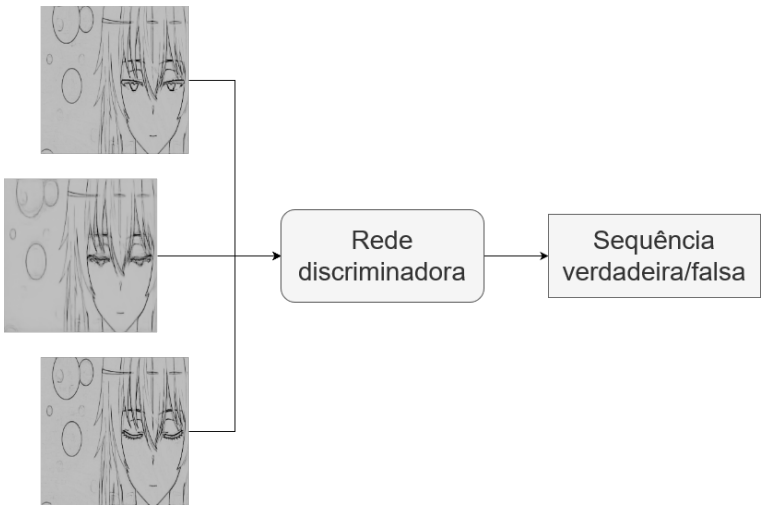


Figura 4.2: *Funcionamento simplificado da rede discriminadora. Esta recebe uma sequência de três quadros, e avalia se esta sequência é válida ou não.*

A rede discriminadora D, como visto na Figura 4.2 utiliza três imagens como entrada: o quadro inicial, o quadro intermediário, e o quadro final. A finalidade de utilizar este conjunto de três quadros como entrada é treinar D para identificar uma sequência válida de quadros. Ou seja, dado um instante i de uma animação, a rede recebe uma sequência de quadros (f_{i-1}, f_i, f_{i+1}), tal que a sequência descrita pelos quadros f_{i-1} e f_{i+1} tenha quadro intermediário o quadro f_i. Estas imagens de entrada são concatenadas na rede, e então passam por blocos de convolução, diminuindo pela metade o seu tamanho, e dobrando o número de *kernels* à medida que passam pelas camadas da rede, conforme visto na Figura 4.3. A saída desta rede é um número real no intervalo $[0, 1]$, obtido por uma camada de ativação com a função *sigmoid*, que representa o quanto as três imagens se assemelham a uma sequência real de quadros em uma animação. Para qualquer instante i de uma animação, ambas as sequências de quadros (f_{i-1}, f_i, f_{i+1}) e (f_{i+1}, f_i, f_{i-1}) são válidas. Uma vez que, para avaliar a sequência, a rede analisa a similaridade entre o quadro intermediário e os quadros inicial e final, a rede G é incentivada a aprender a gerar quadros similares aos usados como entrada.

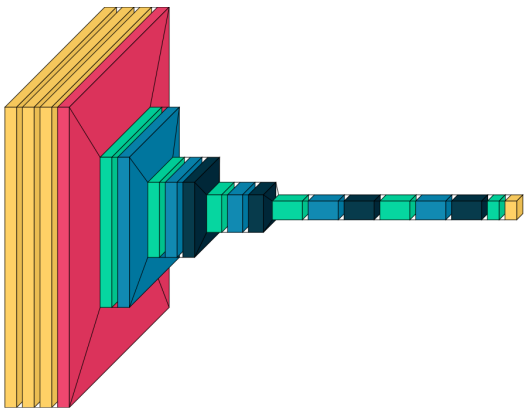


Figura 4.3: *Arquitetura da rede discriminadora D desenvolvida neste trabalho. Figura gerada através do pacote visualkeras (Gavrikov, 2020).*

4.3 Conjunto de imagens e Treino

Com o objetivo de interpolar quadros de animações que ainda estão na fase de definição dos contornos (douga), se vê necessário utilizar no treino quadros que possuam apenas contornos. Contudo, a disponibilidade destes é limitada, uma vez que são conteúdo proprietário de seus criadores. Apesar de existirem repositórios online para quadros deste tipo, como *SAKUGABOORU* (Cirugeda, 2023), nem todas as imagens possuem a mesma qualidade e aspecto de uma célula de animação original, o que poderia afetar negativamente os resultados.

Uma das contribuições deste trabalho é um conjunto curado de imagens para treinar GANs com a finalidade de interpolar quadros compostos apenas por contornos. Foram utilizadas imagens de três origens. A primeira foi o próprio *SAKUGABOORU*, fazendo uma filtragem fina dos mais adequados, e obtendo 7500 triplas de imagens. A segunda foi o *dataset ATD-12K*, disponibilizado por Siyao *et al.*, 2021, contendo 10000 triplas de imagens, tanto de animações orientais quanto ocidentais.

Como a finalidade deste trabalho é interpolar quadros que possuem apenas contornos, foi necessário tratar as imagens destes dois conjuntos utilizando a rede *sketchKeras* (lllyasviel, 2017). Esta permite que quadros de animação coloridos possam ser convertidos para seu estágio *douga* (ou seja, possuindo apenas contornos). Como é difícil encontrar grandes quantidades de quadros *douga*, esta conversão possibilita o treino da rede, pois quadros de animações finalizadas podem ser encontrados com muita facilidade. Após a conversão, as imagens possuem apenas 1 canal de cor, sendo constituídas por níveis de cinza, e são redimensionadas para 256 x 256 píxeis ao serem processadas pela rede. Essa mudança de dimensão ocorre por conta de uma limitação da rede *Pix2Pix*, que consegue processar apenas entradas deste tamanho. O uso dos conjunto do *SAKUGABOORU* é detalhado no Capítulo 4.3.2.

A terceira e última origem das imagens para o *dataset* é um conjunto de 10 mil quadros de animação simulados. Estas imagens não possuem o nível de detalhe de uma animação real. São compostas por retângulos, elipses, e triângulos. Estes realizam movimentos uniformes em direções fixas, conforme detalhado no Capítulo 4.3.1. Ao treinar a rede apenas com estas formas, foi constatado que a rede foi capaz de gerar quadros intermediários adequados, identificando corretamente a posição das formas e pintando seus contornos.

Ao combinar os quadros de animação com as imagens simuladas, a rede foi capaz de apreender características pertinentes à movimentação expressada pelos dois quadros de entrada, conforme já havia sido feito ao treiná-la apenas com as imagens simuladas, e também estender essas informações à casos mais complexos, que são as próprias animações reais.

Durante o treino foi verificado que a rede geradora poderia adicionar artefatos à imagem para minimizar o erro calculado com a rede discriminadora, como faixas cinzas horizontais e verticais. Resultados deste tipo são comuns durante o treinamento de redes GAN, e a sua minimização é um tema de estudo por si só (C. Wang *et al.*, 2018).

Para este treinamento, foi definido um conjunto de validação que poderia ser usado a cada intervalo de iterações para verificar a qualidade da imagem gerada. A presença

de artefatos na imagem pode ser observada através deste conjunto. Ao aparecerem, a continuação do treino faz apenas com que seus efeitos se agravem. Por isso, uma vez que for detectada a sua presença, os pesos das camadas da rede são retornados para como estavam em uma iteração anterior, e então o treinamento é iniciado novamente. Utilizar esta estratégia de retorno mostrou resultados positivos, considerando que os resultados finais não apresentaram artefatos, e não foi necessário o uso de processamento pesado para forçar a GAN a evitá-los.

4.3.1 Uso de imagens simuladas

Devido à escassez de quadros de animações que possuam apenas contornos, em especial imagens de qualidade, foram utilizados dois métodos para fazer a aumentação do conjunto de dados : O uso de animações simuladas, e a conversão de animações finalizadas para quadros *douga*.

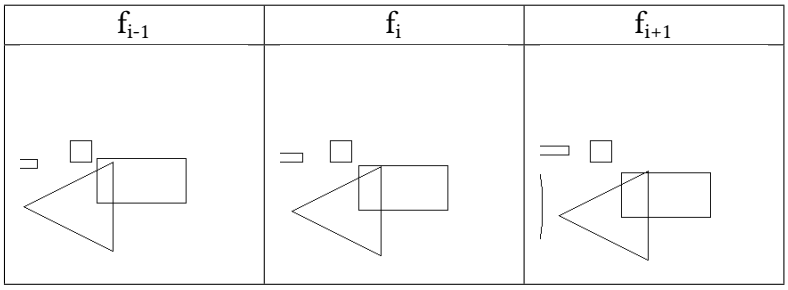


Figura 4.4: *Tripla de quadros simulados, amostrada de uma das cenas geradas para o conjunto de quadros simulados utilizados no treino da rede.*

A geração de animações simuladas consiste em criar programaticamente sequências de quadros descrevendo movimentos de formas simples, como retângulos e círculos. Estes movimentos possuem direções, sentidos e velocidades variadas, sendo que cada animação descreve o movimento de uma a três formas simultaneamente. Na Figura 4.4 estão dispostos três quadros, cada um deles com quatro formas (dois retângulos, um quadrado, e um triângulo). O movimento descrito é o de um retângulo estático, enquanto o quadrado se move para a esquerda e para cima, e o triângulo e segundo retângulo se movem para a direita e para cima. No último quadro também é possível ver uma circunferência entrando na cena, também se movendo para a direita. As formas se cruzam em diversos momentos, o que ajuda no treinamento do reconhecimento da separação de elementos na cena. As sequências utilizando animações simuladas totalizam aproximadamente 11,000 triplas de imagens, divididas em 200 sequências.

A geração dessas imagens foi feita através do programa *forms_experiments.py*, e segue o seguinte método. Para cada cena, as seguintes variáveis são definidas através de um gerador de números aleatórios com distribuição uniforme: o número de formas presentes na cena (máx. 5), o tipo de cada forma (retângulo, elipse, triângulo), a posição inicial de cada forma (representado pelo vetor b), o número de píxels que cada forma se move horizontal e verticalmente a cada quadro (representado pelo vetor a), e os vértices de cada forma. Para cada quadro i de uma determinada cena, a posição p de cada forma é dada pela equação 4.1.

$$
\begin{aligned}
a &= [a_x, a_y] \\
b &= [b_x, b_y] \\
p_i &= a * i + b
\end{aligned}
\tag{4.1}
$$

Para retângulos e elipses, tanto sua altura quanto sua largura são definidos através do gerador de números aleatórios com distribuição uniforme dentro do intervalo [30, 120] píxels. Para triângulos, a coordenada x e y de cada vértice é definida pelo mesmo gerador aleatório, cada um dentro do intervalo [15, 150]. Estes intervalos foram escolhidos pois possibilitam a geração de formas que não ocupem o espaço inteiro da imagem e não excedam o tamanho da imagem. Este intervalo também permite com que as formas sejam grandes o bastante para terem uma alta probabilidade de se intersectarem em suas trajetórias.

Como há elementos em uma animações que podem se manter parados durante a cena, enquanto outros se movem ao redor dela, foram geradas cenas que possuem uma forma estática para emular esse fenômeno. Uma em cada 3 cenas possui esta forma estática, cuja posição é a mesma em todos os quadros, enquanto as outras formas se movimentam normalmente.

O vetor a de movimento, e o vetor b da posição inicial, também são definidos através do gerador aleatório. Os componentes x e y de a são gerador dentro de um intervalo uniforme [-10, 10] píxels, e os de b dentro de um intervalo [0, 256], sendo 256 tanto a altura quanto a largura em píxels da imagem.

Definidas as formas, posições iniciais, e velocidades, o programa gera uma sequência de 20 quadros com essas formas seguindo suas respectivas trajetórias. São geradas 20 cenas deste tipo, sendo que os vetores a e b de cada cena seguinte são rotacionados com relação ao centro da imagem. Isso permite com que sejam geradas novas cenas com as mesmas formas, mas com movimentos diferentes. Foi decidido fazer 20 rotações para que cada cena visualmente distinta da anterior, visto que um número muito maior de rotações poderia gerar animações que são quase idênticas.

4.3.2 *Dataset* de *douga*

A parte do *dataset* composta por quadros *douga* foi criada baseado nas animações disponíveis no site SAKUGABOORU. Este site é um repositório para imagens e vídeos de animações, sendo possível filtrar seu conteúdo através de *tags*. Utilizando a *tag* "*animated*", é possível encontrar milhares de vídeos de animações finalizadas, contendo todos os quadros chave e intermediários. Foram escolhidas 26 animações utilizando este método, aderindo a alguns critérios.

1. A posição e forma dos elementos dentro da cena não pode variar demasiadamente de um quadro para o quadro seguinte. Na Figura 4.5 é possível visualizar um caso que não se encaixa neste critério, visto que o movimento descrito é altamente exagerado, criando novos elementos na cena sem utilizar transições.

2. A cena não deve possuir efeitos de pós-processamento em demasia, como aumento de luminosidade, adições de partículas visuais, fumaça, objetos 3D feitos em computação

gráfica, ou outros efeitos que afetem a visualização das bordas desenhadas pelo artista. Na Figura 4.6, é possível visualizar efeitos de luminosidade e partículas, e como estas se sobrepõem aos contornos.

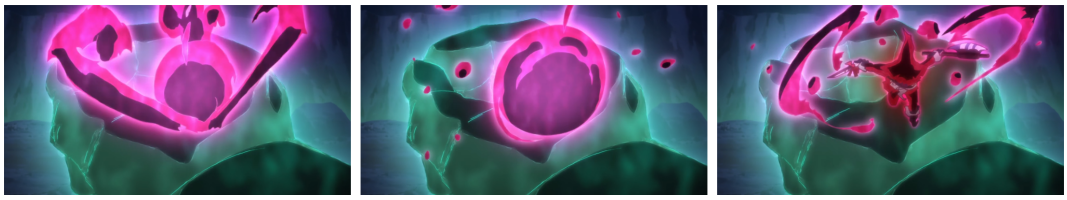


Figura 4.5: *Exemplo de animação que não se encaixa nos critérios para a montagem do conjunto. Os três quadros são consecutivos, porém o movimento descrito por eles dificilmente poderia ser interpolado, visto que muitos dos elementos presentes na cena somem e aparecem em intervalos de 1 quadro. Imagens extraídas de SAKUGABOORU.*

O primeiro critério, que limita a variação na cena de um quadro para o outro, diminui consideravelmente a quantidade de cenas que podem ser utilizadas, visto que a maioria delas, seguindo o padrão da animação limitada, utiliza a pouca continuidade entre quadros para descrever seus movimentos de forma mais exagerada.

Figura 4.6: *Exemplo de animação que não se encaixa nos critérios para a montagem do conjunto. Devido aos efeitos de pós-processamento, que neste caso aumentam a luminosidade de um local da imagem, além das partículas de faíscas adicionadas, muitas das bordas originais são afetadas. Imagem extraída de SAKUGABOORU.*

Na Figura 4.7, há dois exemplos de pares de *frames*. O primeiro possui pouca variação nos traços, sendo possível observar apenas uma leve movimentação nos fios de cabelo do personagem, e portanto foi incluído no *dataset*. O segundo já possui uma diferença maior, pois conta com um fenômeno comum em animações, que é a aparição brusca de objetos dentro da cena. O objeto no segundo quadro, neste caso um lápis, já se encontra quase na metade da altura do quadro, enquanto no primeiro ele não está visível, o que torna sua interpolação muito difícil.

Com relação ao segundo critério, animações finalizadas passam por um pós-processamento que adiciona efeitos gráficos, como iluminação e *blur*. Estes efeitos afetam negativamente a extração das bordas, visto que eles podem torná-las mais claras ou mais borradas, o que faz com que não sejam reconhecidas corretamente como contornos.

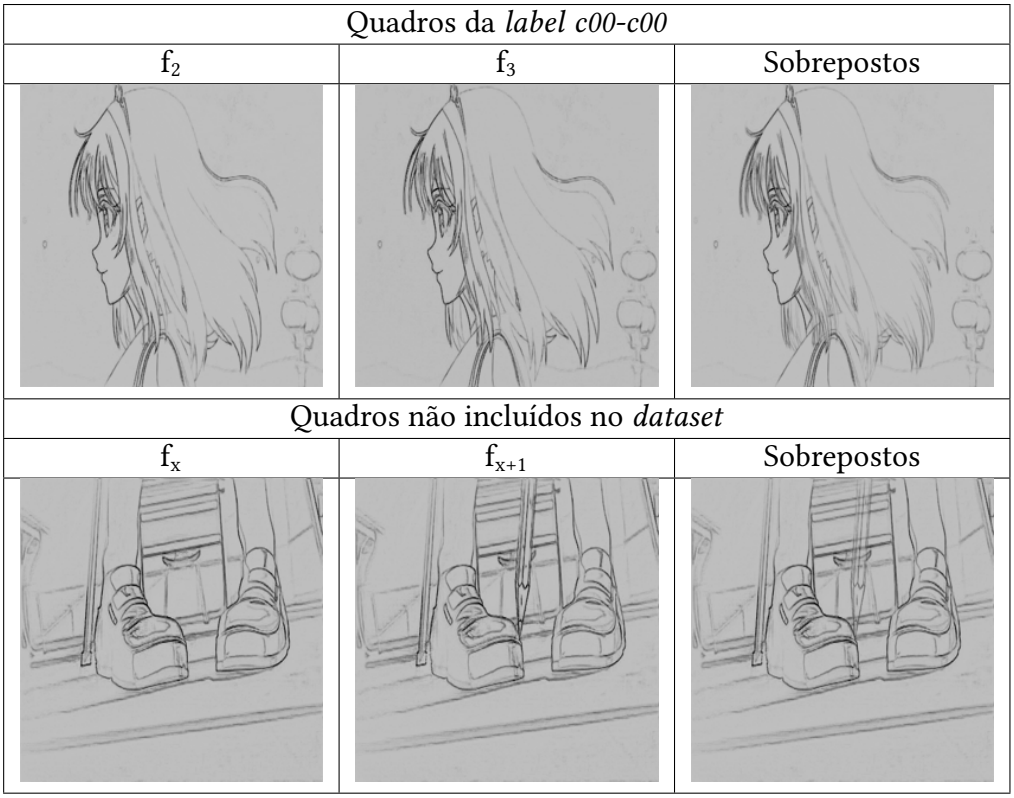


Figura 4.7: *Dois pares de quadros em sua forma douga. O primeiro foi selecionado para ser incluído no dataset, já que a diferença entre os dois quadros não é muito grande, e não há ocultação de elementos. No segundo, a variação á é muito maior, e portanto ele não foi incluído no dataset*

Além da divisão dos quadros por animação, também foi feita a divisão por cortes. Um corte é definido como uma mudança brusca entre duas sequências de quadros, não possuindo quadros intermediários entre elas. Como não há necessidade de gerar quadros entre essas sequências, eles são separados para que as imagens de um conjunto não sejam inseridas na rede com as imagens de outro, visto que não há uma imagem intermediária entre eles. No total, as 26 animações foram subdivididas manualmente em 178 sequências.

Finalizada a separação, as imagens foram processadas utilizando a rede *sketchKeras*. Cada quadro foi usado como entrada na rede, gerando assim uma versão *douga* aproximada de cada um deles. Na Figura 4.8, é possível visualizar um quadro após ser processado pela rede *sketchKeras*. Os traços da imagem gerada são notavelmente nítidos, apresentando todos os contornos presentes na imagem original. Além disso, é possível visualizar o quanto certos efeitos de pós-processamento afetam a imagem gerada, como pode ser visto no reflexo da luz no cabelo da personagem, que acaba gerando contornos para estas partes iluminadas. O impacto desses efeitos de pós-processamento foram o motivo para que estes fossem considerados nos critérios de escolha das cenas.

Figura 4.8: *Na esquerda, o quadro f_{142} da cena a08-c00. Na direita, o mesmo quadro após ser processado pela rede sketchKeras.*

Capítulo 5

Resultados

Este capítulo está dividido em duas seções principais: Resultados experimentais, e Avaliação qualitativa. No primeiro, relatamos as considerações e alterações feitas durante os experimentos, e como elas afetaram os resultados da pesquisa. Na seção a seguir, avaliamos os resultados obtidos, demonstramos as capacidades da rede, mostramos exemplos de quadros gerados por ela, e também falamos sobre os casos que não foram interpolados corretamente pela rede.

5.1 Experimento

Nesta seção, serão detalhados os experimentos realizados com a rede de geração de quadros, os seus parâmetros, e o *dataset* utilizado.

Com o modelo da rede definido na Seção 4.2, realizamos os treinamentos utilizando uma taxa de aprendizado de 0.0002. Cada um dos treinamentos foi feito utilizando um conjunto diferente de imagens, a fim de analisar o impacto do *dataset* na qualidade dos quadros gerados pela rede. Esses treinamentos levaram à definição do *dataset* finalizado, utilizado no treino da rede que gerou os melhores resultados quantitativos e qualitativos.

5.1.1 Variação de *dataset*

A base da realização dos experimentos foi a utilização de diferentes *datasets* para averiguar os resultados qualitativos e quantitativos da rede após ser treinada com cada um dos conjuntos. Para isso, podemos identificar os seguintes *datasets*:

- DS-DOUGA-1: *Dataset* construído a partir da extração de 7500 triplas de quadros de animações finalizadas do site *SAKURABOORU*, e então convertendo-as para o estágio *douga* utilizando a rede *sketchKeras*. Detalhada no Capítulo 4.3.2.
- DS-SIM: Conjunto de 10000 triplas de animações simuladas com formas e movimentos simples, geradas programaticamente. Detalhada no Capítulo 4.3.1.
- DS-DOUGA-SIM: Combinação dos conjuntos DS-DOUGA-1 e DS-SIM.

- DS-DOUGA-SIM-FLOW: Conjunto DS-DOUGA-SIM, com a adição do cálculo do *optical flow*, que será detalhado neste capítulo.
- DS-ULT: Conjunto DS-DOUGA-SIM, combinado com o *dataset ATD-12K*, totalizando 27500 triplas de imagens.

Para fazer a comparação dos quadros originais e dos quadros gerados artificialmente durante os experimentos, realizamos o treinamento iterativo da rede utilizando triplas de imagens. Para cada momento *i* dentro de uma cena, criamos uma tripla de imagens composta pelos quadros f_{i-1}, f_i e f_{i+1}. Ao utilizar os quadros f_{i-1} e f_{i+1} como entradas da rede, é gerado um quadro interpolado, que pode ser comparado com f_i tanto através de métricas de comparação de imagens, como também visualmente, observando a diferença entre os traços da imagem original e da gerada pela rede. Para medir a similaridade entre as imagens geradas e as originais, foram utilizadas as medidas PSNR e SSIM.

Apesar do número considerável de imagens do *dataset* inicial DS-DOUGA-1, a rede frequentemente não só falhava ao calcular quadros intermediários para casos fora do conjunto de treino, e também possuía precisão baixa para as imagens usadas durante o treino. Avaliando o caso, foi possível identificar que a falta de precisão com o conjunto de teste se deu por conta da falta de variedade dos casos no conjunto de treino, causando um cenário de *overfitting*. E quanto à baixa precisão dentro do treino, uma possível causa analisada foi a quantidade insuficiente de imagens de treino.

Para casos cuja variação de posição dos objetos entre os quadros fosse maior, a rede apresentava ruídos em demasia, que podem ser descritos como tons de cinza em torno nas bordas existentes em f_{i-1} que não existiam em f_{i+1}, e das bordas existentes em f_{i+1} que não existiam em f_{i-1}.

Utilizando o *dataset* DS-ULT, com 27.500 quadros, a precisão durante o treino aumentou expressivamente, conforme será detalhado na Seção 5.2. Além disso, montando o conjunto de treino amostrado deste *dataset*, uma quantidade maior de cenas foram utilizadas para o treinamento. Com isso, também aumentou o número de estilos de animação, ajudando a solucionar o problema de *overfit*.

Na Tabela 5.1 estão dispostas as medidas PSNR e SSIM médias calculadas sobre o conjunto de validação, com a rede treinada utilizando os *datasets* listados na esquerda. Inicialmente, foi feito o treino apenas com o *dataset* DS-DOUGA-1, que contém as 7500 triplas originais geradas através da conversão de animações finalizadas em animações de estágio *douga*. Após isso, foi feito um treino apenas com o *dataset* DS-SIM, que contém apenas as formas simuladas. Nota-se uma queda nas medidas ao utilizar apenas este *dataset*, o que é esperado visto que a rede não foi treinada com nenhum caso real de animação neste ponto. Ao se combinar os dois *datasets* mencionados, listado como DS-DOUGA-SIM, percebe-se uma melhora considerável no PSNR, visto que a rede teve tanto exemplos reais como exemplos de movimento simulados, e ambos se complementam.

Foi listado também o *dataset* DS-DOUGA-SIM-FLOW, que representa um estágio da pesquisa em que foi considerado utilizar o *optical flow* calculado para cada par de quadros consecutivos no *dataset* para tentar aumentar a precisam. Porém, como já foi mencionado, o *optical flow* é muito difícil de ser calculado corretamente para quadros de animação, em especial para quadros de animação apenas com bordas, então o seu uso como condicionador

piorou ambas as estatísticas, o que fez com que o *optical flow* não fosse mais utilizado.

Finalmente, temos o *dataset* final, DS-ULT. Ele representa a combinação dos dois *datasets* já utilizados, os *dougas* originais e as formas simuladas, junto com o *dataset* ATD12K, que também foi convertido para ter apenas bordas através da rede *sketchKeras*. Agora, com as 27.500 triplas de quadros, tanto a PSNR quanto a SSIM atingiram sua maior medida registrada nessa pesquisa, de 24.904 e 0.828 respectivamente.

Dataset	PSNR	SSIM
DS-DOUGA-1	23.734	0.808
DS-SIM	20.078	0.636
DS-DOUGA-SIM	24.144	0.798
DS-DOUGA-SIM-FLOW	23.973	0.787
DS-ULT	**24.904**	**0.828**

Tabela 5.1: *Foram calculadas as médias das medidas PSNR e SSIM sobre o conjunto de validação. Em cada linha, o dataset respectivo da rede que alcançou cada medida. É possível notar como a escolha de datasets mais diversos impactou positivamente em ambas as métricas.*

É possível visualizar a diferença na qualidade dos traços das imagens geradas na Figura 5.1, tendo como referência os quadros anterior e seguinte da sequência. Nesta sequência de quadros, o movimento observado está nas bolhas da esquerda, que se movem para baixo em velocidades diferentes uma da outra, e nos olhos da personagem, que se fecham.

Com o primeiro *dataset*, há a presença de alguns traços, porém sem definição, e é possível perceber muito ruído em volta deles. Já com o segundo *dataset*, os traços possuem mais definição, porém muitos elementos deixam de ser pintados, como pode ser observado nas bolhas do canto superior esquerdo, e nos cílios da personagem, que não possuem contorno definido. Finalmente, com o último *dataset*, os traços estão muito mais definidos, especialmente nos locais que mais precisavam ser interpolados. As bolhas estão desenhadas corretamente, tanto em definição quanto posição, e os olhos da personagem apresentam-se numa posição semi-aberta, o que é indicativo da posição do *frame*, que se encontra entre um quadro em que a personagem está com os olhos fechados, e outro em que a personagem está com os olhos abertos. Também é possível notar bem menos ruído na imagem, pois foram desenhados apenas os traços que já estavam presentes nos quadros originais, ou os que cuja posição e forma precisavam ser interpolados (as bolhas e os olhos). Essa melhora nos traços se reflete no aumento da PSNR, o que indica uma correlação entre essa medida e a qualidade dos traços do quadro.

5.1.2 Aumentação do *dataset* de treino

Para aumentar o número de imagens disponíveis para o treinamento da rede, as triplas de imagens do conjunto de treino poderiam ser, aleatoriamente durante o treinamento, rotacionadas em 90°, 180°, ou 270°, e também invertidas, resultando em um aumento de 8 vezes no número de triplas disponíveis. As três imagens sempre recebiam a mesma alteração, mantendo assim a consistência da animação, uma vez que rotacionar apenas uma das imagens poderia quebrar a sequência de movimento.

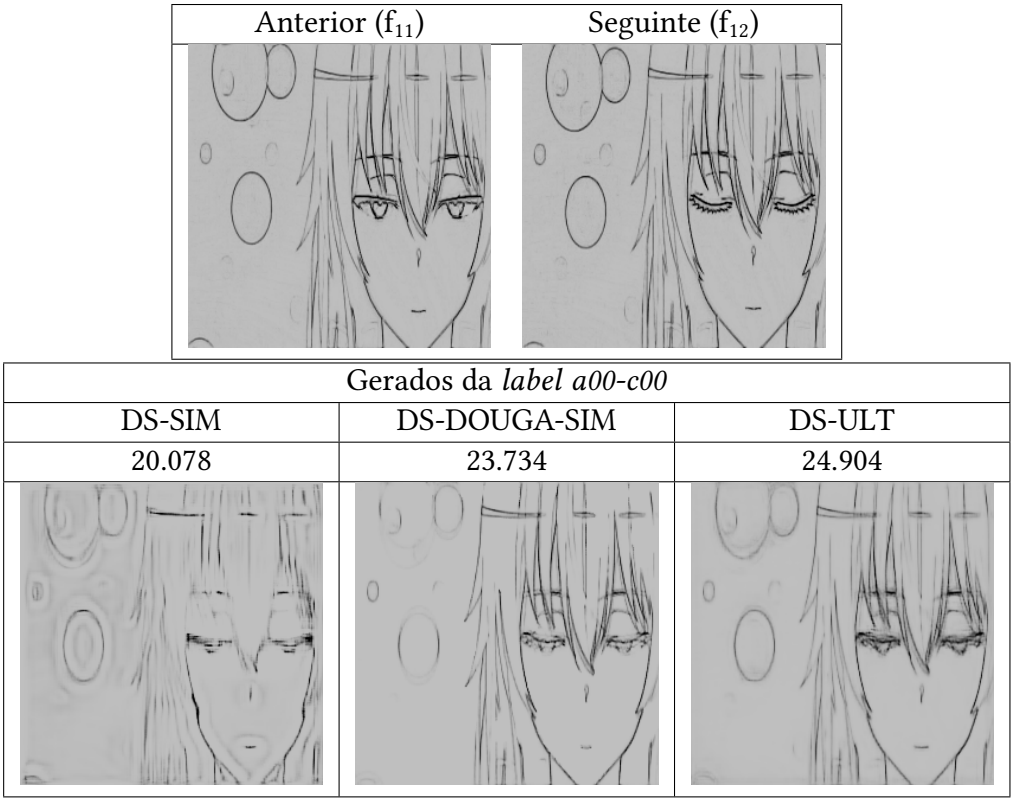


Figura 5.1: *Comparação entre as imagens geradas por algumas das redes avaliadas. Na parte superior, os dois quadros utilizados como entrada. Na parte inferior, o PSNR e o quadro inferido por cada uma das redes. É possível notar que na primeira imagem, o excesso de ruído em volta dos traços faz diminuir o valor da PSNR, enquanto que na segunda imagem, apesar de possuir uma PSNR maior, possui vários traços incompletos, ou regiões inteiras sem ser desenhadas. No terceiro, tanto o problema do excesso de ruído e dos traços incompletos foram melhorados, e por isso apresentou a melhor métrica do conjunto.*

Outros métodos comuns de aumento de *dataset* como *scaling* (Krizhevsky *et al.*, 2012) não foram utilizados, uma vez que as bordas não podem ser distorcidas. Como imagens reais possuem bordas com um aspecto padrão, ou seja, todos utilizam poucos píxeis de grossura, distorcer as imagens poderia fazer com que essas bordas deixassem de ter esse aspecto, afetando assim o treino e também os resultados.

5.2 Avaliação qualitativa

Nesta seção, serão analisados e mensurados as imagens geradas pela rede treinada nos experimentos da seção anterior. Serão exibidos tanto exemplos de quadros gerados condizentes com a entrada da rede, como também quadros em que a rede não produziu resultados de acordo com o esperado.

Ao se utilizar um par de quadros consecutivos como entrada da rede, espera-se que o quadro gerado pela rede tenha:

1. Os mesmos elementos que estão presentes em ambos os quadros de entrada.
2. Os elementos ocupando posições intermediárias do movimento descrito pelos qua-

dros de entrada.

3. Traços com mesma espessura dos utilizados nos quadros de entrada.

Observando as imagens geradas pela rede nos conjuntos de validação e de teste, foi possível verificar que ela é capaz de gerar quadros intermediários condizentes com estes quesitos. A Figura 5.2 mostra um exemplo de dois quadros originais e o quadro intermediário gerado pela rede final. Os quadros f_0 e f_1 demonstram a movimentação dos fios de cabelo do personagem, sendo que é possível gerar uma posição intermediária para esses fios, o qual é feito com sucesso pela rede e é exibido no quadro "Gerado".

Optou-se por incluir também uma imagem "Sobrepostos", que coloca um quadro sobre o outro utilizando-se transparência, para que ambos sejam visíveis na mesma imagem e fique mais fácil de visualizar a diferença entre os dois. Neste caso, é possível visualizar a diferença da posição dos fios de cabelo, e como a parte da imagem sobreposta em que os fios não se tocam foi interpolada pela imagem "Gerado".

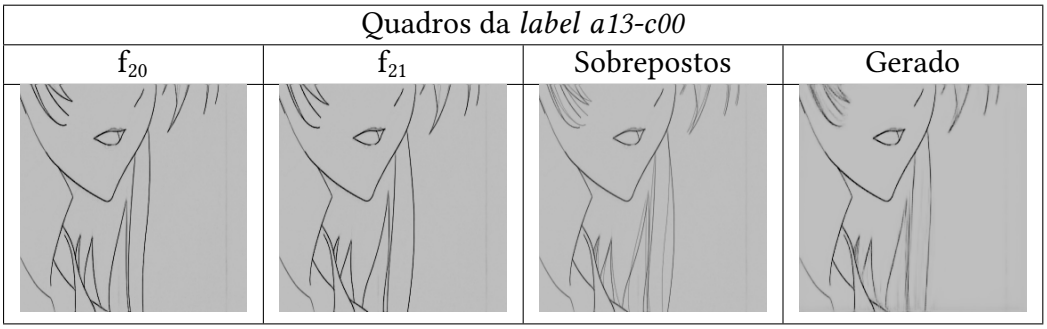


Figura 5.2: *Exemplo de um quadro interpolado (com o rótulo "Gerado") pela rede com sucesso. Os dois quadros originais (f_{20} e f_{21}), e os dois sobrepostos (rótulo "Sobrepostos").*

É possível notar que, apesar de os traços da imagem gerada estarem nas posições corretas, sua definição (grossura em píxeis) é menor do que nas imagens de entrada. Em geral, seus traços são mais claros nas regiões interpoladas, enquanto que nas regiões onde os traços permaneceram na mesma posição entre as duas imagens de entrada, sua grossura é igual.

Para casos em que a distância é muito grande, a rede deixa de desenhar as linhas correspondentes ao objeto que se moveu, por não ser capaz de definir a posição correta. Isso é consequência também do treinamento feito com a rede, uma vez que traços adicionados em posições incorretas, como ruídos, impactavam negativamente nas métricas de treinamento. Por conta disso, a rede deixa de tentar desenhar traços onde há pouca confiança da precisão de sua posição, e por outro lado, a rede deixa de adicionar traços não-condizentes com os elementos dos quadros de entrada. Na Figura 5.3 é possível ver um desses casos, em que a rede coloca traços borrados ao invés de traços bem definidos por não saber posicionar corretamente os elementos presentes na cena. Neste caso, o movimento da mão se aproximando do centro da imagem não foi interpolado corretamente, possivelmente por conta de a rede não ter sido capaz de identificar que os dedos do quadro f_{154} correspondiam à mão (agora mais visível) e os dedos no quadro f_{156}. Enquanto isso, os elementos da cena que permaneceram fixos, foram pintados corretamente pois a rede pôde fazer a correspondência de traços que não tiveram variação entre os quadros.

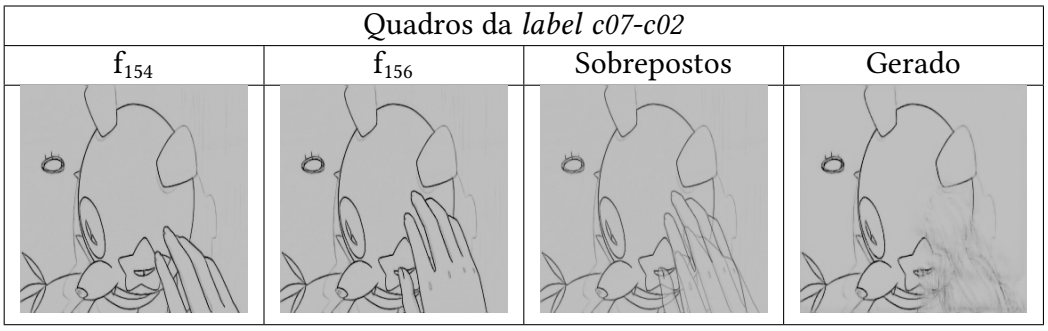


Figura 5.3: *Exemplo de conjunto de quadros interpolado pela rede em que o resultado não apresenta o desenho correto. Por conta da distância, e da ocultação para fora do quadro, a mão não foi desenhada conforme se esperaria.*

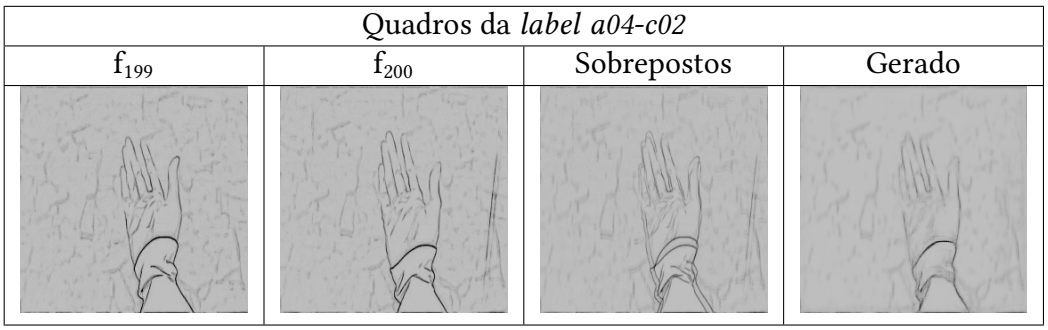


Figura 5.4: *Exemplo de conjunto de quadros interpolado pela rede. Neste caso, a cena envolve um movimento de uma mão também, porém agora interpolando corretamente. Isso é mais notável na região dos pulsos.*

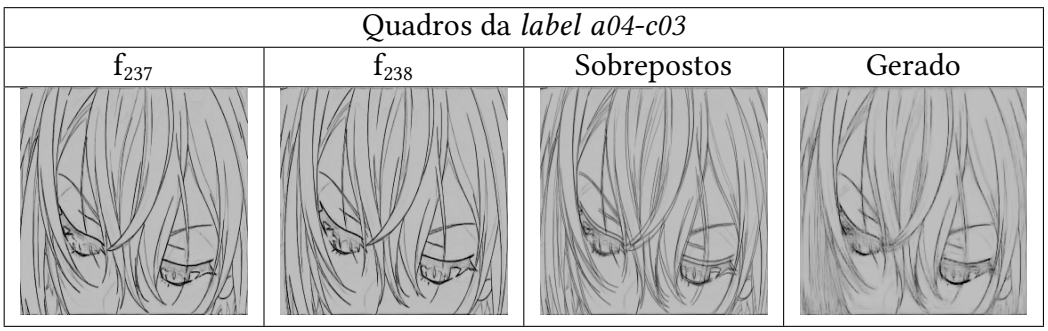


Figura 5.5: *Exemplo de conjunto de quadros interpolado pela rede. Os cabelos são interpolados corretamente, apesar de as partes nas extremidades da imagem ficarem mais borradas.*

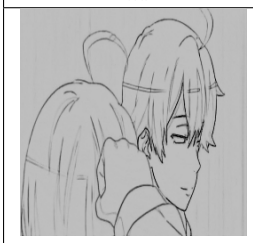
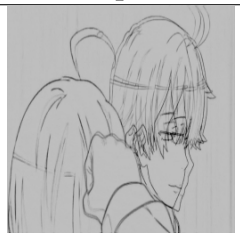
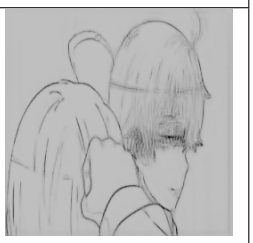
Quadros da *label a08-c00*			
f_{24}	f_{25}	Sobrepostos	Gerado
			

Figura 5.6: *Exemplo de conjunto de quadros interpolado pela rede. Por conta da mudança de ângulo da cabeça do personagem ao rotacionar, muitos traços são alterados de forma que a rede não conseguiu fazer a correspondência entre eles nos quadros consecutivos, e portanto não conseguiu interpolar corretamente.*

Para rotações como na Figura 5.6, a rede também pode apresentar dificuldades ao interpolar por conta da mudança de perspectiva, tornando alguns elementos visíveis, enquanto outros ficam ocultos. Nesses casos, é possível notar regiões mais cinza, que apesar de apresentar traços de ambos os quadros, não representa um quadro intermediário adequado.

Nas Figuras 5.4 e 5.5, já se notam interpolações melhores. Na Figura 5.4, apesar de a mão estar realizando uma rotação leve, a diferença não excedeu o limite que a rede consegue interpretar, criando assim um quadro intermediário melhor, incluindo outros movimentos, como o do pulso, que se afasta do centro da imagem. Na Figura 5.5, é possível notar algumas regiões cinzentas nos cantos da imagem, onde mechas de cabelo são introduzidas à cena em um curto intervalo, porém as mechas que já se encontravam na cena, como as do meio, são interpoladas corretamente.

A rede também foi capaz de se adaptar bem a casos em que há múltiplos elementos em movimento na cena, sendo capaz de calcular a posição correta de cada elemento individualmente. Um exemplo é a Figura 5.1, que possui tanto as bolhas descendo no lado esquerdo, como o rosto da personagem se erguendo no lado direito, além dos olhos se abrindo. Este aspecto é de grande importância, pois cenas de animação costumam ser compostas por diferentes elementos, cada um com seu movimento. Em contrapartida, quando as bordas de dois ou mais elementos entram em contato uma com a outra, a rede apresentou dificuldades em pintar a região onde ocorre esse contato, apresentando ruídos ou áreas vazias em casos do tipo.

Um dos casos em que a rede apresentou dificuldades para interpolar foi na presença de alguma borda cuja posição no quadro final ocupe a mesma posição que outra borda ocupava no quadro inicial, como pode ser visto nos cabelos do personagem na Figura 5.6. Casos assim fazem com que a rede tente introduzir características de ambas as bordas, adicionando ruído ao quadro interpolado e possivelmente deixando de pintar a posição correta das bordas envolvidas.

5.3 Programa de geração de frames

Para ilustrar e oferecer a possibilidade de outros usuários usarem a rede desenvolvida neste projeto, foi desenvolvido um programa que apresenta uma interface gráfica para o usuário ser capaz de gerar os próprios quadros intermediários.

5.3.1 Objetivos

Para os fins desta pesquisa, este programa possui os seguintes objetivos:

- Possibilitar ao usuário a fácil visualização dos quadros que ele já possui, assim como dos quadros que ele deseja gerar.
- Ser executável em computadores com poucos recursos, não necessitando de GPUs avançadas.
- Oferecer um ambiente em que seja possível gerar quadros novos e compará-los com os já existentes.

5.3.2 Tecnologias

Este programa foi desenvolvido em Python 3.6, utilizando as bibliotecas *Tkinter* 3.9.17 para a parte gráfica, e *Keras* 2.11.0 para executar a rede neural. Os pesos da rede ficam armazenados em um arquivo de extensão `.h5`, possibilitando a modularização do programa para que a rede treinada de outros métodos possa ser facilmente executada através deste programa.

As imagens de entrada para o programa podem ser salvas em uma pasta, a qual deve ser selecionada ao executar o programa. Os arquivos devem estar em formato PNG, e seus nomes serão usados para definir a ordem em que as imagens estarão dispostas na cena.

5.3.3 Fluxo

Para utilizar este programa, o usuário precisa de uma pasta contendo o conjunto de *frames* que será processado, separados em arquivos `.png`, cada um com um quadro. O programa os ordena baseado no nome dos seus arquivos. Dentro do programa, o usuário possui um menu para selecionar os arquivos com os quadros que serão processados, e ao escolhê-los, estes são carregados para a memória do programa, e são exibidos na linha do tempo na tela principal.

Na Figura 5.7, é possível visualizar a primeira tela do programa, que permite ao usuário escolher qual modo ele deseja utilizar. O programa possui dois modos de operação: o modo Geração, que utiliza os quadros definidos para gerar mais quadros intermediários; e o modo Comparação, que utiliza as triplas dos quadros definidos para gerar quadros interpolados com a finalidade de comparar o original com o quadro gerado.

No modo de geração, a tela principal do programa esta disposta de forma que seja fácil para o usuário escolher os pares de quadros com os quais ele quer gerar quadros

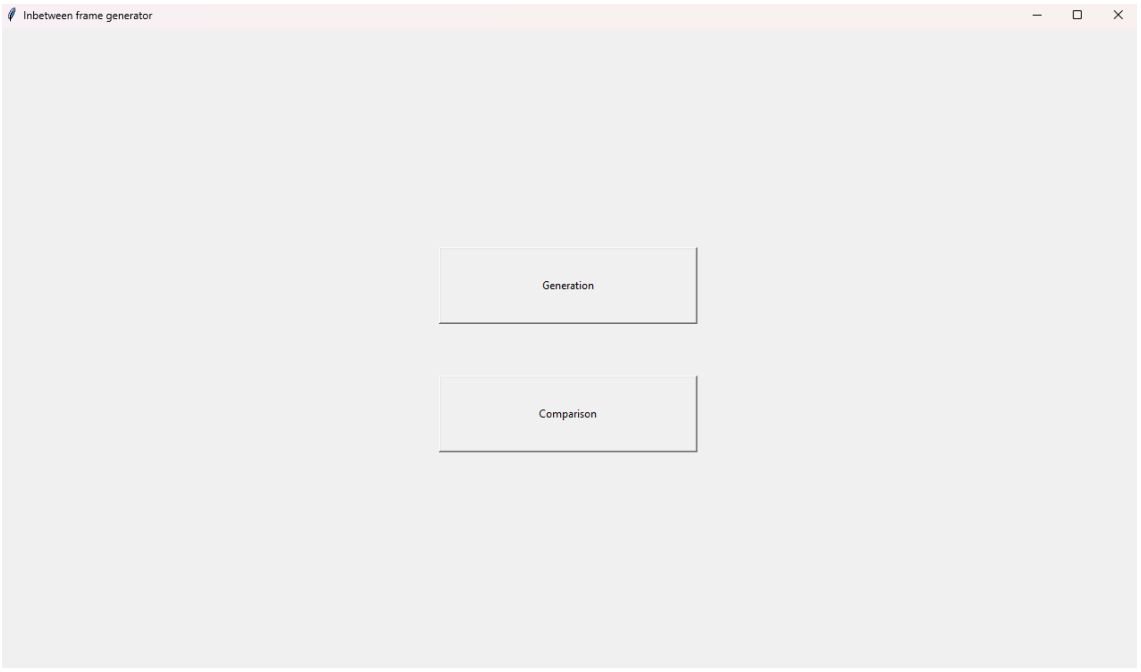


Figura 5.7: *A tela inicial do programa desenvolvido. Nela, é possível escolher entrar no modo Geração, ou no modo Comparação.*

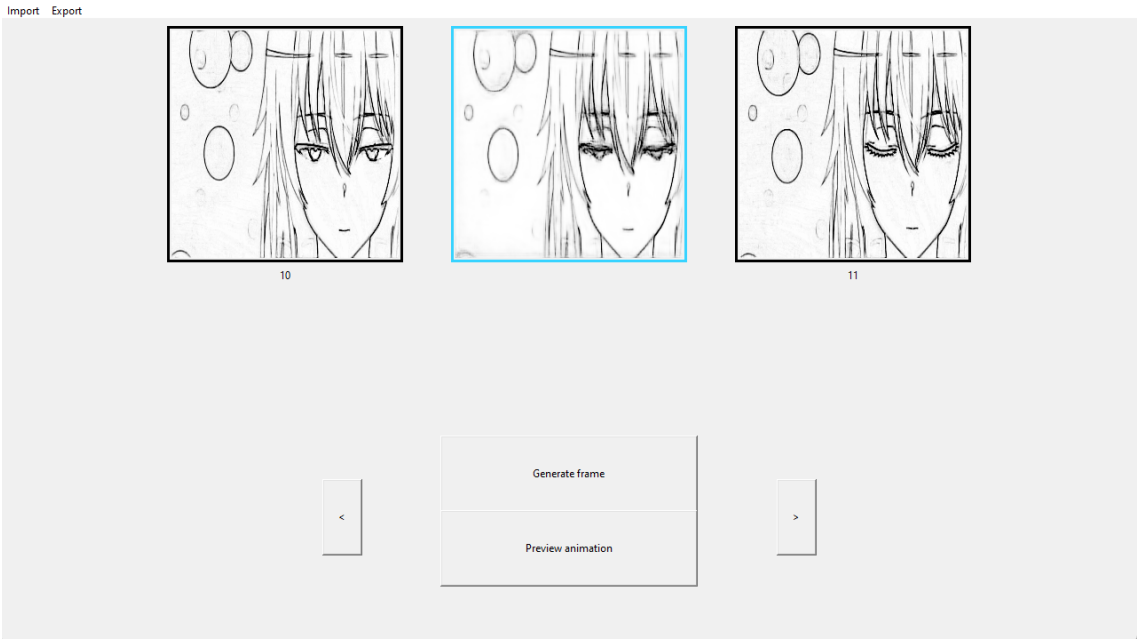


Figura 5.8: *O programa desenvolvido, no modo geração. O quadro do meio foi gerado por comando do usuário, utilizando o momento na linha do tempo em que o usuário definiu.*

intermediários. Ao escolher um par de quadros, basta clicar no botão "Gerar quadro"para criar um quadro intermediário e adicioná-lo à sequência de *frames* na memória, como visto na Figura 5.8. Nesta, observa-se novamente os quadros da *label a00-c00*, e no centro deles, o quadro intermediário gerado pelo programa. Através desta interface, também é possível comparar os quadros originais e os quadros gerados pelo programa, sendo dispostos tanto lado-a-lado, quanto alternadamente.

Para diferenciar os tipos de quadros, o programa altera a borda em volta da imagem, utilizando uma borda azul para os quadros que ele gerou, e uma borda preta para os originais.

No botão "Prever animação", é possível exibir a animação quadro-a-quadro, incluindo os quadros gerados pelo programa. Isso permite ao usuário visualizar a animação final, e fazer ajustes à linha do tempo caso deseje. Pode se observar isso na Figura 5.9, novamente com a *label a00-c00*.

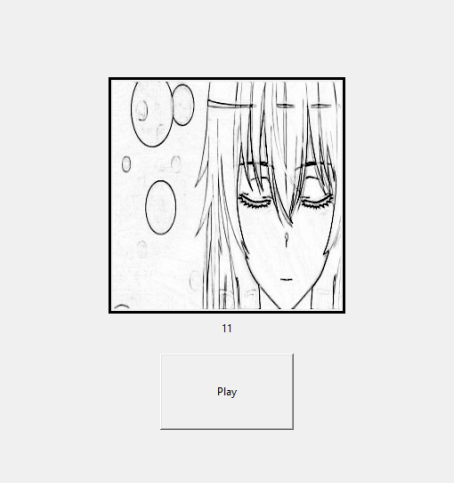


Figura 5.9: *A tela aberta pelo botão de prever animação. O botão "play"permite com que a animação seja exibida quantas vezes for desejada.*

A sequência de quadros, inclusive os gerados, pode então ser exportada para arquivos de imagem através do botão "Exportar". Ao fazer isso, os arquivos dos quadros originais são salvos com os nomes que tinham ao serem importados.

Os quadros gerados pelo programa possuem uma nomenclatura distinta para não afetar os nomes dos originais. Dado que um número n de quadros foram gerados entre os quadros originais `x.png` e `y.png`, os arquivos dos *frames* interpolados terão a nomenclatura: `x-0.png`, `x-1.png`, ..., `x-(n-2).png`, `x-(n-1).png`. Isso deixa claro que são quadros interpolados, e a posição deles na linha do tempo dos quadros originais.

Já no modo comparação, são dispostas triplas de imagens na linha do tempo. Para cada uma delas, são utilizados o primeiro e terceiro quadro para gerar um *frame* intermediário, que pode ser comparado com o segundo quadro da tripla. Como este é o *ground truth*, é possível analisar visualmente as diferenças entre ambos. Isso demonstra resultados interessantes, como visto na Figura 5.10, pois há casos em que mesmo no conjunto original de quadros, algumas transições de objetos não são feitas, como é de se esperar do estilo

Figura 5.10: *O programa desenvolvido, no modo comparação. É possível notar o frame gerado, contornado em azul, interpolando as diferenças entre os quadros 7 e 9, mesmo com o 8 do conjunto original não fazendo essa transição.*

anime e sua movimentação espaçada. Até mesmo nessas situações, o programa busca gerar o movimento intermediário, e é possível fazer a comparação do original com o gerado através desse menu.

Capítulo 6

Conclusão

Neste capítulo, abordamos os principais resultados alcançados com esta pesquisa, suas possíveis aplicações em cenários reais, e pontos que podem ser mais estudados em trabalhos futuros.

A proposta desta pesquisa era estudar a viabilidade de se utilizar uma rede neural que recebe dois quadros desenhados por um artista, e gerar um quadro intermediário que fosse condizente com os dois quadros originais, tanto em estilo de desenho, quanto na posição dos objetos na cena.

Utilizando uma rede generativa treinada com um *dataset* feito para esta pesquisa, foi possível fazer a geração de quadros intermediários através da entrada de quadros reais. Neste aspecto, podemos verificar a eficácia do uso de aprendizado de máquina na análise e sintetização de quadros intermediários.

Foi possível constatar que, apesar da arquitetura relativamente simples da rede, esta foi capaz de identificar a movimentação de elementos nas cenas. Os contornos que representam esses elementos foram desenhados corretamente em muitos casos, preenchendo o espaço temporal entre os dois quadros usados como entrada.

Os quadros gerados que apresentaram as melhores medidas de similaridade com os quadros originais foram os de cenas em que a movimentação dos seus elementos teve pouca variação espacial. Ou seja, pouca diferença entre a posição dos contornos dentro dos quadros inicial e final. Isso permite com que a rede possa ser utilizada por artistas, pois há cenas em que são necessários muitos quadros com pouca variação entre si, com o intuito de deixar a animação mais polida.

Para casos em que a variação foi muito alta, a rede apresentou dificuldades em definir a posição correta dos contornos. Essa dificuldade se torna aparente nos quadros gerados, onde podem ser observadas regiões em branco, ou acinzentadas, nos locais onde os contornos deveriam estar. Isso é de pouca utilidade para o artista, visto que a integridade dos traços é crucial para a confecção da animação.

A criação do *dataset* foi uma contribuição deste trabalho. Devido à escassez de quadros de animação em estágio *douga* no domínio público, ter uma forma consistente de gerá-los possibilita o treino de outras redes que interajam com a etapa *douga* da animação. Em

especial, a conversão para *douga* possibilitada pelo uso da rede *sketchKeras*, que em outros trabalhos é utilizada apenas com imagens estáticas, foi de grande utilidade, pois permitiu com que o *dataset* fosse montado através de animações finalizadas, que são abundantes em repositórios na Internet.

Além do uso da conversão para *douga*, também foi importante o estudo das animações simuladas. Apesar de serem apenas formas simples se movimentando, sua variedade e quantidade contribuíram para o aprendizado da rede, como foi visto nos resultados visuais e numéricos das imagens geradas pela rede. Isso significa que essa técnica pode ser potencialmente refinada para incluir mais tipos de movimento, ou outros casos que complementem o aprendizado da rede.

Por fim, o programa desenvolvido dispõem de forma simples as funcionalidades necessárias para a geração de quadros intermediários através de desenhos já prontos. Ele cumpre seu papel de dar ao artista uma ferramenta para fazer essa geração, porém seria de maior utilidade se esta funcionalidade fosse implementada em outros programas utilizados por animadores, de forma que a geração de quadros fosse mais uma opção dentro do acervo de ferramentas das quais o artista dispõem.

Considerando a rede neural treinada, o programa que faz uso dela, e a metodologia por trás da criação do *dataset*, este trabalho possui contribuições tanto para artistas, que desejam fazer uso da geração de quadros intermediários, quanto para futuros estudos, que tenham como foco a produção de animações tradicionais.

6.1 Trabalhos futuros

Apesar dos bons resultados apresentados neste trabalho, um uso real na indústria, ou até mesmo um uso real com animadores independentes, ainda requer diversas melhorias.

A principal se trata da resolução da imagem. Atualmente, por conta da arquitetura rede *Pix2Pix* utilizada, a rede suporta um tamanho de imagem fixo de 256x256 píxeis, o que é adequado para alguns casos, mas não para todos. Animações de diferentes fins possuem dimensões diferentes, e em geral, muito maiores do que esta. Uma possível alternativa seria o desenvolvimento de uma rede capaz de sintetizar o quadro final em partes, ou seja, dividindo o quadro em áreas de 256x256 píxeis, que então são desenhados separadamente para depois serem reunidos e formar o quadro final de área maior, porém seria necessário treinar a rede para lidar com as bordas sendo divididas em múltiplas partes.

Um outro aspecto importante é a variação entre quadros consecutivos. Para casos de variação maior, a arquitetura utilizada nesta rede parece não ser capaz de aprender as informações necessárias para conseguir gerar quadros intermediários adequados, com os resultados atuais estando próximos do limite da arquitetura. Para solucionar isso, uma possível solução seria a criação de uma rede com mais módulos, cada um capaz de utilizar outras características extraíveis dos quadros inseridos, e também módulos capazes de utilizarem estas informações para sintetizar quadros de melhor qualidade. Outro ponto é a não-linearidade de movimentos. Como a rede recebe apenas 2 quadros como entradas, a posição intermediária é considerada a média linear entre eles. Para movimentos que possuem aceleração, seria necessário receber ao menos 3 quadros de entrada.

Seria de grande relevância algo que seja capaz de substituir o *optical flow*, usado em redes de interpolação de vídeo. Possivelmente, até um *optical flow* treinado especificamente para quadros *douga*, que dependa menos de informações de baixo nível dos píxels individuais, e que faça uso das regiões definidas pelas bordas do desenho.

6.2 Considerações finais

O problema do intensivo trabalho manual na criação de quadros de animação continua a ser uma questão abordada não só na literatura, como também dentro de estúdios de animação. Ele permanece não-resolvido, sendo necessário grande investimento de tempo e dinheiro para se chegar a uma animação com uma quantidade boa de quadros. Esperamos que as pequenas ideias e contribuições apresentadas neste trabalho sejam úteis para pesquisas futuras, e que o a qualidade do ofício de desenhistas de animação possa se elevar como consequência.

Referências

[Amari 1967] Shunichi Amari. "A theory of adaptive pattern classifiers". Em: *IEEE Transactions on Electronic Computers* EC-16.3 (1967), pgs. 299–307. doi: 10.1109/PGEC.1967.264666 (citado na pg. 14).

[Anonymous *et al.* 2021] Anonymous, Danbooru community e Gwern Branwen. *Danbooru2020: A Large-Scale Crowdsourced and Tagged Anime Illustration Dataset*. https://gwern.net/Danbooru2020. dataset. Accessed: 2023-06-11. Jan. de 2021. url: https://gwern.net/Danbooru2020 (citado na pg. 3).

[Balasubramaniam e Pasricha 2022] Abhishek Balasubramaniam e Sudeep Pasricha. *Object Detection in Autonomous Vehicles: Status and Open Challenges*. 2022. arXiv: 2201.07706 [cs.CV] (citado na pg. 17).

[Bao *et al.* 2019] Wenbo Bao *et al.* "Depth-aware video frame interpolation". Em: (2019). arXiv: 1904.00830 [cs.CV] (citado nas pgs. 2, 21, 22).

[Carvalho *et al.* 2017] Leonardo Carvalho, Ricardo Marroquim e Emilio Vital Brazil. "Dilight: digital light table – inbetweening for 2d animations using guidelines". Em: 65 (2017), pgs. 31–44. issn: 0097-8493. doi: https://doi.org/10.1016/j.cag.2017.04.001. url: https://www.sciencedirect.com/science/article/pii/S0097849317300390 (citado na pg. 26).

[Chen e Zwicker 2022] Shuhong Chen e Matthias Zwicker. *Improving the Perceptual Quality of 2D Animation Interpolation*. 2022. arXiv: 2111.12792 [cs.CV] (citado nas pgs. 24, 25).

[Churchill e Chao 2019] Spencer Churchill e Brian Chao. *Anime Face Dataset*. https://www.kaggle.com/datasets/splcher/animefacedataset/data. dataset. Accessed: 2023-10-24. Out. de 2019. url: https://www.kaggle.com/datasets/splcher/animefacedataset/data (citado na pg. 3).

[Ci *et al.* 2018] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li e Zhongxuan Luo. "User-guided deep anime line art colorization with conditional adversarial networks". Em: *arXiv ePrints* (2018) (citado na pg. 3).

[Cirugeda 2023] Kevin Cirugeda. *SAKUGABOORU*. SAKUGABOORU. url: https://blog.sakugabooru.com/about/ (acesso em 11/09/2023) (citado na pg. 30).

[Cortes e Vapnik 1995] Corinna Cortes e Vladimir Naumovich Vapnik. "Support-vector networks". Em: *Machine Learning* 20 (1995), pgs. 273–297. URL: https://doi.org/10.1007/BF00994018 (citado na pg. 13).

[Eberle 2015] Kara Eberle. *Animation Styles: What Makes Anime Unique*. Show Me The Animation. 2015. URL: https://showmetheanimation.com/features/animation-styles-what-makes-anime-unique/ (acesso em 27/08/2023) (citado na pg. 3).

[Even *et al.* 2023] Melvin Even, Pierre Bénard e Pascal Barla. "Non-linear rough 2d animation using transient embeddings". Em: *Computer Graphics Forum* 42.2 (2023), pgs. 411–425. DOI: https://doi.org/10.1111/cgf.14771. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14771. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14771 (citado na pg. 25).

[Freitas Pereira *et al.* 2022] Tiago de Freitas Pereira *et al. Eight Years of Face Recognition Research: Reproducibility, Achievements and Open Issues*. 2022. arXiv: 2208.04040 [cs.CV] (citado na pg. 17).

[Fukushima 1969] Kunihiko Fukushima. "Visual feature extraction by a multilayered network of analog threshold elements". Em: *IEEE Transactions on Systems Science and Cybernetics* 5.4 (1969), pgs. 322–333. DOI: 10.1109/TSSC.1969.300225 (citado na pg. 15).

[Fukushima 1980] Kunihiko Fukushima. *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position*. 1980 (citado na pg. 17).

[Gavrikov 2020] Paul Gavrikov. *visualkeras*. https://github.com/paulgavrikov/visualkeras. 2020 (citado nas pgs. 28, 29).

[Goodfellow *et al.* 2014] Ian J. Goodfellow *et al.* "Generative adversarial networks". Em: (2014). arXiv: 1406.2661 [stat.ML] (citado na pg. 17).

[Greenemeier 2017] Larry Greenemeier. "20 years after deep blue: how ai has advanced since conquering chess". Em: *Scientific American* (2 de jun. de 2017). URL: https://www.scientificamerican.com/article/20-years-after-deep-blue-how-ai-has-advanced-since-conquering-chess/ (acesso em 04/07/2023) (citado na pg. 13).

[Horé e Ziou 2010] Alain Horé e Djemel Ziou. "Image quality metrics: psnr vs. ssim". Em: *2010 20th International Conference on Pattern Recognition*. 2010, pgs. 2366–2369. DOI: 10.1109/ICPR.2010.579 (citado na pg. 11).

[Isola *et al.* 2016] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou e Alexei A. Efros. "Image-to-image translation with conditional adversarial networks". Em: *arXiv ePrints* (2016) (citado nas pgs. 4, 14, 17, 18, 27, 28).

[Ivakhnenko e Lapa 1967] A.G. Ivakhnenko e V.G. Lapa. *Cybernetics and Forecasting Techniques*. Modern analytic and computational methods in science and mathematics. American Elsevier Publishing Company, 1967. ISBN: 9780444000200. URL: https://books.google.com.br/books?id=rGFgAAAAMAAJ (citado na pg. 14).

[Kang *et al.* 2020] Jaeyeon Kang, Younghyun Jo, Seoung Wug Oh, Peter Vajda e Seon Joo Kim. *Deep Space-Time Video Upsampling Networks*. 2020. arXiv: 2004.02432 [cs.CV] (citado na pg. 17).

[Kerlow 2004] I.V. Kerlow. *The Art of 3D: Computer Animation and Effects*. Wiley, 2004. ISBN: 9780471430360. URL: https://books.google.com.br/books?id=5G-5iyasSk8C (citado na pg. 7).

[Kong *et al.* 2022] Lingtong Kong *et al. IFRNet: Intermediate Feature Refine Network for Efficient Frame Interpolation*. 2022. arXiv: 2205.14620 [cs.CV] (citado nas pgs. 22, 23).

[Krizhevsky *et al.* 2012] Alex Krizhevsky, Ilya Sutskever e Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". Em: *Advances in Neural Information Processing Systems*. Ed. por F. Pereira, C.J. Burges, L. Bottou e K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (citado na pg. 40).

[Lecun *et al.* 1998] Y. Lecun, L. Bottou, Y. Bengio e P. Haffner. "Gradient-based learning applied to document recognition". Em: *Proceedings of the IEEE* 86.11 (1998), pgs. 2278–2324. DOI: 10.1109/5.726791 (citado na pg. 14).

[LeCun *et al.* 1989] Y. LeCun *et al.* "Backpropagation applied to handwritten zip code recognition". Em: *Neural Computation* 1 (1989), pgs. 541–551 (citado nas pgs. 16, 17).

[Li *et al.* 2022] Xiaoyu Li, Bo Zhang, Jing Liao e Pedro V. Sander. "Deep sketch-guided cartoon video inbetweening". Em: *IEEE Transactions on Visualization and Computer Graphics* (ago. de 2022), pgs. 2938–2952 (citado na pg. 25).

[lllyasviel 2017] lllyasviel. *sketchKeras*. Acessado em 2021-05-23. 2017. URL: https://github.com/lllyasviel/sketchKeras (citado nas pgs. 26, 30).

[Long *et al.* 2015] Jonathan Long, Evan Shelhamer e Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. 2015. arXiv: 1411.4038 [cs.CV] (citado na pg. 17).

[Margolis 2019] Eric Margolis. "The dark side of japan's anime industry". Em: *Vox* (2 de jul. de 2019). URL: https://www.vox.com/culture/2019/7/2/20677237/anime-industry-japan-artists-pay-labor-abuse-neon-genesis-evangelion-netflix (acesso em 13/07/2021) (citado na pg. 11).

[Narita *et al.* 2019] Rei Narita, Keigo Hirakawa e Kiyoharu Aizawa. "Optical flow based line drawing frame interpolation using distance transform to support inbetweenings". Em: *2019 IEEE International Conference on Image Processing* (2019) (citado na pg. 23).

[Plaut *et al.* 1986] David C. Plaut, Steven J. Nowlan e Geoffrey E. Hinton. "Experiments on learning by back propagation." Em: 1986. URL: https://api.semanticscholar.org/CorpusID:15150815 (citado na pg. 13).

[Ronneberger *et al.* 2015] Olaf Ronneberger, Philipp Fischer e Thomas Brox. "U-net: convolutional networks for biomedical image segmentation". Em: *arXiv ePrints* (2015) (citado nas pgs. 4, 19, 28).

[Schmidhuber 2022] Juergen Schmidhuber. *Annotated History of Modern AI and Deep Learning*. 2022. arXiv: 2212.11279 [cs.NE] (citado na pg. 14).

[Sherman 2019] Jennifer Sherman. "Japan's anime market hits record high for 6th consecutive year". Em: *Anime News Network* (29 de nov. de 2019). URL: https://www.animenewsnetwork.com/news/2019-11-29/japan-anime-market-hits-record-high-for-6th-consecutive-year/.153817 (acesso em 10/06/2021) (citado nas pgs. 1, 11).

[Shi *et al.* 2020] Min Shi *et al. Deep Line Art Video Colorization with a Few References*. 2020. arXiv: 2003.10685 [cs.CV] (citado na pg. 26).

[Siyao *et al.* 2021] Li Siyao *et al.* "Deep animation video interpolation in the wild". Em: *arXiv ePrints* (2021) (citado nas pgs. 2, 23, 24, 30).

[Strutz 2023] Tilo Strutz. *The Distance Transform and its Computation*. 2023. arXiv: 2106.03503 [cs.CV] (citado na pg. 25).

[Such *et al.* 2019] Felipe Petroski Such, Dheeraj Peri, Frank Brockler, Paul Hutkowski e Raymond Ptucha. *Fully Convolutional Networks for Handwriting Recognition*. 2019. arXiv: 1907.04888 [cs.CV] (citado na pg. 17).

[Sultana *et al.* 2018] Farhana Sultana, Abu Sufian e Paramartha Dutta. "Advancements in image classification using convolutional neural network". Em: *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE, nov. de 2018. DOI: 10.1109/icrcicn.2018.8718718. URL: https://doi.org/10.1109%2Ficrcicn.2018.8718718 (citado na pg. 17).

[Thomas e Johnston 1981] F. Thomas e O. Johnston. *The Illusion of Life: Disney Animation*. Disney Editions, 1981. URL: https://books.google.com.br/books?id=k5TMoAEACAAJ (citado na pg. 10).

[C. Wang *et al.* 2018] Chaoyue Wang, Chang Xu, Xin Yao e Dacheng Tao. "Evolutionary generative adversarial networks". Em: *arXiv ePrints* (2018) (citado na pg. 30).

[Z. Wang *et al.* 2004] Zhou Wang, A.C. Bovik, H.R. Sheikh e E.P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". Em: *IEEE Transactions on Image Processing* 13.4 (2004), pgs. 600–612. DOI: 10.1109/TIP.2003.819861 (citado na pg. 11).

[Williams 2009] R. Williams. *The Animator's Survival Kit: A Manual of Methods, Principles and Formulas for Classical, Computer, Games, Stop Motion and Internet Animators*. Faber & Faber, 2009. ISBN: 9780571238330. URL: https://books.google.com.br/books?id=ERDRkQEACAAJ (citado na pg. 8).

[Zhang *et al.* 2017] Lvmin Zhang, Yi Ji e Xin Lin. "Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan". Em: *arXiv ePrints* (2017) (citado nas pgs. 3, 14).