UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DANIEL AUGUSTO MULLER

COLORIZAÇÃO DE MAPAS DE JOGOS 2D UTILIZANDO UMA REDE ADVERSÁRIA GENERATIVA CONDICIONAL

PATO BRANCO

2023

DANIEL AUGUSTO MULLER

COLORIZAÇÃO DE MAPAS DE JOGOS 2D UTILIZANDO UMA REDE ADVERSÁRIA GENERATIVA CONDICIONAL

Colorization of 2D game maps using a condicional generative adversarial network

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Erick Oliveira Rodrigues

Coorientador: Prof. Dr. Dalcimar Casanova

PATO BRANCO

2023

DANIEL AUGUSTO MULLER

COLORIZAÇÃO DE MAPAS DE JOGOS 2D UTILIZANDO UMA REDE ADVERSÁRIA GENERATIVA CONDICIONAL

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação do Curso de Bacharelado em Engenharia de Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 12/junho/2023

Dalcimar Casanova
Dr.
Universidade Tecnológica Federal do Paraná

Jefferson Tales Oliva
Dr.
Universidade Tecnológica Federal do Paraná

Rúbia Eliza de Oliveira Schultz Ascari
Dra.
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2023

Para minha amada família e amigos, que me apoiaram e me encorajaram durante toda esta jornada, dedico este trabalho a vocês.

AGRADECIMENTOS

O presente trabalho não teria sido concluído sem a valiosa ajuda de diversas pessoas e/ou instituições, às quais presto meu sincero agradecimento. Certamente, essas palavras não são suficientes para expressar a gratidão que sinto em relação àqueles que contribuíram para esta importante fase da minha vida.

Gostaria de expressar minha gratidão à minha família pelo amor, incentivo e apoio em todos os momentos da minha vida. Ao meu orientador, agradeço pelos valiosos conselhos, orientações e confiança depositada em mim. Também quero agradecer a todos os professores e colegas do curso, que contribuíram direta ou indiretamente para a realização e conclusão deste trabalho. Por fim, quero agradecer a todos os demais que, de alguma forma, contribuíram para o meu crescimento pessoal, acadêmico e profissional.

RESUMO

Atualmente, o mercado de jogos tem assumido um papel de destaque na indústria do entretenimento, tornando-se uma das formas mais populares de entretenimento digital. Com avanços tecnológicos e o crescimento do acesso à internet e dispositivos móveis, os jogos eletrônicos têm conquistado uma audiência global e diversificada. Nesse contexto, o desenvolvimento de jogos torna-se uma atividade complexa e desafiadora, exigindo o domínio de diversas áreas, incluindo design, programação e arte. A colorização dos mapas desses jogos é uma etapa importante nesse processo, pois contribui para a criação de ambientes visualmente atrativos e imersivos ao jogador. Muitas vezes nessa etapa, são utilizados rascunhos pelo artista, que podem ser em escala de cinza, ou até mesmo somente os contornos do ambiente que se tornará o mapa completo no futuro. Este trabalho explora métodos que automatizem a colorização de mapas em jogos 2D com a utilização de Rede Adversária Generativa Convolucional (cGAN) através dos *frameworks pix2pix* e *CycleGAN*. Para isso foram criados três bancos de imagens: um deles com jogos variados e os outros dois de um mesmo jogo, porém em resoluções diferentes. Utilizando medidas de similaridade entre as imagens geradas pelos *frameworks* e as reais, foi possível observar que os mapas de um mesmo jogo e com resolução mais baixa, foram coloridos com um menor erro em quase todas as medidas. Com isso, em um âmbito de produção em massa, é possível obter uma diminuição no tempo de desenvolvimento artístico de jogos 2D, economizando o tempo dos artistas que podem focar em tornar o jogo ainda mais imersivo.

Palavras-chave: jogos; colorização; gan; *line-art*; extração de contornos.

ABSTRACT

Currently, the gaming market has taken a prominent role in the entertainment industry, becoming one of the most popular forms of digital entertainment. With technological advancements and the growth of internet access and mobile devices, electronic games have gained a global and diverse audience. In this context, game development has become a complex and challenging activity, requiring mastery of various areas, including design, programming, and art. The colorization of game maps is an important step in this process, as it contributes to the creation of visually appealing and immersive environments for players. Often, artists use sketches during this stage, which can be in grayscale or even just outlines of the environment that will become the complete map in the future. This work explores methods that automate the colorization of maps in 2D games using Conditional Generative Adversarial Networks (CGAN) through the pix2pix and CycleGAN frameworks. For this purpose, three image databases were created: one with various games and the other two with the same game but in different resolutions. By using similarity measures between the images generated by the frameworks and the real ones, it was observed that maps of the same game with lower resolution were colorized with less error in almost all measures. Consequently, in a mass production context, it is possible to reduce the artistic development time of 2D games, saving artists' time so they can focus on making the game even more immersive.

Keywords: games; colorization; gan; line-art; contour extraction.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

cGAN	Rede Adversária Generativa Convolucional
CNN	Rede Neural Convolucional
DCGAN	Rede Adversária Generativa Convolucional Profunda
GAN	Rede Adversária Generativa
MAE	Erro Absoluto Médio
MSE	Erro Médio Quadrático
PSNR	Relação Sinal-Ruído de Pico
RMSE	Raiz do Erro Médio Quadrático
RPG	*Role-playing game*

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

A indústria de jogos deixou de ser um mercado nichado, tanto por idade quanto por gênero, e se tornou uma mídia principal no mundo todo. Numa pesquisa realizada em 2020 foi apurado que 214 milhões de americanos jogam jogos eletrônicos e que 75% das casas norte-americanas tem ao menos um morador que é jogador (LUGRIS, 2020). Em relatório anual do *Global Games Market Report* sobre o mercado de jogos, foi constatado que em 2021 o valor investido globalmente nessa indústria foi de, aproximadamente, $196.8 bilhões de dólares, e tem previsão de crescer 14% até 2025 (NEWZOO, 2022). O valor agregado desse mercado é composto desde a publicação, distribuição, vendas das plataformas e componentes, e claro, o desenvolvimento dos jogos.

Com diversas maneiras de jogar disponíveis atualmente como *desktop*, *mobile*, consoles, jogos em nuvem e mais recentemente, a realidade virtual, houve um aumento exponencial no número de jogadores, e a demanda por jogos cresce em uma taxa proporcional, encurtando o tempo para os desenvolvedores entregarem o produto final bem elaborado. Além disso, a evolução rápida em componentes como placas de vídeo e processadores, possibilitou um grande avanço gráfico tanto nas plataformas como nos próprios jogos.

Um *game asset* é um recurso físico ou digital utilizado na criação de um jogo. Eles podem incluir um vasto número de elementos como modelos 3D, texturas, animações, efeitos sonoros, música e elementos de interface do usuário. Os *assets* são geralmente criados pelos desenvolvedores e artistas do jogo e são utilizados para construir os personagens e o ambiente do jogo. Eles podem ser criados com diversas ferramentas como *softwares* de modelagem e editores gráficos.

Hoje, com uma maior liberdade criativa, os artistas podem investir mais tempo na estilização, sombreamento e colorização dos *assets.* Apesar de serem tarefas a princípio simples, esses objetos podem possuir um grande número de cores, sombras e estilos diferentes, demandando de pessoas capacitadas, tempo e trabalho para produzirem bons resultados (ATTAIKI, 2019).

Por esses motivos, muitos artistas optam por fazer uma *line-art* inicial dos seus projetos para depois criarem uma arte mais complexa. Uma *line-art* é criada utilizando apenas linhas e formas, sem o uso de cores ou sombreamentos, normalmente é empregada em ilustrações, design e pinturas digitais. Além disso, as *line-arts* também são utilizadas em desenhos técnicos e diagramas, onde a precisão e clareza são importantes. Com essa arte criada a colorização da mesma se torna mais prática.

Colorização é o processo de adição de cores para uma imagem ou vídeo sem cor. Pode ser feito manualmente por um artista utilizando um *software* de edição de imagens ou automaticamente, utilizando algoritmos que analisam o conteúdo da imagem e aplicam cores baseadas

em certos padrões. Diversos trabalhos (SILVA *et al.*, 2019) (ISOLA *et al.*, 2017) (RODRIGUES; CLUA; VITOR, 2022) (ATTAIKI, 2019) (HATI *et al.*, 2019) já aplicaram diferentes técnicas de inteligência artificial capazes de colorir automaticamente imagens em escala de cinza ou contornos para inúmeras aplicações.

Com isso em mente, este trabalho apresenta uma solução para a colorização de mapas de jogos 2D, visando acelerar sua produção. Com o emprego algoritmos de redes neurais conhecidas como cGAN, é possível diminuir o tempo gasto no design de cores e estilos do mapa. Para isso, foram empregados *frameworks* populares desse tipo de rede chamados *pix2pix* e *CycleGAN* que, utilizando bancos de imagens de mapas em escala de cinza e de seus contornos, tiverem a tarefa de colorir essas imagens e consequentemente diminuir o tempo de desenvolvimento artístico do mesmo.

Por não ter sido encontrado na literatura algum trabalho que utiliza-se de tais *frameworks* para a colorização de mapas de jogos 2D, conclui-se ser uma pesquisa de caráter experimental. Além disso, a criação das bases de imagens, o estudo de diferentes métodos de extração de contornos e de medidas para a avaliação da colorização também são pesquisas alvo desse trabalho.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Desenvolver um método de colorização automática de mapas em jogos 2D, testando diferentes paradigmas de extrações de contornos e de colorização, utilizando redes generativas adversárias condicionais através dos *frameworks pix2pix* e *CycleGAN*.

1.2.2 OBJETIVOS ESPECÍFICOS

- Criar uma base de dados com imagens de mapas de jogos 2D diversos e outras duas de um único jogo porém de resoluções diferentes;
- Realizar o pré-processamento para adequar as imagens ao formato necessário;
- Realizar a extração de contornos das imagens, testando diferentes técnicas e variações de parâmetros;
- Executar o treinamento dos principais *frameworks* de colorização, *pix2pix* e *CycleGAN*, com imagens em escala de cinza e de contornos com o objetivo de colorir as imagens automaticamente;

- Comparar os resultados gerados visualmente e matematicamente através de diferentes medidas de similaridade;

1.3 ESTRUTURA DO TRABALHO

O trabalho apresenta em seu primeiro capítulo a introdução do problema, objetivos gerais e específicos, assim como a justificativa para sua realização. Já no segundo capítulo, o referencial teórico é exposto juntamente com os trabalhos relacionados. A metodologia e materiais empregados são discutidos no terceiro capítulo. Em sequência no quarto e quinto capítulo são apresentadas as configurações experimentais e os resultados, além de uma comparação entre eles. No penúltimo capítulo são feitas as considerações e discussões finais e também aprimoramentos para trabalhos futuros. Ao final encontram-se as referências utilizadas no decorrer do trabalho.

2 REFERENCIAL TEÓRICO

Nesse capítulo, será abordado de forma introdutória as principais teorias, métodos e algoritmos envolvidos e aplicados nesse trabalho. No início é discutido o conceito de aprendizado de máquina (2.1), e logo após uma introdução à redes neurais (2.2). O *deep learning* (2.3) é explicado na sequência, juntamente a organização e arquitetura de uma Rede Adversária Generativa (GAN) (2.4), conceitos considerados fundamentais para o entendimento do trabalho. Os *frameworks pix2pix* e *CycleGAN* (2.5) são explicados no seguimento, e por fim métodos de extração de contornos (2.6), métodos de similaridade (2.7) e trabalhos relacionados (2.8) concluem este capítulo.

2.1 APRENDIZADO DE MÁQUINA

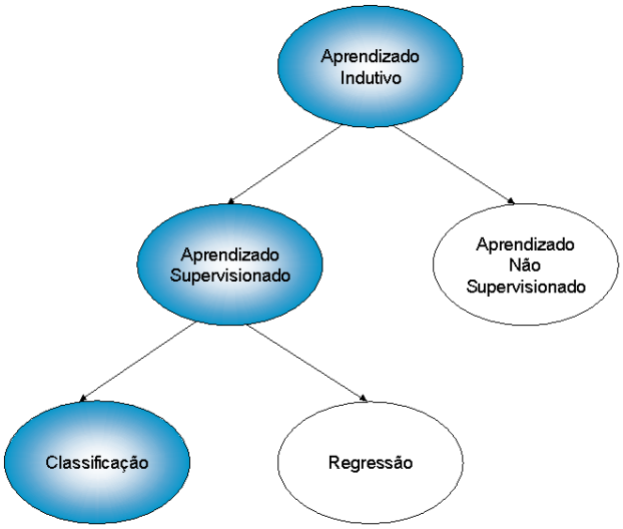
A busca pela resolução de problemas cientificamente mais complexos de maneira mais simples, resultou na criação do conceito de um tipo diferente de máquinas, como menciona Mahesh (2019). Para Janiesch, Zschec e Heinrich (2021), Aprendizado de Máquina descreve a capacidade de sistemas de aprenderem a partir de um conjunto de dados de treino específico, para automatizar o processo de desenvolvimento de modelos analíticos que resolvam tarefas associadas ao problema.

Um sistema de aprendizado indutivo ou inteligente é um programa que toma decisões baseado em experiências acumuladas através de soluções bem sucedidas de problemas anteriores (MONARD; BARANAUSKAS, 2003). O computador cria um modelo que pode ser usado para explicar ou representar um novo dado (do mesmo domínio do conjunto de dados inicial) nunca antes visto pelo sistema que será apresentado futuramente, como explica Brunialti *et al.* (2015), e por isso são conhecidos também como sistemas inteligentes.

O aprendizado indutivo geralmente pode ser divido em duas classes: supervisionado e não-supervisionado. Um sistema não-supervisionado analisa os dados sem rótulos e tenta agrupá-los de uma maneira lógica, e posteriormente estudá-los para compreender o que os agrupamentos representam para o problema em questão, como explicam Monard e Baranauskas (2003). Já os supervisionados fornecem ao algoritmo um conjunto de dados com exemplos já rotulados com a resposta esperada de saída, e podem ser de duas diferentes categorias: classificação e regressão. Na classificação são preditos resultados categóricos (por exemplo, determinar se um usuário pode pedir crédito ao banco), e na regressão, um valor contínuo (por exemplo, estipular o valor de um imóvel). Essa arquitetura é apresentada na Figura 1 que destaca em azul os tipos de aprendizados utilizados nesse trabalho, além de demonstrar que os diversos tipos de aprendizado de máquina derivam do aprendizado indutivo.

Com diferentes tipos de algoritmos e diversas metodologias, escolher quais técnicas se encaixam melhor com o problema não é uma tarefa simples. Mahesh (2019) explica que cientistas de dados fazem suas escolhas para o aprendizado de máquina baseadas em diversas

Figura 1 – Hierarquia do aprendizado de máquina.



Fonte: Autoria própria (2023).

medidas, como tipo de problema, número de variáveis, tipo de modelo e tamanho do banco de dados.

2.2 REDES NEURAIS

De forma geral o cérebro humano, mais precisamente o sistema nervoso, é composto de centenas de bilhões de neurônios interconectados que recebem e enviam informações a outros neurônios por meio de impulsos elétricos. É por meio desse sistema que temos a capacidade de aprender e é desse modo que uma rede neural artificial funciona. Segundo Keller, Liu e Fogel (2016), uma rede neural é uma estrutura distribuída altamente paralelizada que tem a habilidade de aprender e generalizar a partir de inúmeros dados de entrada.

2.2.1 PERCEPTRON DE CAMADA ÚNICA

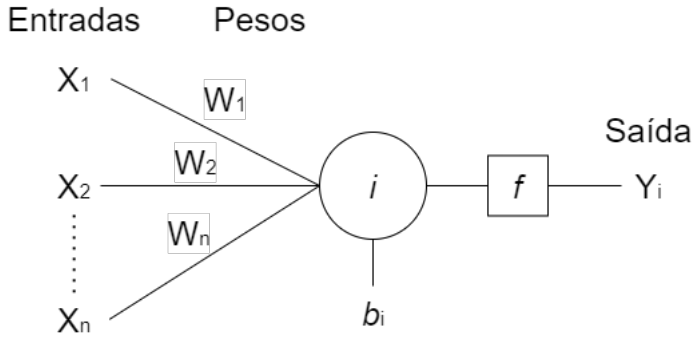
A primeira concepção de um neurônio artificial foi feita em 1957 por Frank Rosenblatt Rosenblatt (1958). Baseado no sistema nervoso presente no cérebro humano e receptores do sistema biológico, o cientista criou o perceptron, o primeiro neurônio artificial de apenas uma camada, fundamentado em uma simples fórmula matemática de somas, multiplicações e comparações (KANAL, 2003).

A estrutura de um perceptron de camada única pode ser observada na Figura 2. Na esquerda encontram-se as *n* entradas do sistema (*X*), onde cada uma é atribuída a um peso individual (*W*), que então são multiplicadas aos seus respectivos pesos e depois aplicados em um somatório (*i*) (KANAL, 2003). A variável *bi* é chamada de viés, ou *bias*, e tem geralmente seu valor atribuído em 1. Posteriormente, uma função (*f*) de escolha do desenvolvedor é destinada

para funcionar como o limiar, ou *threshold*, do sistema. Caso seja maior que esse valor, a saída (*Y*) é 1, caso contrário 0, categorizado assim como um algoritmo de saída binária.

Com essa limitação, o perceptron de camada única só pode ser utilizado para classificar até duas classes de padrões diferentes. Porém, o surgimento de novos desafios computacionais, que não poderiam ser resolvidos linearmente, levaram o neurônio de camada única a ser geralmente substituído pelo perceptron de múltiplas camadas, onde esse problema é solucionado.

Figura 2 – Estrutura de um neurônio artificial - Perceptron de camada única. Vetor de dados *X* é atrelado à um peso *W*, e caso o somatório da multiplicação de *X* por *W* seja maior que a função limiar *f* é gerada a saída da função *Y*.



Fonte: Autoria própria (2023).

2.2.2 PERCEPTRON DE MÚLTIPLAS CAMADAS

Em 1986, David Rumelhart e James McClelland (SHINDE; SHAH, 2018) apresentaram um modelo de perceptron de múltiplas camadas, mostrando que, com a concatenação de diversos neurônios de camadas únicas, era possível aproximar qualquer função matemática. Além disso, a rede conseguia modelar funções altamente não-lineares e generalizar precisamente dados nunca antes vistos pelo algoritmo, algo que parecia impossível poucos anos antes. Essa descoberta foi primordial para a evolução das redes neurais, como discute Gardner e Dorling (1998).

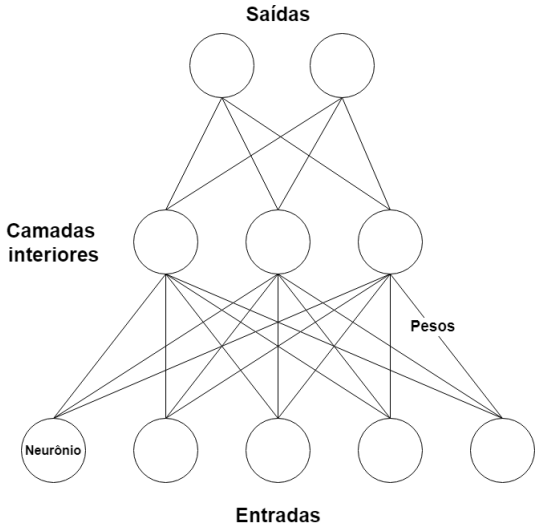
Um perceptron de múltiplas camadas é composto de um sistema de perceptrons de camadas únicas interconectados, que compõem as camadas ocultas ou internas da rede. Os nós são conectados por sinais de saída e pesos que se ajustam proporcionalmente aos erros da rede, sendo esse chamado de processo de aprendizado (WANG, 2003). Considerando que a saída de um neurônio é a entrada de outro, implica-se que existe uma direção de informações e por isso os perceptrons multi-camadas mais simples são conhecidos como modelos *feed-forward* de rede neural (GARDNER; DORLING, 1998).

Como explica Gardner e Dorling (1998), a arquitetura dessa rede é variável, e a escolha da quantidade de camadas internas pode levar a problemas. Uma seleção de poucas camadas internas pode tornar a solução do problema lenta ou até incapaz de ser resolvida, enquanto uma

escolha excessiva das mesmas pode causar *over-fitting*, ou super-ajuste, dos dados de treinamento (RAMCHOUN *et al*., 2016). A Figura 3 representa de maneira simplificada a arquitetura dessa rede, onde na parte inferior estão os neurônios de entrada de primeira camada, que são conectados por pesos até os da camada interior ou oculta, que levam a uma ou várias saídas do sistema. A Figura 3 dispõe a arquitetura simplificada de um perceptron de múltiplas camadas, ela contém na parte inferior os neurônios de entrada do sistema conectados via pesos às camadas interiores no centro e posteriormente resultando nas saídas na parte superior.

O poder matemático dessas redes é tão grande que no teorema descrito por Cybenko (1989), o autor afirma que uma rede neural com uma única camada oculta pode aproximar qualquer função contínua de um número finito de variáveis de entrada, desde que use funções de ativação não-lineares. Essa propriedade é fundamental, pois permite que as redes neurais sejam aplicadas em uma ampla gama de problemas de aprendizado, desde tarefas simples até problemas complexos de reconhecimento de padrões e processamento de dados.

Figura 3 – Arquitetura simplificada de uma rede neural multicamadas, composta de neurônios, pesos e saídas.



Fonte: Autoria própria (2023).

2.3 DEEP LEARNING

Técnicas de aprendizado de máquinas convencionais requerem o domínio e engenharia da área que está sendo estudada, pois é preciso extrair as características dos dados e transformá-los em vetores, para o treinamento da rede neural. O *deep learning*, também conhecido como aprendizado por representação, refere-se a um conjunto de métodos que permitem a máquina ser alimentada com dados no seu estado natural e automaticamente aprender representações necessárias para sua classificação, constituindo-se de múltiplas camadas de abstração, como discorrem LeCun Bengio (2015) e Rusk (2016).

O ponto de maior importância no aprendizado por representação é que suas inúmeras camadas internas não foram construídas por um engenheiro ou especialista, e sim pela própria máquina. Isso é possível pois o sistema indica a ele mesmo quais parâmetros internos mudar em cada camada anterior a fim de minimizar o erro, tarefa chamada de *backpropagation* (HAO; ZHANG; MA, 2016). As camadas inferiores, próximas dos dados de entrada, aprendem características simples, como formas geométricas e cores, enquanto as mais afastadas aprendem características complexas derivadas das camadas anteriores (SHINDE; SHAH, 2018). Além disso, é um estudo de modelos que envolvem uma quantidade de dados muito maiores comparados aos que compõem o tradicional aprendizado de máquina (GOODFELLOW; BENGIO; COURVILLE, 2016).

Como citam Shinde e Shah (2018), as razões para o amplo crescimento desse campo de estudo se deve ao aumento drástico do processamento de chips como placas de vídeo, o barateamento desses dispositivos e dos avanços recentes em pesquisas de aprendizado de máquina e processamento de informações. Existem diversos exemplos de aplicações do *deep learning* atualmente como: o mecanismo de busca do *Google*, descobrimento de novos remédios, imageamento médico, carros autônomos, previsão do mercado de ações, geração de artes a partir de texto, reconhecimento de objetos, voz e áudio (SHINDE; SHAH, 2018).

2.4 REDES ADVERSÁRIAS GENERATIVAS

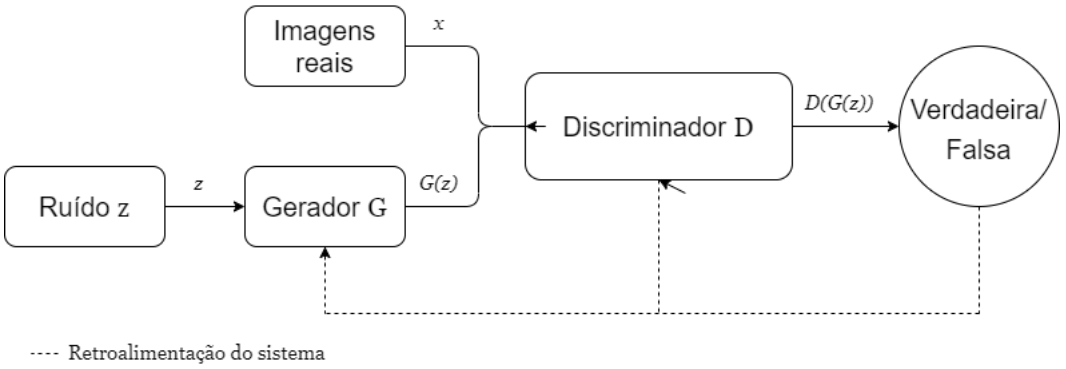
GAN (do inglês, *Generative Adversarial Network*), inicialmente introduzida por Goodfellow *et al.* (2014), é um dos vários ramos de estudo do *deep learning*. O ponto que difere uma GAN das demais redes neurais, é o fato dela contar com duas redes neurais competindo uma contra a outra. Uma dessas redes é chamada de gerador (G), que tem como objetivo criar imagens realistas a partir de um ruído (z) de entrada, que sejam suficientemente reais para enganar a outra rede. O discriminador (D), tem a tarefa de diferenciar duas imagens, a gerada e uma original, e determinar qual é a verdadeira (CRESWELL *et al.*, 2018).

O treinamento das duas redes é simultâneo, pois as duas utilizam do mesmo parâmetro para atualizar suas camadas. O gerador tem seu treino restrito pois ele não tem acesso às imagens reais (x) que deve tentar reproduzir, então é alimentado pela resposta do discriminador à sua tentativa de enganá-lo, ou seja, o erro. Já o discriminador geralmente é binário, e tem acesso tanto às figuras genuínas quanto às geradas pela outra rede, e é treinado pelo seu próprio erro, caso tenha acertado se a imagem é original ou não (WANG *et al.*, 2017). A arquitetura dessa rede pode ser observada na Figura 4.

As duas redes são tipicamente implementadas como redes multi-camadas, podendo ser totalmente conectadas, convolucionais ou condicionais. Em uma rede totalmente conectada, é utilizado o modelo tradicional de redes neurais multicamadas, com atualização de pesos e *backpropagation* (representado com linhas pontilhadas na Figura 4), como explica Hsu, Li e Psaltis (1990). As redes convolucionais são geralmente encontradas no seu modelo Rede Adversária

Generativa Convolucional Profunda (DCGAN), com camadas convolucionais e de *pooling* empilhadas (RADFORD; METZ; CHINTALA, 2015), seguidas por uma ou mais camadas totalmente conectadas (LIU *et al.*, 2017). Já uma cGAN, contém dados adicionais de entrada, como imagens ou texto, que são concatenados ao ruído e possibilitam mais controle sobre as amostras geradas (CRESWELL *et al.*, 2018).

Figura 4 – Arquitetura de uma Rede GAN: Gerador *G* recebe um ruído *Z* como entrada, então o Discriminador *D* tenta diferir imagens reais e falsas, a partir do acerto ou erro de *D*, *G* têm seu treinamento aprimorado.



Fonte: Autoria própria (2023).

Para Mirza e Osindero (2014), as GAN podem ser modificadas para seu modelo condicional (cGAN) caso o gerador e o discriminador sejam apresentados à uma informação extra, como rótulos de classes, adicionando uma camada adicional de entrada. Nessa configuração, o ruído de entrada e o dado complementar são combinados em uma única representação e alimentados ao gerador. Para o discriminador, tanto a nova informação quanto a original são apresentadas para uma função discriminativa como entrada (MIRZA; OSINDERO, 2014).

2.5 PIX2PIX E CYCLE-GAN

Uma das grandes áreas de estudo do *deep learning*, pelas suas inúmeras aplicações em diversas áreas, são as tarefas de tradução imagem-imagem. A tradução imagem-imagem é o processo em que um algoritmo é usado para converter uma imagem de um domínio para outro. Esse processo é usualmente utilizando no campo de visão computacional e aprendizado de máquina, e pode ser aplicado para inúmeras tarefas como, transferência de estilos, super-resolução e colorização de imagens (ALOTAIBI, 2020).

Porém, até alguns anos atrás, a implementação de métodos para a resolução desse problema era de extrema dificuldade, dado que cada aplicação tinha fórmulas de erros e entradas diferentes. Dessa forma, a fim de tornar a utilização da GAN mais prática, *frameworks* foram desenvolvidos (ISOLA *et al.*, 2017) (ZHU *et al.*, 2017) para facilitar e generalizar os problemas imagem-imagem. A criação dessas ferramentas expandiu exponencialmente os experimentos

na computação visual e gráfica, visto que distancia o usuário da engenharia pura do algoritmo enquanto o *framework* se modela automaticamente, abrindo espaço para mais pesquisas.

2.5.1 PIX2PIX

Proposto por Isola *et al.* (2017), o *pix2pix* é um poderoso *framework* de cGAN, que tem como objetivo a tarefa de tradução imagem-imagem. Seu treinamento consiste na apresentação de exemplos de pares de imagens (*Xi* e *Yi*) de estilos diferentes, e que ao seu final deve transferir o estilo com sucesso de um para o outro, detalhado na Figura 5.

Figura 5 – Treinamento com pares de imagens utilizado no *pix2pix*. A coluna da esquerda representa os contornos das imagens e a da direita são as imagens reais.

Fonte: Zhu *et al.* (2017).

Um dos problemas encontrados no gerador em uma GAN convencional, é que ele requer que a informação passe por todas as camadas que progressivamente vão reduzindo as amostras, até a camada de "gargalo", onde o processo é invertido. Como cita Isola *et al.* (2017), nesse tipo de rede é necessário que toda a informação passe pela camada de "gargalo". Uma forma de otimizar esse problema é a implementação de uma *"U-Net"* (RONNEBERGER; FISCHER; BROX, 2015), que pula certas camadas intercaladas com o intuito de desviar do gargalo de dados. Já o discriminador foi desenvolvido para restringir sua atenção em pequenos focos na imagem chamados de *patchs*, de tamanhos que podem ser muito menores que o tamanho total da imagem, e que tem menos parâmetros, agilizando o processo. Ao final, cada bloco de informação é ponderado se é real ou gerada pelo gerador, e ao final é feita uma média entre todos os *patchs*, que compõem saída final do discriminador, chamado assim de *"PatchGAN"*. Algumas aplicações do *pix2pix* podem ser observadas na Figura 6, onde são expostas imagens de entrada, a imagem real e a imagem gerada pelo *framework*.

Figura 6 – Diversidade de aplicações que utilizam do *framework pix2pix*.

Fonte: Adaptado de Isola *et al.* (2017).

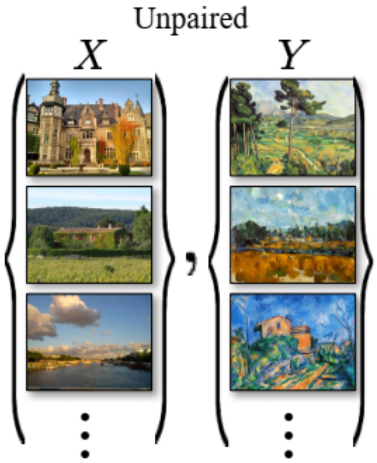
2.5.2 CYCLE-GAN

Outro *framework* para a solução de tradução imagem-imagem foi proposto por Zhu *et al.* (2017). Partindo da base do *pix2pix*, detalhado na seção anterior, o *CycleGAN* coleta características de uma coleção de imagens (X), e então aprende como traduzir elas para um outro conjunto de imagens (Y), utilizando da mesma arquitetura de uma GAN, com gerador condicional e discriminador *"PatchGAN"*.

Sua principal diferença comparado ao *pix2pix* é o método de mapeamento, que não depende de um par de imagens no treinamento, ou seja, são recebidos como entrada dois conjuntos de imagens distintos, como pode ser visto na Figura 7, onde X e Y são os conjuntos de imagens de entrada. Além disso, outro ponto importante de mudança entre os *frameworks*, é não ter uma relação pixel-a-pixel como encontrado no *pix2pix* e seu caráter cíclico. No *CycleGAN*, imagens traduzidas fazem o caminho reverso no algoritmo e deveriam, teoricamente, voltar à imagem original, método chamado de tradução reversa ou consistência cíclica (ZHU *et al.*, 2017). Essa operação é exemplificada na Figura 8, onde a entrada (x) é o conjunto de imagens reais, a saída ($G(x)$) do *CycleGAN* e por ultimo a reconstrução ($F(G(x))$) feita pelo algoritmo.

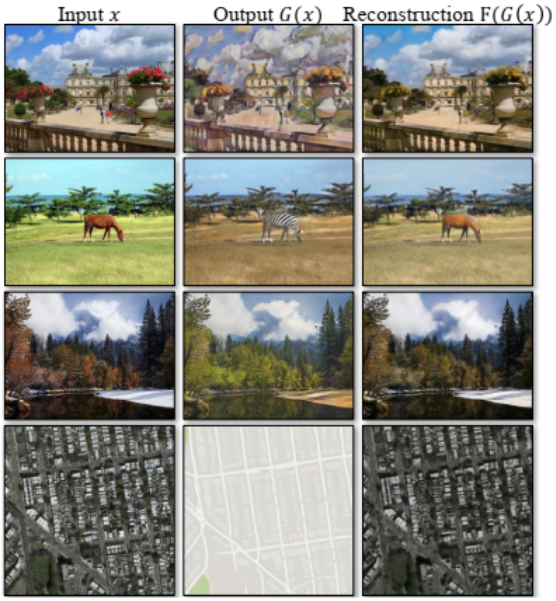
Assim como no *pix2pix*, a vantagem de existir um *framework* para a tradução imagem-imagem são suas inúmeras aplicações, além da praticidade do seu uso. O *CycleGAN* por exemplo, funciona muito bem na transferência de estilos entre imagens, troca de estações do ano, gerações de fotos a partir de pinturas, dentre outras. A utilização do *CycleGAN* na transformação de estilos como: pinturas para fotos reais, zebras para cavalos, verão para inverno, são mostrados na Figura 9. É importante ressaltar que todas as traduções são válidas para a volta, ou seja, o caminho inverso também pode ser seguido.

Figura 7 – Treinamento sem pares de imagens utilizado no *CycleGAN*. A coluna da esquerda exibe as imagens reais que serão traduzidas para o estilo da coluna da direita.



Fonte: Zhu *et al.* (2017).

Figura 8 – Exemplo de tradução reversa aplicada no *CycleGAN*: À esquerda as imagens reais *x*, ao centro sua tradução *G(x)*, e à direita sua reconstrução *F(G(x))*.



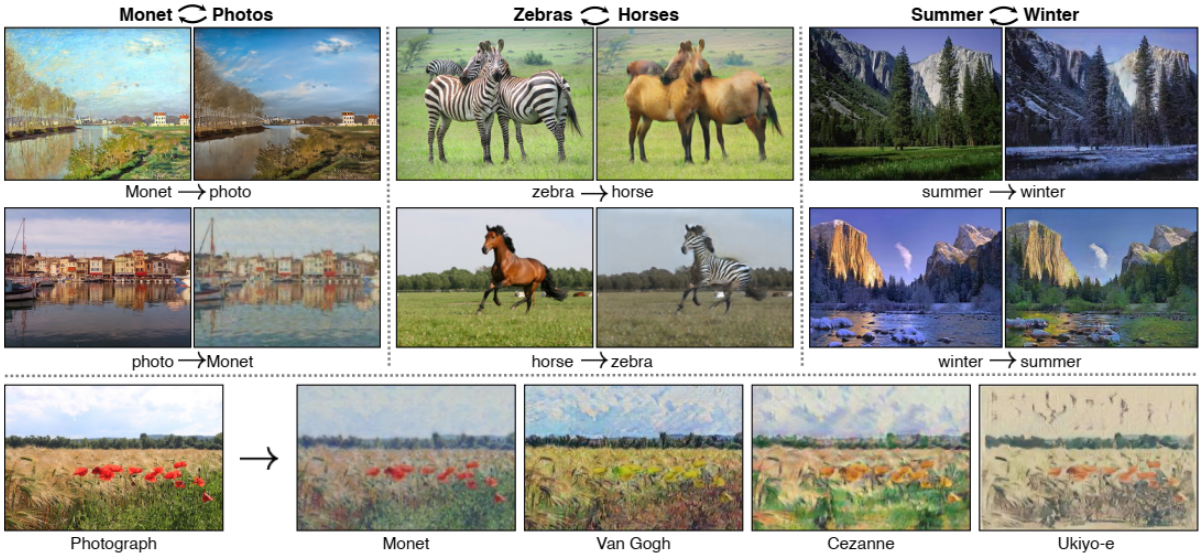
Fonte: Zhu *et al.* (2017).

2.6 EXTRAÇÃO DE CONTORNOS

Para ambos os *frameworks* apresentados, é necessário um estilo de camada a ser traduzido ou colorizado. Para gerar imagens coloridas é comum pensar somente em imagens em escala de cinza, mas é possível gerar cores com apenas algumas linhas de contorno da imagem. Para isso, foram desenvolvidos, ao longo dos anos, diversos métodos para detecção de bordas e contornos de objetos em imagens (MING; LI; HE, 2016).

A utilização de diversos tipos de redes neurais para a extração de contornos de imagens se tornou alvo de inúmeros trabalhos científicos na computação pois, embora seja um conceito

Figura 9 – Exemplos de aplicações do *framework CycleGAN* na conversão de estilos entre imagens.



Fonte: Zhu *et al.* (2017).

simples ao senso comum humano, não existe uma definição matemática exata. Segundo Ming, Li e He (2016), um contorno é definido como a linha, borda ou curva que representa a forma de um objeto. Sua definição ainda se relaciona a outros dois conceitos que são: borda, que é a mudança de pixel de um objeto ou superfície para outra, e aresta, que é representada pela mudança de intensidade na imagem, podendo ser detectada por alterações de brilho ou cor (MARTIN; FOWLKES; MALIK, 2004). Para Papari e Petkov (2011), a criação de modelos computacionais que entendam os contornos de uma imagem vai além do entendimento do sistema visual humano, havendo aplicações práticas desde reconhecimento de objetos até análises de imagens médicas.

A fim de extrair os contornos de imagens, Wen Chuan Lin e Cui (2022) aplicou uma Rede Neural Convolucional (CNN), devido a sua ampla capacidade de aprendizado e funções poderosas. Um dos métodos mais populares e utilizado até hoje para a extração de contornos foi proposto em 1986, conhecido como detector de bordas *Canny* (CANNY, 1986). Outros trabalhos como Winnemöller, Kyprianidis e Olsen (2012) propuseram o *XDoG*, uma ferramenta baseada da diferença gaussiana, que conseguia além de extrair contornos, estilizar a imagem ao alterar alguns parâmetros de limiarização. Além dos contornos citados, bilbiotecas de visão computacional em *Python* como a *OpenCV* (BRADSKI, 2000) oferecem diversas funções para a extração de contornos utilizando valores de *threshold*. Uma comparação de alguns métodos de extração de contornos é demonstrada na Figura 10, onde o autor define os parâmetros de cada método manualmente para qualidade ótima e visualização mútua entre as imagens (WINNEMöLLER; KYPRIANIDIS; OLSEN, 2012).

É possível observar que os diferentes métodos oferecem resultados diversos. No *Canny* as linhas de contorno são menos detalhadas e por isso geram menos ruído quando comparados

Figura 10 – Diferentes métodos de extração de contornos. Da esquerda à direita: imagem real, *Canny*, *XDoG* com *threshold* adaptado, *XDoG* estilizado.

Fonte: Adaptado de Winnemöller, Kyprianidis e Olsen (2012).

ao *XDoG* ao lado, que por sua vez oferecem linhas mais destacadas de contorno. Para a aplicação na extração de contornos em mapas de jogos 2D, é necessário extrair as principais bordas da imagem e gerar a menor quantidade de ruído possível, por isso foram testados diferentes algoritmos e parâmetros para encontrar a melhor solução.

2.7 MEDIDAS DE SIMILARIDADE ENTRE IMAGENS

As medidas de similaridade de imagens permitem comparar as imagens geradas pelos modelos com as imagens originais e avaliar a fidelidade da geração de cores e texturas. Existem diversas medidas de similaridade de imagens disponíveis, cada uma com sua própria definição e aplicação. Dentre as medidas de similaridade de imagens mais comuns, podemos citar a Distância Euclidiana, a Similaridade por Cosseno, o Erro Absoluto Médio (MAE) (*mean absolute error*), o Erro Médio Quadrático (MSE) (*mean squared error*), o Raiz do Erro Médio Quadrático (RMSE) (*root mean square deviation*) e Relação Sinal-Ruído de Pico (PSNR) (*peak signal-to-noise ratio*) (NIXON; AGUADO, 2012).

A Distância Euclidiana, definida na Equação (1), é uma medida de similaridade utilizada para comparar imagens pixel a pixel. Nesse caso o valor *x* é o valor do pixel da imagem real, e *y* o valor do pixel da imagem falsa. Ela calcula a diferença entre os valores de intensidade dos pixels correspondentes em ambas as imagens e, em seguida, calcula a raiz quadrada da soma dos quadrados dessas diferenças. A similaridade entre duas imagens aumenta à medida que a distância euclidiana entre elas diminui, sendo essa métrica amplamente empregada em situações que envolvem reconhecimento de padrões, análise e processamento de imagens (SANTINI; JAIN, 1999).

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

Definido por Nixon e Aguado (2012), a Similaridade por Cosseno é uma medida utilizada para comparar vetores de características extraídos de imagens. Ela calcula o cosseno do ângulo

entre os dois vetores e produz um valor entre 0 e 1, utilizando a Equação (2), indicando a similaridade entre os vetores. Quanto mais próximo de 1 o valor, maior é a similaridade entre os vetores e, portanto, entre as imagens. Essa medida é comumente usada em aplicações de classificação de imagens e recuperação de informações. Assim como na equação anterior, valor *x* é o pixel da imagem real, e *y* o valor do pixel da imagem falsa.

$$sim(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\|\|\boldsymbol{y}\|} \tag{2}$$

Algumas medidas são usadas para avaliar a diferença entre as intensidades de pixel em duas imagens. Na definição de (JI; GALLO, 2006), o MAE é entre os valores de pixel, enquanto o RMSE é a raiz quadrada do MSE. Quanto menor o valor dessas medidas, menor é a diferença entre as intensidades de pixel das imagens e, portanto, maior é a similaridade entre elas. Essas técnicas são comumente empregadas em situações que envolvem a recuperação ou melhoria de imagens. As fórmulas para MAE, MSE e RMSE são dadas, respectivamente, pelas Equações (3),(4) e (5), onde *y* representa o valor real do pixel e $\hat{y}$, o predito.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

$$RMSE = \sqrt{MSE} \tag{5}$$

A PSNR, é uma medida de qualidade de imagem empregada na avaliação de imagens reconstruídas ou comprimidas em relação à imagem original (SILVA; PANETTA; AGAIAN, 2007). Seu objetivo é medir a proporção entre o sinal (ou seja, a informação da imagem) e o ruído (ou seja, a perda de informação na imagem reconstruída ou comprimida), sendo que valores maiores indicam uma melhor qualidade de imagem. A PSNR é amplamente utilizada em situações que envolvem compressão e transmissão de imagens, e pode ser definida pela Equação (6), onde *MAX* é o valor máximo possível para a intensidade do pixel na imagem, geralmente definido como 255 para imagens de 8 bits por canal e o *MSE* é calculado pela Equação (4).

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{6}$$

2.8 TRABALHOS RELACIONADOS

Atualmente, um editor de imagens tem diversas ferramentas, dentre elas converter uma imagem sem cor para uma colorida, porém, para fazer isso adequadamente, é necessário informações de associações em comum entre cores e objetos no mundo (LUO; STRAKA, 2017) que não são fáceis de serem extraídos nem implementados corretamente. Este é um dos problemas

categorizados na visão computacional como tradução imagem-imagem, que pode ser definido como conversões de representação de uma imagem para outra, a partir de suficientes dados de treinamento (ISOLA *et al.*, 2017).

A colorização é um dos principais tópicos explorados de tradução imagem-imagem. Em algoritmos para esse propósito como os que estão disponíveis nos *frameworks pix2pix* e *CycleGAN*, é recebida uma imagem ou vídeo sem cor como entrada e técnicas de aprendizado de máquina são utilizadas a fim de gerar uma versão colorida da gravura automaticamente. As principais aplicações desse método incluem: troca de cores de objetos em imagens e vídeos, dar vida a mídias antigas e quadrinhos em escala de cinza. O propósito final dessa técnica é gerar a colorização mais natural possível, capaz de enganar até mesmo o olho humano (KUMAR; WEISSENBORN; KALCHBRENNER, 2021). Uma abordagem natural para solucionar problemas imagem-imagem é mapear as imagens de saída dada a entrada, usando modelos generativos profundos, como a cGAN, explorado em capítulos anteriores (SAHARIA *et al.*, 2021). Exemplos de colorização podem ser observados na Figura 11, onde as figuras da esquerda são imagens em escala de cinza e as demais à direita são diferentes colorizações da mesma entrada geradas pelo trabalho de Kumar, Weissenborn e Kalchbrenner (2021).

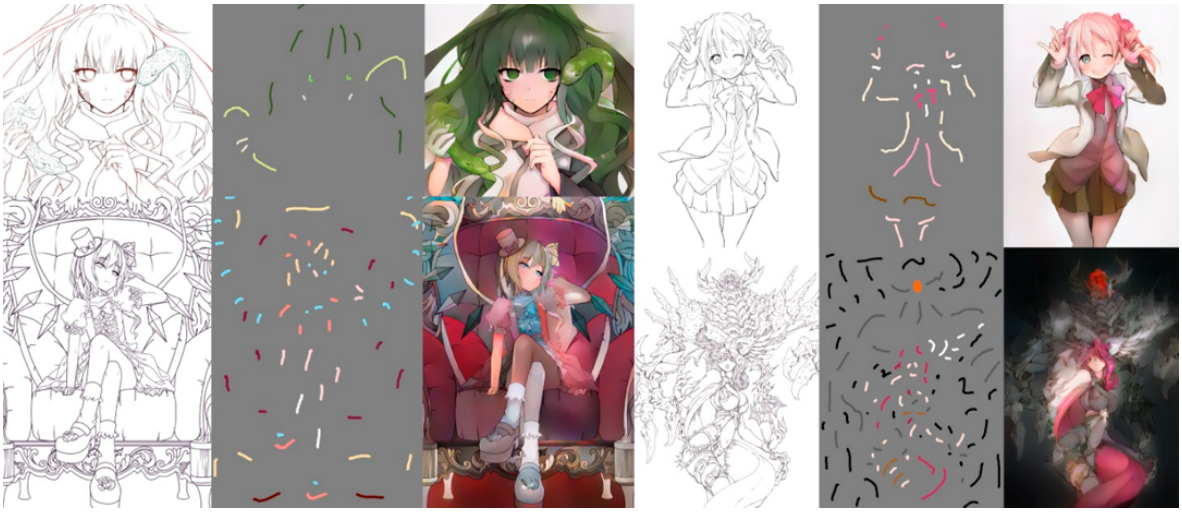
Figura 11 – Exemplos de aplicação da colorização em imagens em escala de cinza.

Fonte: Adaptado de Kumar, Weissenborn e Kalchbrenner (2021).

Alguns trabalhos demonstram que a utilização de uma cGAN pode ser eficaz para a resolução de problemas envolvendo colorização. No trabalho de Hati *et al.* (2019), foi desenvolvido uma ferramenta chamada de *PaintsTorch*. Empregando uma rede GAN com dois geradores, foi construída uma arquitetura para a colorização guiada de *line arts* de personagens de anime, apresentados na Figura 12, podendo ser observado os contornos extraídos da imagem à esquerda, uma dica de cores fornecida ao algoritmo ao centro e a colorização concluída à direita. Neste caso o algoritmo recebe os contornos e as dicas de cores como entrada e gera uma imagem baseada nesses dados. Já Attaiki (2019), aplicou o *framework pix2pix* para colorir mangás

para o estilo anime à esquerda, e transformar imagens de anime para o estilo mangá à direita, obtendo os resultados da Figura 13.

Figura 12 – Resultados da colorização guiada no trabalho de Hati *et al.* (2019).



Fonte: Adaptado de Hati *et al.* (2019).

Figura 13 – Resultados do trabalho de Attaiki (2019).

Fonte: Adaptado de Attaiki (2019).

Figura 14 – Resultados no trabalho de Rodrigues, Clua e Vitor (2022). Da esquerda à direita: contorno com dicas de cores, colorização pelo *pix2pix*, colorização pelo *CycleGAN*, junção *pix2pix* e *CycleGAN*, imagem real.

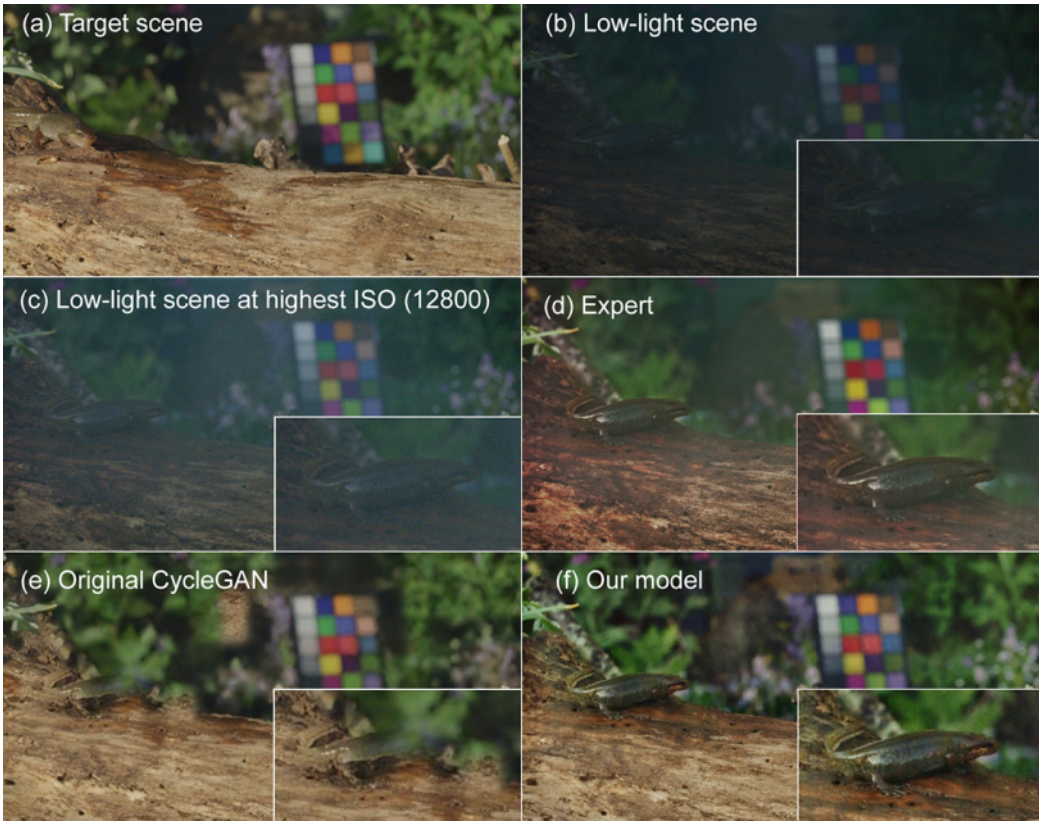
Fonte: Adaptado de Rodrigues, Clua e Vitor (2022).

No trabalho de Rodrigues, Clua e Vitor (2022), foram aplicados ambos os *frameworks, pix2pix* e *CycleGAN*, na colorização guiada (com dicas de cores) de *Fakemons*, monstros do es-

tilo *Pokemon* criados pela comunidade ao redor do mundo, comparando seus resultados lado a lado. O resultado final do trabalho pode ser observado nas imagens da Figura 14, onde visualiza-se que os algoritmos chegaram próximos a imagem real.

Em outra aplicação voltada à uma área totalmente diferente, o *framework CycleGAN* foi empregado para a colorização contextual de imagens de baixa luminosidade. Anantrasirichai e Bull (2021) desenvolveram um algoritmo adaptado à base do *CycleGAN*, que melhora a qualidade de imagens macro de alta resolução mesmo em baixa iluminação. Esse problema pode ser parcialmente contornado com o emprego de edições humanas ou com ferramentas de softwares de edição de imagens, mas podem reduzir significativamente a qualidade da imagem alvo. A Figura 15 mostra os resultados desse trabalho com o método aplicado que tornou a imagem de entrada mais nítida e com cores mais naturais.

Figura 15 – Resultados do trabalho de Anantrasirichai e Bull (2021). a) é a imagem de alta resolução, b) a cena em baixa iluminação, c) imagem com a sensibilidade máxima do sensor da câmera, d) a imagem editada por um profissional, e) *CycleGAN* padrão e f) o modelo proposto no trabalho.

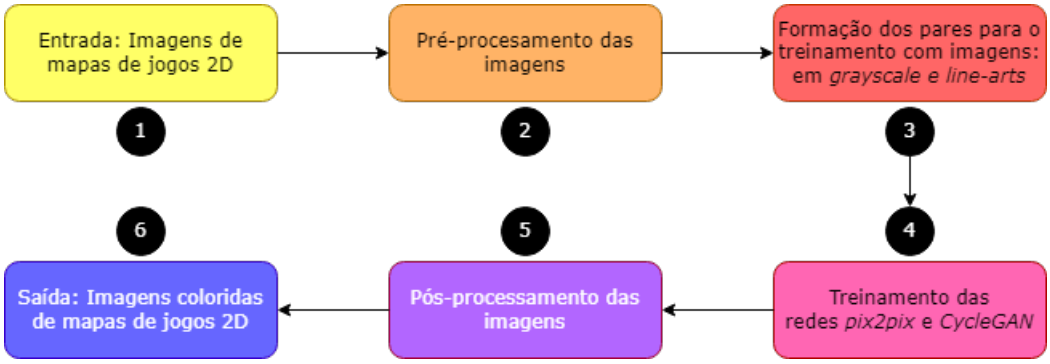


Fonte: Anantrasirichai e Bull (2021).

3 METODOLOGIA

Neste capítulo, é discutido a metodologia utilizada para a colorização de imagens preto e branco e de contornos de mapas de jogos 2D. Durante o trabalho, foram utilizados diferentes métodos de extração de contornos, a fim de compreender qual o mais indicado para o problema em questão. Além disso, foram construídos dois bancos de imagens para os testes. De maneira simplificada, os experimentos seguiram o fluxograma apresentado na Figura 16.

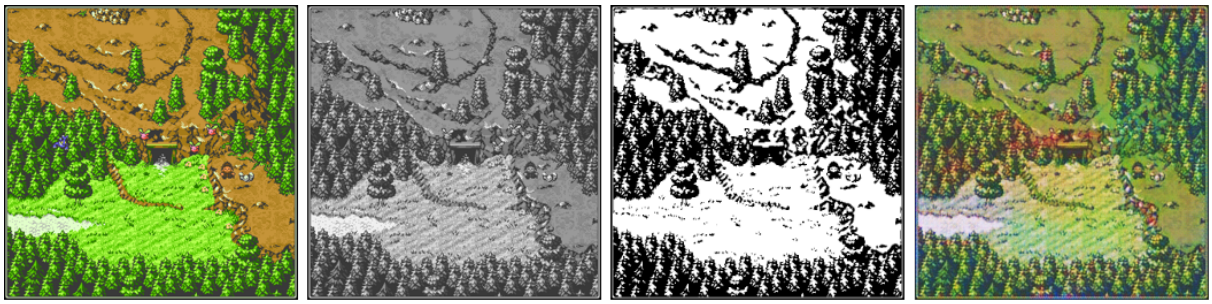
Figura 16 – Fluxo de processos dos experimentos realizados.



Fonte: Autoria própria (2023).

É possível observar o fluxo de uma imagem pelo sistema a partir da Figura 17. Primeiramente foi apresentada a imagem de entrada e o processo de pré-processamento foi utilizado somente em casos que a imagem estivesse fora do padrão do tamanho estipulado para o trabalho. Então a partir da imagem original à esquerda, foi criado um conjunto com imagens em escala de cinza e feito o processo de extração de contornos a partir do método de *thresholding* adaptativo, implementado na biblioteca *OpenCV* (BRADSKI, 2000) em *Python*, formando o conjunto de *line-arts*. A partir dos conjuntos formados, foi realizado o treinamento do *pix2pix* e do *CycleGAN* gerando a imagem da direita e, quando preciso, o pós-processamento das mesmas.

Figura 17 – Exemplificação do fluxograma. Da esquerda à direita: imagem de entrada, imagem em escala de cinza, extração de contornos, resultado da colorização do *pix2pix*.



Fonte: Autoria própria (2023).

3.1 OBTENÇÃO DAS BASES DE IMAGENS

Para a realização dos experimentos foi necessário primeiramente agrupar um conjunto suficiente de imagens de mapas de jogos 2D. Em uma pesquisa inicial, não foi encontrado nada já realizado na literatura relacionado a esse propósito de experimento, portanto, foi necessário coletar diversas imagens que atendessem ao objetivo do trabalho. Na busca pela agilidade em encontrar tais imagens, sites de buscas inteligentes de imagens foram utilizados, como o *Google* Imagens e o *Pinterest*. Esses sites facilitaram esse processo pois ao encontrar uma imagem que satisfazia o problema, foram apresentadas inúmeras imagens semelhantes àquela, processo chamado de busca inteligente.

Como o foco principal do trabalho são imagens de mapas de jogos 2D, foram preferíveis imagens sem personagens, objetos ou efeitos visuais que atrapalhassem a visão da paisagem do mapa e, consequentemente, dificultar o treinamento das redes. Por isso, foram selecionadas, quando possíveis, imagens de paisagens ou caminhos do jogo, e de preferência a uma mesma altura e ângulo uma das outras. Além de mapas de jogos, produções artísticas conhecidas como *pixel arts* também foram incluídas na base de dados. Embora não sejam figuras de jogos propriamente ditos, as *pixel arts* muitas vezes compartilham do mesmo estilo gráfico dos jogos 2D, e por esses motivos elas oferecem um número maior de imagens para o treinamento e teste das redes e consequentemente um melhor resultado final. Foram coletadas imagens para o banco de dados com imagens de mapas e *pixel arts* de diversos estilos de arte. Uma amostra das imagens utilizadas nos experimentos são apresentadas na Figura 18 que, ao final de sua coleta, contabilizou 100 imagens no total.

Figura 18 – Exemplo de imagens utilizadas na primeira base de dados. À esquerda uma *pixel art*, já ao centro e à direita são imagens de diferentes jogos 2D.

Fonte: Autoria própria (2023).

Porém, esse processo depende de encontrar imagens com artes semelhantes entre os jogos e artes e, visto isso, os testes não são idealmente precisos por conta do grande número de escolhas artísticas diferentes. A fim de contornar essa barreira artística entre diferentes jogos e *pixel arts*, foi montada uma segunda base de dados com imagens de um único jogo. O jogo selecionado para a realização deste trabalho foi o *Cendric*, um projeto *open-source* de *Role-playing game* (RPG) disponível em um repositório no *GitHub* (ROESCH, 2018). A escolha

foi baseada em parâmetros criteriosos, tais como a facilidade de localizar imagens do jogo na página do projeto, além da estilização gráfica 2D que se adequa adequadamente à proposta do trabalho. É importante destacar que o mapa do jogo *Cendric* é dividido em quatro imagens de alta resolução, podendo alcançar 1200x1200 pixels, sendo que cada imagem corresponde a uma região específica do jogo. Uma dessas regiões pode ser visualizada na Figura 19. Para assegurar uma quantidade suficiente de imagens destinadas ao treinamento dos *frameworks*, cada uma das quatro imagens originais foi dividida em imagens menores, com aproximadamente 256x256 pixels, conforme apresentado na Figura 20, e ainda menores, com dimensões de 128x128 pixels, conforme exibido na Figura 21. Ao final da criação das duas bases, a de maior resolução totalizou 60 imagens coletadas, e a de menor resolução 297.

Figura 19 – Uma das grandes regiões de *Cendric*.

Fonte: Autoria própria (2023).

3.2 EXTRAÇÃO DE CONTORNOS

Para os experimentos do trabalho, foram conduzidos testes com diferentes métodos de extração de contornos das imagens. Para isso, foram selecionados alguns métodos, que se encaixam com o objetivo final, sendo esses o método *XDoG* (WINNEMöLLER; KYPRIANIDIS; OLSEN, 2012), a função em *Python* disponível na biblioteca *OpenCV* de *thresholding* adaptativo e do método *Canny* (CANNY, 1986), também já implementado nessa biblioteca. Esses métodos foram selecionados por terem uma relevância acadêmica, sendo citados em diversos artigos como (WINNEMöLLER; KYPRIANIDIS; OLSEN, 2012) (CANNY, 1986) (BRADLEY; ROTH, 2007). Em todos os métodos de extração, foi necessário que a imagem fosse convertida

Figura 20 – Parte da região de *Cendric* em resolução 256x256.

Fonte: Autoria própria (2023).

Figura 21 – Parte da região de *Cendric* em resolução 128x128.

Fonte: Autoria própria (2023).

para escala de cinza para gerar melhores resultados, utilizando a função *cvtColor* disponível na biblioteca *OpenCV*.

3.3 MÉTODOS AVALIATIVOS

Para Salimans *et al.* (2016), a avaliação matemática entre diferentes modelos de Redes GAN é uma tarefa difícil, pois ela envolve julgamentos subjetivos sobre o realismo, precisão e

estética das cores resultantes. Porém, apesar de não existirem métodos que simulam a percepção humana, é possível utilizar medidas de erro como MAE, MSE, RMSE e a Distância Euclidiana para avaliar a diferença entre os pixels das imagens geradas pelos algoritmos e a original.

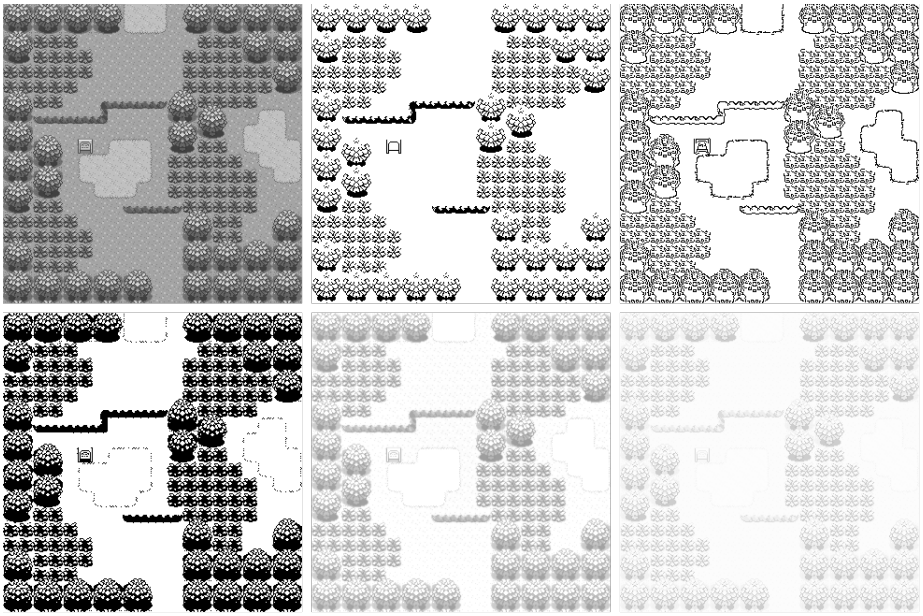
Além dos erros médios, o método da Similaridade por Cosseno e o PSNR auxiliam a entender o quão eficiente foram as imagens geradas, resultando em valores que, quanto mais altos, melhor a geração da imagem. Em complemento aos erros e medidas apresentadas, os desvios padrões de cada um dos resultados foi calculado, fornecendo assim uma representação matemática da consistência da geração de imagens por cada *framework*.

4 CONFIGURAÇÃO EXPERIMENTAL

4.1 EXTRAÇÃO DE CONTORNOS

Experimentos com a extração de contornos foram realizados, verificando as diferentes características de cada método e variando seus parâmetros, que serão discutidos na seção seguinte. Para observar com detalhes os experimentos da retirada dos contornos dos mapas, foi utilizado uma imagem do banco de dados de jogos diversos. A Figura 22 mostra os algoritmos empregados na detecção de bordas com os parâmetros otimizados para melhor visualização.

Figura 22 – Diferentes algoritmos para detecção de contornos. Da esquerda à direita: imagem em escala de cinza original, *threshold* binário, *Canny*, *DoG*, *XDoG*, diferentes parâmetros do *XDoG*.



Fonte: Autoria própria (2023).

Em cada método foi analisado a combinação de parâmetros que trouxesse o melhor resultado visual da extração de contornos. Visando simplificar e obter os melhores resultados possíveis, o método Canny foi utilizado para extrair os contornos das imagens que foram colocadas para treino das redes. Este método consiste em definir 2 limiares, superior e inferior, que definirão se o valor do pixel analisado é um contorno, caso esteja entre os valores ou conectado à um pixel que esteja, ou não, caso esteja acima ou abaixo (CANNY, 1986). Por padrão, os parâmetros são 100 para o limiar inferior e 200 para o superior, porém ao analisar alguns casos de imagens do banco foi notado uma recorrência em imagens mais escuras. Por isso os parâmetros foram alterados para 220 e 250 com o intuito de que detalhes menores das imagens não fossem extraídos, deixando em sua maioria somente as principais informações do ambiente.

4.2 TREINAMENTO, PARÂMETROS E AMBIENTE

O treinamento das redes, pré-processamento das imagens, criação dos bancos para o treinamento e conversão de imagens para escala de cinza foi realizado no ambiente online do Google Colab. O Google Colab é uma plataforma na nuvem que permite aos usuários escrever e executar códigos Python em um ambiente colaborativo. Ele é baseado no Jupyter Notebook e permite que os usuários criem e compartilhem documentos que contêm código, gráficos, equações e texto explicativo. O Colab é gratuito e oferece recursos de computação, como CPUs e GPUs, sem custo adicional. O Colab é amplamente utilizado em tarefas de aprendizado de máquina, análise de dados, ciência de dados, pesquisa acadêmica, e pela sua praticidade foi utilizado nesse projeto.

Após recolhidas as imagens que serviriam de treino para as redes, foi criado um *script* em *Python* que separara aleatoriamente 80% dos dados para treino e 20% para testes. Além disso cria as pastas com essas mesmas imagens em escala de cinza, e no caso do *pix2pix*, por ter um treinamento com conjunto de imagens, estas foram renomeadas para o mesmo nome das imagens originais, tratamento que deve ser realizado seguindo as instruções dos criadores do algoritmo.

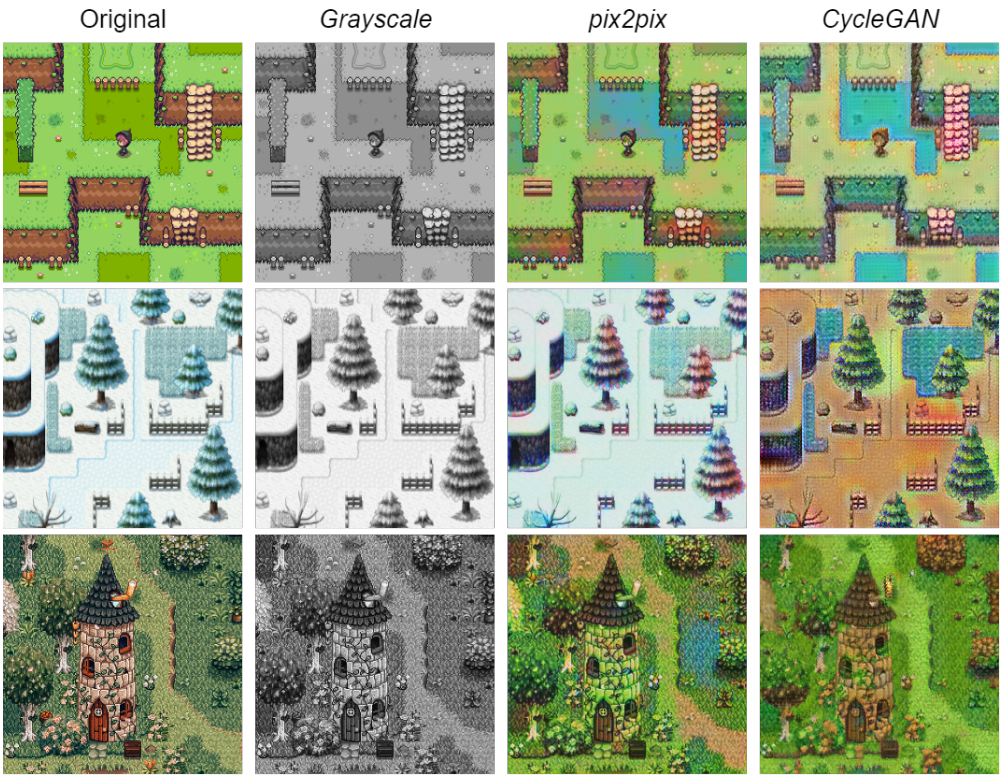
Os parâmetros selecionados para o treinamento dos dois *frameworks* foram os mesmos, sugeridos e disponibilizados pelos próprios autores, dentre eles um treinamento em lote de tamanho 1, ganho inicial de 0,02, taxa de aprendizado inicial de 0,0002, 3 camadas internas para o Discriminador, e 100 épocas totais. Ainda, todas as imagens foram reduzidas ou redimensionadas de suas resoluções iniciais para 256x256, exceto nos testes envolvendo imagens de resolução 128x128, onde essa resolução foi mantida. Foi empregada uma taxa de aprendizado dinâmica, iniciando em 0,0002, que varia de acordo com as mudanças nos valores do gradiente. Essa abordagem permite ajustar a taxa de aprendizado de forma apropriada, evitando ajustes excessivos dos pesos em regiões de superfície de erro com inclinações extremas, garantindo um treinamento eficiente, como explica Yu, Chen e Cheng (1995).

Para a extração dos contornos, foi empregado o algoritmo Canny. A bilioteca *Open CV* disponibiliza em forma de função essa fórmula matemática que, além da imagem, recebe também dois outros parâmetros, um limiar superior e inferior. Valores de pixel acima do limiar máximo são considerados contornos, valores abaixo do limiar mínimo não, e caso esteja entre os dois, é verificado se o pixel em questão está conectado à um pixel contorno, caso sim, é considerado um contorno também.

5 RESULTADOS

Depois da preparação dos dados no pré-processamento, foram realizados experimentos com as imagens da primeira base de dados, composta por mapas de jogos e *pixel arts*, com os *frameworks pix2pix* e *CycleGAN*. A Figura 23 mostra alguns exemplos da aplicação dos algoritmos no banco de dados de imagens diversas, que contou com 100 imagens separadas em 80% para treinamento e 20% para testes. É possível notar que na imagem ao topo, como se trata de um mapa com design mais simples, ambos os algoritmos conseguiram colorir de uma maneira parecida a imagem, cometendo inclusive erros similares, ao pintarem de azul uma parte da grama mais escura. Na imagem da segunda linha, o *pix2pix* conseguiu colorir com mais exatidão o mapa de inverno ao contrário do *CycleGAN* que, embora tenha colorido as árvores de verde, cometeu um erro ao colorir de maneira alaranjada a neve.

Figura 23 – Resultados da colorização de jogos diversos em resolução 256x256.

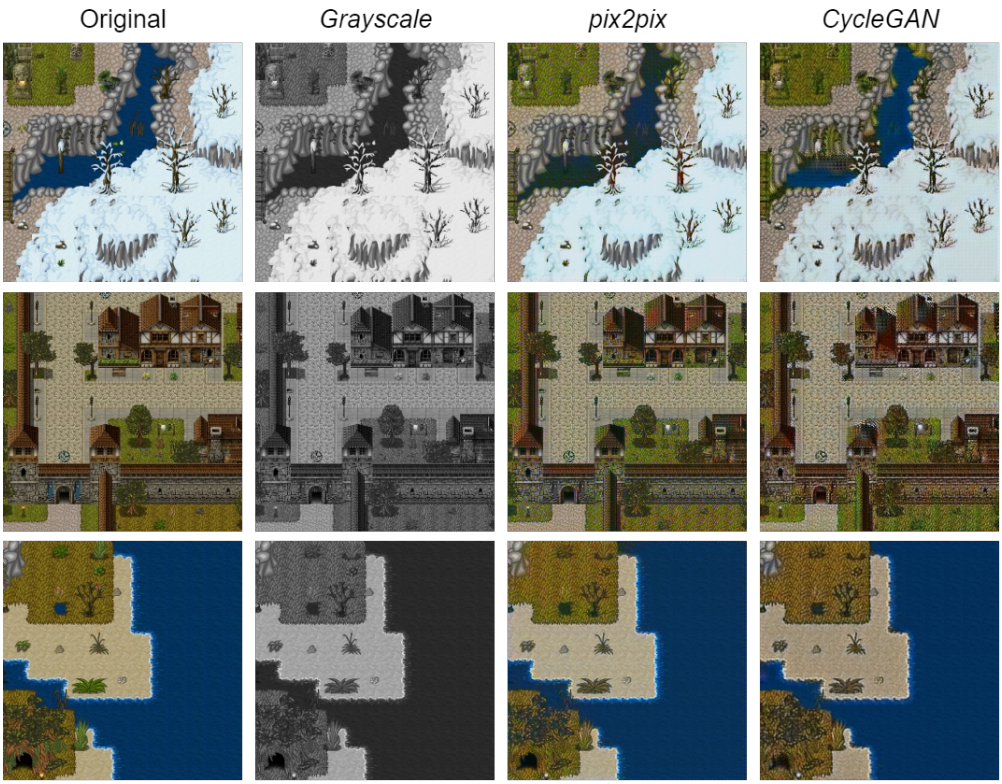


Fonte: Autoria própria (2023).

Já no segundo experimento foi utilizada como entrada somente imagens da segunda base de dados, composta somente por imagens do jogo *Cendric*. Esse experimento foi repetido em duas situações diferentes, com imagens de resolução 256x256 e depois de 128x128, com o objetivo de determinar se imagens com menos detalhes tem menos erro na colorização. No primeiro foram utilizadas 60 imagens e no segundo 297, e a divisão de treinamento e teste foi idêntica ao primeiro experimento, 80% para treino e o restante para testes. Os resultados podem ser vistos na Figura 24. Destaca-se que, por terem um mesmo design de arte, os algoritmos

conseguiram distinguir mais facilmente objetos que anteriormente foram problemáticos como construções, diferentes cenários e contornos de objetos.

Figura 24 – Resultados da colorização do jogo *Cendric* em resolução 256x256.

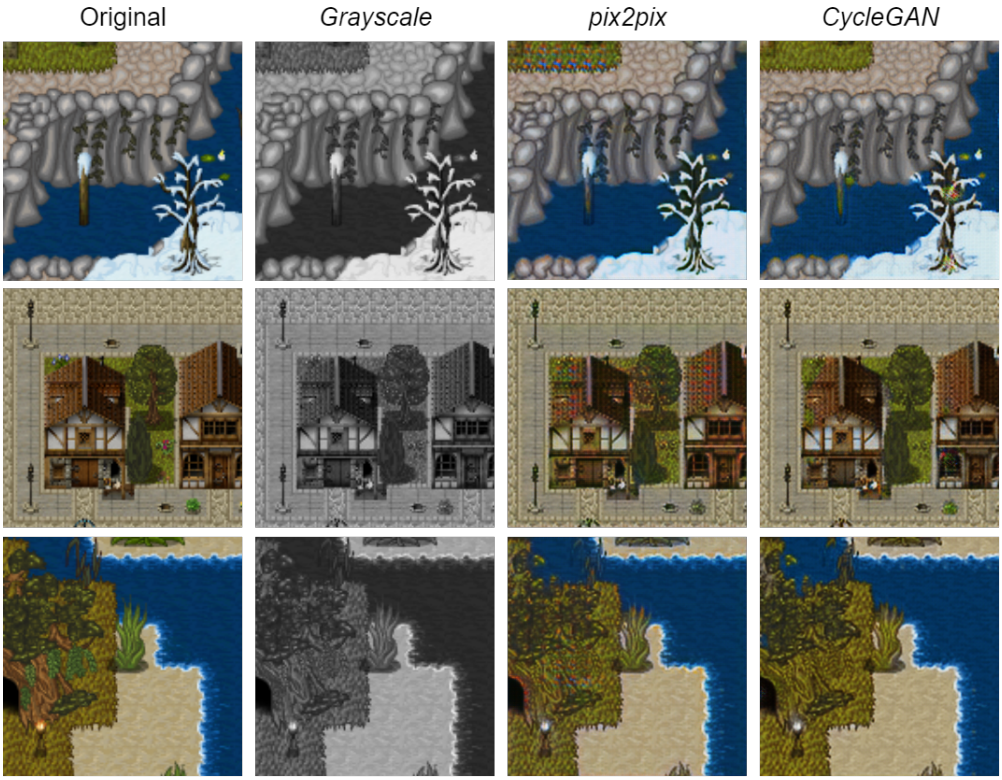


Fonte: Autoria própria (2023).

Por último imagens do jogo *Cendric* em resolução 128x28 foram utilizadas no treinamento. Por conta dos tamanhos das imagens, ambos os algoritmos tiveram suas base de imagens quadruplicadas, ocasionado pela divisão de uma imagem de 256x256 em 4 imagens de 128x128, tanto nos testes como no treinamento. A Figura 25 expõe os resultados desse experimento. Quando comparadas visualmente, à primeira vista, é quase imperceptível observar diferenças entre as imagens originais e as imagens colorizadas pelos algoritmos. Por serem de um jogo do mesmo estilo e de resolução mais baixa, os *frameworks* conseguiram realizar suas tarefas de maneira quase perfeita com mínimos detalhes em conflito quando comparados ao dado real.

Ao mensurar as medidas matemáticas de similaridade entre as imagens reais e as colorizadas pelos algoritmos, é necessário entender antes quais os significados de seus resultados. Para o MAE, MSE e RMSE, que quantificam a diferença entre os valores de intensidade dos pixels de imagens, é esperado que imagens com mais detalhes e características distintas apresentem valores maiores de erro em relação às imagens com menos detalhes e que variações maiores resultem em maiores valores de erro. A Distância Euclidiana também avalia a diferença entre os valores de intensidade dos pixels das imagens. Ademais, imagens com diferentes re-

Figura 25 – Resultados da colorização do jogo *Cendric* em resolução 128x128.



Fonte: Autoria própria (2023).

soluções podem apresentar valores distintos na medida Euclidiana, visto que a distância entre os pixels é calculada diretamente em relação às coordenadas dos pixels na imagem.

As imagens da base de dados com imagens de jogos diversos é a base que tem diferentes estilos gráficos, cores e cenários, por esse motivo, seus erros médios foram superiores aos experimentos com o jogo *Cendric*. Apesar de partirem de uma mesma base, o experimento de colorização de imagens em diferentes resoluções mostrou que, para os erros médios e para a Distância Euclidiana, uma resolução menor teve menos erro, logo, torna a imagem gerada mais parecida com a original. No caso da Similaridade por Cossenos, valores maiores são indicadores de similaridade melhores. Isso ocorre porque a medida de similaridade por cossenos retorna um valor entre -1 e 1, onde 1 significa que as imagens são idênticas e -1 significa que elas são completamente diferentes. Portanto, quanto mais próximo o valor for de 1, maior é a similaridade entre as imagens.

O PSNR, é uma medida de qualidade de imagem que compara a imagem original com a imagem de saída, e um valor maior indica uma maior semelhança entre as duas imagens. As imagens de treinamento com menos detalhes e mais semelhantes entre si obtiveram um resultado superior quando comparado às outras bases que, embora seja uma pequena diferença numericamente, representam superioridade na colorização final. As medidas matemáticas de similaridade entre a imagem original e as colorizadas pelo *pix2xpix* e *CycleGAN*, podem ser observadas na Tabela 1 e 2 respectivamente

Na Tabela 1 com os resultados das medidas de similaridade entre as imagens geradas pelo *pix2pix* nas bases de imagens, as imagens *Cendric* de resolução menores, como observado nos dados apresentados experimento anterior, tiveram a colorização mais fiel, o que se reflete nos números das médias das medidas das imagens. De acordo com os resultados da tabela, os conjuntos de dados *Cendric* 256x256 e 128x128 apresentaram os melhores resultados nas métricas de média e similaridade, em comparação com o conjunto de dados com imagens diversas. Especificamente, *Cendric* de resolução 128x128 obteve os melhores resultados em diversas métricas, como MAE, MSE, RMSE, distância euclidiana, similaridade por cosseno e PSNR, indicando um desempenho superior em termos de precisão da colorização e qualidade das imagens geradas. Esses resultados reforçam a vantagem da utilização de imagens de resolução menor na tarefa de colorização com o modelo *pix2pix*.

Tabela 1 – Tabela com o resultado das medidas de similaridade do experimento de colorização de imagens em escala de cinza com o *framework pix2pix*.

		Datasets		
		mixed_maps	***Cendric_256x256***	***_128x128***
Média	**MAE**	0,474	0,091	**0,073**
	MSE	98,070	48,740	**38,269**
	RMSE	9,903	6,981	**6,186**
	Distância Euclidiana	4387,125	3089,9622	**2728,628**
	Similaridade por Cosseno	1,043e-07	4,518e-08	**8,498e-08**
	PSNR	28,231	31,283	**32,396**
		*valores em **negrito** representam os melhores resultados		

Fonte: Autoria própria (2023).

Ao analisar as métricas da Tabela 2, é possível observar diferenças entre os conjuntos. Por exemplo, o conjunto de dados de mapas diversos apresenta valores mais baixos para métricas como MAE e MSE, indicando uma menor taxa de erro médio e erro quadrático médio. Por outro lado, o conjunto de dados de resolução 128x128 se destaca por sua menor dispersão de erros, conforme evidenciado pelo valor reduzido de RMSE. Além disso, esse conjunto exibe uma maior similaridade por cosseno média, indicando uma maior semelhança entre os vetores de características. Adicionalmente, também se sobressai no quesito qualidade de imagem, demonstrando um valor mais elevado de PSNR.

Observa-se que apesar da base *Cendric* 128x128 possuir os melhores resultados de similaridade, possui os desvios padrões mais elevados em comparação com os outros conjuntos, indicando maior variabilidade e inconsistência nos resultados das métricas. Os dados de desvios padrões para a colorização de imagens em escala de cinza no *pix2xpi* e no *CycleGAN* são exibidos na Tabela 3 e Tabela 4.

Os resultados de colorização de mapas de jogos utilizando somente os contornos extraídos pelo método Canny são dispostos abaixo. A Figura 26 exibe uma amostra de resultados tanto da extração de contornos quanto do *framework pix2pix* e sua tentativa de colorir a ima-

Tabela 2 – Tabela com o resultado das medidas de similaridade do experimento de colorização de imagens em escala de cinza com o *framework CycleGAN*.

		Datasets		
		mixed_maps	***_256x256***	***_128x128***
Média	**MAE**	0,443	**0,196**	0,373
	MSE	102,513	79,312	**74,655**
	RMSE	10,125	8,906	**8,640**
	Distância Euclidiana	4488,260	3936,122	**3740,599**
	Similaridade Cosseno	5,887e-08	6,134-08	**7,035e-08**
	PSNR	28,027	29,192	**29,854**
		*valores em **negrito** representam os melhores resultados		

Fonte: Autoria própria (2023).

Tabela 3 – Tabela com o desvio padrão das medidas de similaridade do experimento de colorização de imagens em escala de cinza com o *framework pix2pix*.

		Datasets		
		mixed_maps	***_256x256***	***_128x128***
Desvio Padrão	***MAE***	0,286	**0,013**	**0,013**
	MSE	8,180	**6,155**	7,662
	RMSE	2,860	**2,481**	2,768
	Distância Euclidiana	**185,530**	186,811	280,479
	Similaridade Cosseno	**1,072e-07**	2,627e-08	6,869e-08
	PSNR	**0,372**	0,504	0,922
		*valores em **negrito** representam os melhores resultados		

Fonte: Autoria própria (2023).

Tabela 4 – Tabela com o desvio padrão das medidas de similaridade do experimento de colorização de imagens em escala de cinza com o *framework CylceGAN*.

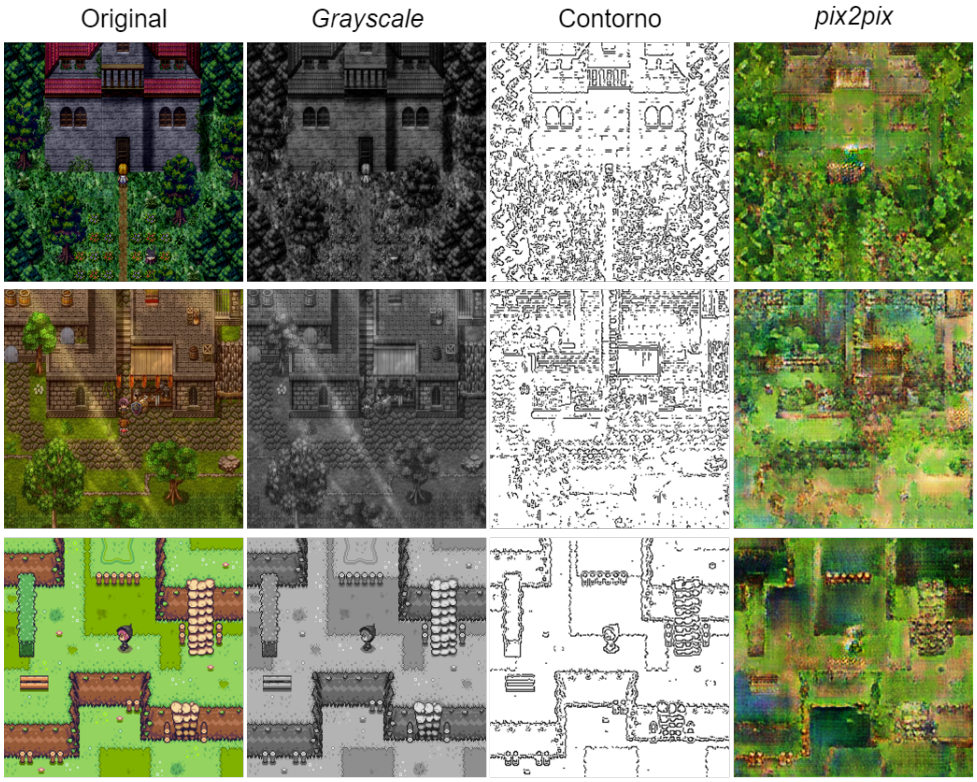
		Datasets		
		mixed_maps	***_256x256***	***_128x128***
Desvio Padrão	***MAE***	**0,124**	0,128	**0,293**
	MSE	**4,667**	13,384	29,939
	RMSE	**2,160**	3,658	5,472
	Distância Euclidiana	**102,004**	316,836	828,064
	Similaridade Cosseno	4,710e-08	**3,289e-08**	5,412e-08
	PSNR	**0,197**	0,667	2,152
		*valores em **negrito** representam os melhores resultados		

Fonte: Autoria própria (2023).

gem. Por ser um banco com imagens muito diferentes, diversos detalhes, texturas, iluminação, sombras e até mesmo reflexos, o algoritmo tem dificuldade para definir contornos que, à olho humano, são mais simples de observar. Outra dificuldade encontrada foi definir o valor de limiar superior e inferior do método Canny, visto que valores mais baixos levariam a mais pixels serem

considerados contornos e imagens compostas de diversos elementos teriam a extração mais confusa e resultados "poluídos". Por outro lado, valores mais altos desse limiar, deixavam as imagens mais simples praticamente sem contornos. No que diz respeito à colorização dos contornos, as cores do ambiente e elementos ficaram confusos nesse teste, puxando muito mais para um tom esverdado na maior parte das imagens utilizadas como teste. Isso se deve ao fato de haverem muito mais imagens com padrões de pintura verdes do que de outros tons e da dificuldade para se obter um contorno mais exato das imagens tanto de treino como de teste.

Figura 26 – Resultados da colorização de contornos de jogos diversos em resolução 256x256.



Fonte: Autoria própria (2023).

Já os resultados do jogo *Cendric*, tanto na resolução de 256x256 (Figura 27) quanto de 128x128 (Figura 28), os resultados seguiram uma mesma linha e ficaram muito próximos visualmente, não sendo notado uma diferença expressiva na extração de contornos ou na colorização. Os contornos gerados pelo método Canny nessas bases de dados ficaram mais próximos ao real, e por isso, a colorização das imagens ficaram notadamente melhores que as do teste anterior com mapas diversos. Porém, embora melhores, ficaram longe de perfeitos, sendo observado diversos focos de desordem nas cores dos pixels, devido à recortes mal executados de contorno.

Na Tabela 5 fica evidente a distância dos resultados gerados com as imagens originais. Para efeito comparativo, ao observar na Tabela 1 com as medidas de similaridade do *pix2pix* na colorização de imagens preto e branco, o melhor resultado do MAE médio foi de 0,073. Já no experimento de colorização de contornos, foi de 0,439, um aumento significativo no erro. Esse

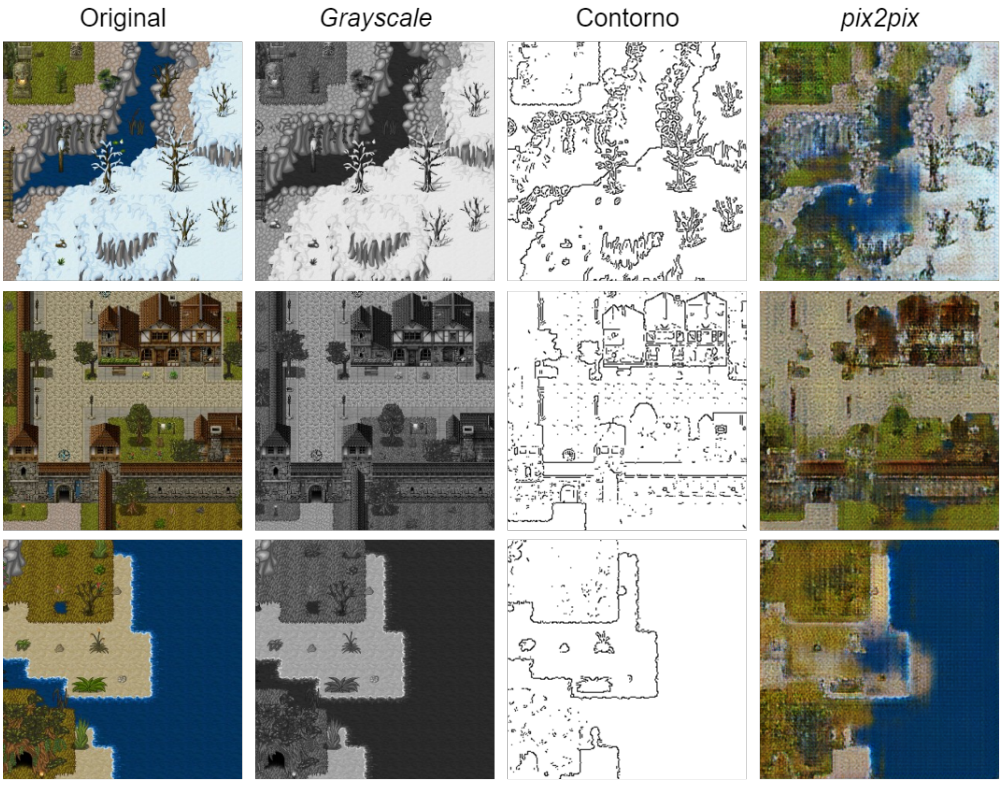
Tabela 5 – Tabela com o resultado das medidas de similaridade do experimento de colorização de contornos com o *framework pix2pix*.

		Datasets		
		mixed_maps	***Cendric_256x256***	***Cendric_128x128***
Média	**MAE**	0,698	0,475	**0,439**
	MSE	105,402	97,715	**93,713**
	RMSE	10,267	9,885	**9,681**
	Distância Euclidiana	4551,961	4378,578	**4247,505**
	Similaridade Cosseno	8,377e-08	**5,801e-08**	8,815e-08
	PSNR	27,903	28,250	**28,652**
		*valores em negrito representam os melhores resultados		

Fonte: Autoria própria (2023).

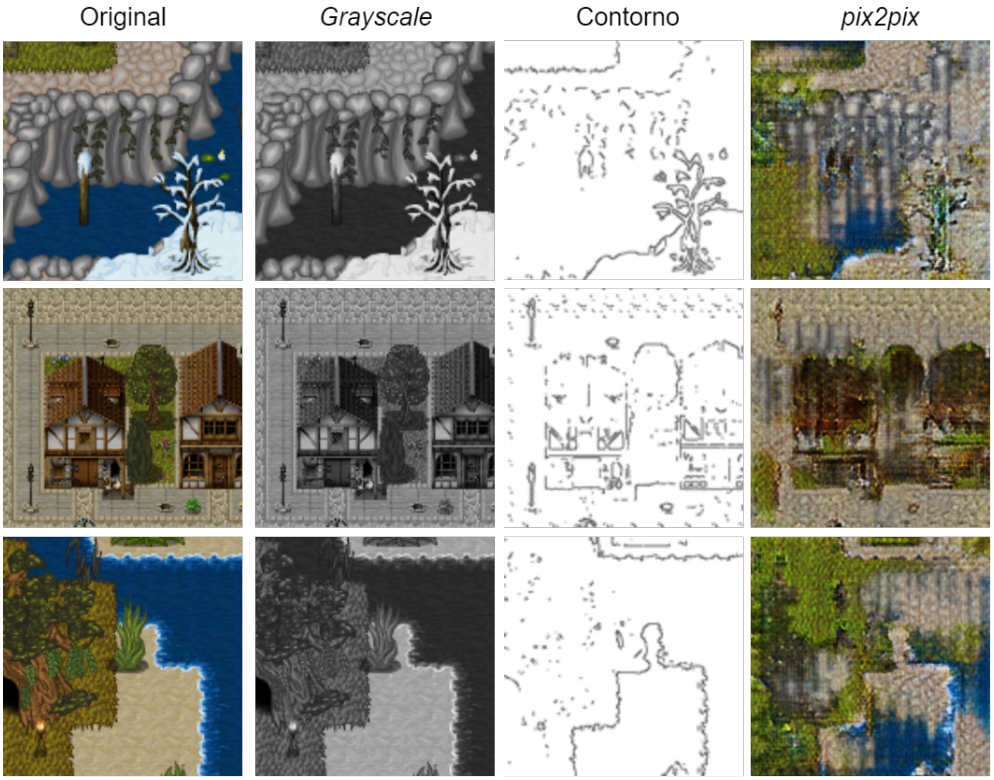
padrão de diminuição na qualidade da imagem gerada seguiu-se por todas as outras medidas, mostrando a dificuldade da colorização nesse cenário.

Figura 27 – Resultados da colorização de contornos do jogo em resolução 256x256.



Fonte: Autoria própria (2023).

Figura 28 – Resultados da colorização de contornos do jogo em resolução 128x128.



Fonte: Autoria própria (2023).

6 CONCLUSÃO

Com os resultados dos experimentos realizados com os *frameworks pix2pix* e *CycleGAN* nas imagens coletadas, ficou evidente que a resolução das imagens e a natureza dos conjuntos de dados desempenharam um papel crucial. No teste de colorização em escala de cinza do *pix2pix* por exemplo, o aumento de 0,073 para 0,091 no MAE entre as imagens Cendric de baixa e alta resolução, representou uma diferença de aproximadamente 25% no erro, e comparado ao banco do primeiro experimento foi ainda maior, aproximadamente 500%. Esse efeito se repetiu nas outras medidas de erro, Distância, Similaridade por Cosseno e PSNR que, por sua vez, teve a menor das diferenças nesse teste, com uma melhora de 14% na similaridade calculada a partir dessa medida, da menor 28,231 (mapas diversos) para a maior 32,396 (Cendric 128x128), vale lembrar que para o PSNR quanto maior o valor, mais similar a imagem é. Esses valores podem ser encontrados na Tabela 1, discorridos no capítulo anterior.

Para a colorização preto e branco com o *CycleGAN*, apesar de menores variações, não foi diferente. As medidas registradas na Tabela 2 mostram que mesmo com a troca de *framework*, a resolução menor e a similaridade visual nas imagens de treino tiverem um impacto positivo nos testes, e no caso da Distância Euclidiana desse teste, a menor medida de 3740 do banco Cendric 128x128 e a maior medida de 4488 dos mapas diversos, teve um salto de 20%. A diferença de resoluções não teve um impacto tão grande nas medidas de ambos os *frameworks* e, apesar de erros menores, não tiveram uma alteração grande nos resultados finais. No RMSE do teste com o *CycleGAN* por exemplo, do banco de dados Cendric de resolução 128, teve uma diferença de apenas 2% comparado ao de 256.

No entanto, apesar das medidas de erros menores na colorização usando o conjunto de dados com imagens de um único jogo em escala de cinza realizada nos experimentos, foi observado que ainda existem desafios e imperfeições na tarefa de colorização, principalmente quando à extração de contornos. Durante o processo, alguns problemas surgiram, como a ocorrência de imperfeições na definição dos contornos, o que afetou diretamente a colorização das imagens. Esses erros resultaram em áreas coloridas de forma inadequada e em desordens nas cores dos pixels. Uma solução possível para contornar esse problema, seria explorar métodos mais robustos para a extração de contornos, como as DCGAN, a fim de obter resultados mais precisos e evitar distorções na etapa de colorização. Apesar disso, em um cenário realista, esses contornos seriam desenhados pelo próprio artista do mapa, e não por um algoritmo, sendo assim eles ficariam mais fiéis ao mapa final e contornariam a dificuldade do algoritmo de extrair os contornos.

Outro aspecto a ser considerado é a diversidade dos conjuntos de dados. Notou-se que o conjunto de dados com imagens diversas, que apresentava uma variedade de estilos gráficos, cores e cenários, apresentou erros médios maiores em comparação aos experimentos realizados com o jogo Cendric em ambas as resoluções. Portanto, em experimentos futuros, seria prudente considerar a criação de conjuntos de dados mais coesos em termos de estilo

visual, a fim de obter resultados com erros menores que os obtidos. Um ponto a ser considerado é a otimização dos parâmetros dos modelos como: aumentar o número de épocas, utilizar um *dataset* para a validação durante o treino, utilizar mais camadas internas, para assim melhorar a qualidade das imagens geradas, diminuindo seus erros médios e da Distância Euclidiana, e aumentando os valores da Similaridade por Cosseno e do PSNR em todos os testes.

Na prática, quando comparados os resultados das medidas de similaridade da colorização em escala de cinza e de contornos, observa-se uma diferença significativa na taxa de erro do *pix2pix*. Comparando a Tabela 1 e Tabela 5, as imagens coloridas a partir de preto e branco tiveram seus erros médios menores, no melhor caso do MAE médio um aumento de 500%, e apesar da diferença pequena no PSNR e na Similaridade por Cosseno, teve seus valores maiores. Esses dados mostram que ainda há muito a melhorar na extração de contornos e, como citado anteriormente, além de mudanças no treinamento da rede *pix2pix*, variações de *treshold* mínimo e máximo no método Canny podem alterar os resultados finais.

Contudo, ao questionar alguns artistas acostumados a desenharem *line-arts* e a colorirem imagens sobre o tempo estimado de produção de cada uma dessas tarefas, embora com tempos diferentes, a resposta foi unânime. Todos responderam que o tempo destinado à colorização é muito maior, chegando até o triplo, que o de fazer somente os contornos, que tem uma média de 1 à 3 horas segundo os artistas. Ainda quando questionados sobre a duração de uma colorização em preto e branco ao invés de com cores, a resposta foi semelhante, com projetos que ultrapassavam 8 horas de desenvolvimento em escala de cinza e até dias para colorir uma imagem. Muitos deles pontuaram que mesmo as *line-arts* podem ser muito complexas de se criarem, principalmente aquelas com efeitos de sombreamento e movimento, porém colorir demanda muito mais trabalho visto que deve-se prestar atenção na ambientação, temática e visual esperado da arte, o que reitera a ideia de que uma automação eficaz para esse processo seria de grande ajuda.

Em conclusão, os resultados dos experimentos de colorização de mapas de jogos 2D, forneceram uma visão inicial do potencial e das limitações dos *frameworks* estudados nesse trabalho. Através de abordagens como otimização de parâmetros, diversificação do conjunto de treinamento, pré-processamento aprimorado e uso de arquiteturas de redes mais avançadas, é possível aperfeiçoar a qualidade e a precisão da colorização das imagens, diminuindo os erros médios e seus desvios padrões. Este trabalho desenvolveu um método inicial de colorização automática de mapas de jogos 2D a partir de imagens em escala de cinza e de contornos. Através a obtenção de bases de imagens que foram utilizadas para o treinamento e teste nos *frameworks pix2pix* e *CycleGAN*, foi possível comprovar que um treinamento em resolução mais baixa e a utilização de imagens com mesmo estilo gráfico são fatores diferenciais matematicamente e visualmente na tarefa de colorização.

REFERÊNCIAS

ALOTAIBI, A. Deep generative adversarial networks for image-to-image translation: A review. **Symmetry**, v. 12, n. 10, 2020. ISSN 2073-8994. Disponível em: https://www.mdpi.com/2073-8994/12/10/1705.

ANANTRASIRICHAI, N.; BULL, D. Contextual colorization and denoising for low-light ultra high resolution sequences. *In*: . [*S.l.*: *s.n.*], 2021. p. 1614–1618.

ATTAIKI, S. Manga colorization using generative adversarial nets. 02 2019.

BRADLEY, D.; ROTH, G. Adaptive thresholding using the integral image. **J. Graphics Tools**, v. 12, p. 13–21, 01 2007.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000.

BRUNIALTI, L. *et al.* Aprendizado de máquina em sistemas de recomendação baseados em conteúdo textual: Uma revisão sistemática. *In*: **Anais do XI Simpósio Brasileiro de Sistemas de Informação**. [*s.n.*], 2015. p. 203–210. ISSN 0000-0000. Disponível em: https://sol.sbc.org.br/index.php/sbsi/article/view/5818.

CANNY, J. A computational approach to edge detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PAMI-8, n. 6, p. 679–698, 1986.

CRESWELL, A. *et al.* Generative adversarial networks: An overview. **IEEE Signal Processing Magazine**, v. 35, p. 53–65, 2018.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. 1989.

GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric Environment**, v. 32, n. 14, p. 2627–2636, 1998. ISSN 1352-2310. Disponível em: https://www.sciencedirect.com/science/article/pii/S1352231097004470.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S.l.*: *s.n.*], 2016. http://www.deeplearningbook.org.

GOODFELLOW, I. J. *et al.* **Generative Adversarial Networks**. 2014. Disponível em: https://arxiv.org/abs/1406.2661.

HAO, X.; ZHANG, G.; MA, S. Deep learning. **International Journal of Semantic Computing**, v. 10, p. 417–439, 09 2016.

HATI, Y. *et al.* Paintstorch: a user-guided anime line art colorization tool with double generator conditional adversarial network. *In*: **European Conference on Visual Media Production (CVMP)**. [*s.n.*], 2019. p. 1–10. Disponível em: https://hal.archives-ouvertes.fr/hal-02455373.

HSU, K.-Y.; LI, H.-Y.; PSALTIS, D. Holographic implementation of a fully connected neural network. **Proceedings of the IEEE**, v. 78, n. 10, p. 1637–1645, 1990.

ISOLA, P. *et al.* Image-to-image translation with conditional adversarial networks. *In*: **2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S.l.*: *s.n.*], 2017. p. 5967–5976.

JANIESCH, C.; ZSCHEC, P.; HEINRICH, K. Machine learning and deep learning. 2021.

JI, L.; GALLO, K. An agreement coefficient for image comparison. **Photogrammetric Engineering and Remote Sensing.**, v. 73, p. 823–833, 07 2006.

KANAL, L. N. Perceptron. *In*: ____. **Encyclopedia of Computer Science**. [*S.l.*: *s.n.*], 2003. p. 1383–1385. ISBN 0470864125.

KELLER, J. M.; LIU, D.; FOGEL, D. B. Introduction and single-layer neural networks. *In*: ____. **Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation**. [*S.l.*: *s.n.*], 2016. p. 5–34.

KUMAR, M.; WEISSENBORN, D.; KALCHBRENNER, N. **Colorization Transformer**. 2021.

LECUN BENGIO, H. Deep learning. **Nature**, p. 436–444, 2015.

LIU, M. *et al.* Towards better analysis of deep convolutional neural networks. **IEEE Transactions on Visualization and Computer Graphics**, v. 23, n. 1, p. 91–100, 2017.

LUGRIS, M. **New ESA Report Shows Gaming Is No Longer A Niche Market**. 2020. Disponível em: https://www.thegamer.com/esa-gaming-niche-popular-die-mad-gamers/.

LUO, V.; STRAKA, M. Historical and modern image-to-image translation with generative adversarial networks. *In*: . [*S.l.*: *s.n.*], 2017.

MAHESH, B. **Machine Learning Algorithms -A Review**. 2019.

MARTIN, D.; FOWLKES, C.; MALIK, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 26, n. 5, p. 530–549, 2004.

MING, Y.; LI, H.; HE, X. Contour completion without region segmentation. **IEEE Transactions on Image Processing**, v. 25, n. 8, p. 3597–3611, 2016.

MIRZA, M.; OSINDERO, S. **Conditional Generative Adversarial Nets**. 2014. Disponível em: https://arxiv.org/abs/1411.1784.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *In*: **Sistemas Inteligentes Fundamentos e Aplicações**. 1. ed. [*S.l.*: *s.n.*], 2003. p. 89–114.

NEWZOO. **Global Games Market Report**. 2022. Disponível em: https://newzoo.com/products/reports/global-games-market-report.

NIXON, M.; AGUADO, A. **Feature Extraction and Image Processing for Computer Vision**. [*s.n.*], 2012. ISBN 9780123965493. Disponível em: https://books.google.com.br/books?id=lytnomY-r7YC.

PAPARI, G.; PETKOV, N. Edge and line oriented contour detection: State of the art. **Image and Vision Computing**, v. 29, n. 2, p. 79–103, 2011.

RADFORD, A.; METZ, L.; CHINTALA, S. **Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks**. 2015. Disponível em: https://arxiv.org/abs/1511.06434.

RAMCHOUN, H. *et al.* Multilayer perceptron: Architecture optimization and training. v. 4, p. 26–30, 2016.

RODRIGUES, E. O.; CLUA, E.; VITOR, G. B. Line art colorization of fakemon using generative adversarial neural networks. *In*: **2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)**. [*S.l.*: *s.n.*], 2022. p. 1–6.

ROESCH, T. Z. I. **Cendric2**. 2018. Disponível em: https://github.com/tizian/Cendric2.

RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. 2015. Disponível em: https://arxiv.org/abs/1505.04597.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. 1958.

RUSK. Deep learning. p. 35, 2016.

SAHARIA, C. *et al.* **Palette: Image-to-Image Diffusion Models**. 2021. Disponível em: https://arxiv.org/abs/2111.05826.

SALIMANS, T. *et al.* Improved techniques for training gans. *In*: **Advances in Neural Information Processing Systems**. [*S.l.*: *s.n.*], 2016.

SANTINI, S.; JAIN, R. Similarity measures. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 21, n. 9, p. 871–883, 1999.

SHINDE, P. P.; SHAH, S. A review of machine learning and deep learning applications. *In*: **2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)**. [*S.l.*: *s.n.*], 2018. p. 1–6.

SILVA, E. A.; PANETTA, K.; AGAIAN, S. S. Quantifying image similarity using measure of enhancement by entropy. *In*: AGAIAN, S. S.; JASSIM, S. A. (Ed.). **Mobile Multimedia/Image Processing for Military and Security Applications 2007**. [*s.n.*], 2007. v. 6579, p. 65790U. Disponível em: https://doi.org/10.1117/12.720087.

SILVA, F. C. *et al.* Mangan: Assisting colorization of manga characters concept art using conditional gan. *In*: **2019 IEEE International Conference on Image Processing (ICIP)**. [*S.l.*: *s.n.*], 2019. p. 3257–3261.

WANG, K. *et al.* Generative adversarial networks: introduction and outlook. **IEEE/CAA Journal of Automatica Sinica**, v. 4, n. 4, p. 588–598, 2017.

WANG, S.-C. Artificial neural network. *In*: ____. **Interdisciplinary Computing in Java Programming**. [*S.l.*: *s.n.*], 2003. p. 81–100. ISBN 978-1-4615-0377-4.

WEN CHUAN LIN, F. L. Z.; CUI, L. Information recombination network for contour detection. **International Journal of Computer Assisted Radiology and Surgery**, 2022.

WINNEMöLLER, H.; KYPRIANIDIS, J. E.; OLSEN, S. C. Xdog: An extended difference-of-gaussians compendium including advanced image stylization. **Computers and Graphics**, v. 36, n. 6, p. 740–753, 2012. Disponível em: https://www.sciencedirect.com/science/article/pii/S009784931200043X.

YU, X.-H.; CHEN, G.-A.; CHENG, S.-X. Dynamic learning rate optimization of the backpropagation algorithm. **IEEE Transactions on Neural Networks**, v. 6, n. 3, p. 669–677, 1995.

ZHU, J.-Y. *et al.* **Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks**. 2017. Disponível em: https://arxiv.org/abs/1703.10593.