UNIVERSIDADE FEEVALE

WILLIAN MENDES DE OLIVEIRA

GERANDO MODELOS DE CALÇADOS USANDO TÉCNICAS DE CRIATIVIDADE COMPUTACIONAL

Novo Hamburgo

2020

WILLIAN MENDES DE OLIVEIRA

GERANDO MODELOS DE CALÇADOS USANDO TÉCNICAS DE CRIATIVIDADE COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

Orientador: João Batista Mossmann

Novo Hamburgo

2020

WILLIAN MENDES DE OLIVEIRA

GERANDO MODELOS DE CALÇADOS USANDO TÉCNICAS DE CRIATIVIDADE COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

APROVADO EM: ___ / ___ / _____

JOÃO BATISTA MOSSMANN
Orientador – Feevale

MARTA ROSECLER BEZ
Examinador interno – Feevale

PAULO RICARDO MUNIZ BARROS
Examinador interno – Feevale

Novo Hamburgo
2020

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial: Ao meu orientador, João Batista Mossmann, que me deu todo apoio para a realização deste projeto, aos amigos e à minha família, que convivem comigo diariamente, assim como à minha namorada, minha gratidão, pelo apoio emocional nos períodos mais difíceis do trabalho.

RESUMO

Criatividade computacional é uma área da computação, que se utiliza de técnicas de Inteligência Artificial, na qual se constrói e trabalha com sistemas computacionais que criam artefatos. Usando técnicas de criatividade computacional, muitos pesquisadores foram capazes de desenvolver ferramentas com capacidade de gerar imagens bidimensionais de objetos artificiais baseadas em objetos existentes resultantes de processos criativos. Tendo isto em vista, esta pesquisa buscará aplicar esses conceitos no campo do *design* de calçado, justificando-se pela relevância da região do Vale dos Sinos na indústria coureiro-calçadista, reconhecida como referência nacional. Para que este objetivo fosse atingido, foi definido um processo para a geração de imagens bidimensionais de calçados, o qual foi posteriormente executado. Por fim, os resultados gerados foram analisados quantitativamente e qualitativamente, sendo que através da métrica FID notou-se um aumento da qualidade e diversidade das amostras geradas durante o treinamento.

Palavras-chave: Criatividade Computacional, Geração de Calçados, Redes Neurais.

ABSTRACT

Creative computing is a field of computer science, that uses Artificial Intelligence techniques, where computational system that create artifacts are built. Using Creative Computing techniques, many researches were able of developing tools which were capable of generating two-dimensional images of artificial objects based on real objects resulting from creative processes. This research will seek to apply those concepts in the shoe design field, justified by the relevance of the Vale dos Sinos region in shoe manufacturing, a region that is reference on this field in Brazil. For this goal to be met, a process was defined to generate images of shoes and was later executed. Finally, the generated results were analyzed quantitatively and qualitatively, in which by the use of the FID metric it was noted an increase in quality and diversity of the samples generated during the training.

Keywords: Creative Computing, Shoes Generation, Neural Networks.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CIFAR	*Canadian Institute For Advanced Research*
FID	*Fréchet Inception Distance*
GAN	*Generative Adversarial Network*
GPU	*Graphics processing unit*
IP	*Internet Protocol*
LSTM	*Long Short-Term Memory*
MNIST	*Modified National Institute of Standards and Technology*
MLP	*Multilayer Perceptron*
NN	*Neural Network*
TFD	*Toronto Faces Dataset*

SUMÁRIO

1 INTRODUÇÃO

Criatividade computacional é uma área da computação que se utiliza de técnicas de Inteligência Artificial, trabalhando e desenvolvendo sistemas computacionais que criam artefatos. Estes sistemas são geralmente aplicados em domínios que se associam a pessoas com formação na área criativa, tais como: poesia, composição musical, jogos digitais, arquitetura, design gráfico e até culinária (COLTON; WIGGINS et al., 2012).

Para Toivonen e Gross (2015), a criatividade computacional é caracterizada paralelamente à inteligência artificial: Ao passo que a inteligência artificial se preocupa em como desempenhar tarefas que seriam consideradas inteligentes se fossem executadas por seres humanos, a criatividade computacional estuda performances que seriam consideradas criativas se fossem executadas por humanos. Na esteira das ideias destes autores, o objetivo deste campo é modelar, simular, ou aperfeiçoar a criatividade humana usando métodos computacionais (TOIVONEN; GROSS, 2015).

Usando técnicas de criatividade computacional, muitos estudos obtiveram êxito ao criar conteúdo novo usando como base artefatos já existentes, como o estudo de Loller-Andersen e Gambäck (2018) no artigo *"Deep Learning-based Poetry Generation Given Visual Input"*. Este artigo, descreve a implementação e avaliação de um sistema capaz de gerar poesia satisfazendo critérios rítmicos e de rima dada a entrada de uma imagem. Tal geração é feita através de uma Rede Neural Convolucional para a classificação de objetos na imagem, um módulo para achar palavras relacionadas e palavras que rimam, e uma rede de memória de longo prazo (LSTM, *Long Short-Term Memory*) treinada com um conjunto de dados de letras de músicas compilado para esse trabalho.

Ainda relacionado ao trabalho de Loller-Andersen e Gambäck (2018), foram geradas e avaliadas 153 estrofes em dois experimentos diferentes. Os resultados indicam que o sistema baseado em *Deep Learning* é capaz de gerar poesias subjetivamente poéticas, gramaticalmente corretas e com significado, mas, segundo a avaliação dos autores, ainda não é consistente.

Um grande avanço na área da criatividade computacional aconteceu com a publicação do artigo *"Generative Adversarial Nets"* por Goodfellow et al. (2014), onde é proposto um *framework* para estimativa de modelos generativos através do uso de Redes Neurais Artificiais. Experimentos demonstraram um potencial alto do *framework* através da avaliação qualitativa e quantitativa das amostras geradas.

Ainda relacionado ao trabalho de Goodfellow et al. (2014), na Figura 1 estão amostras de imagens geradas pelos modelos treinados para quatro *datasets* diferentes, sendo estes: (a) um compilado de números escritos à mão chamado MNIST, (b) um conjunto de imagens do rostos chamado TFD, (c, d) um *dataset* de pequenas imagens rotuladas chamado CIFAR-10. Em cada uma das imagens, a coluna à extrema direita contém a imagem do *dataset* usada no treinamento que é mais próxima da amostra vizinha, para evidenciar que a rede não está copiando os resultados dos dados de treinamento, enquanto as outras colunas contêm imagens geradas pelo modelo (GOODFELLOW et al., 2014).

Figura 1: Exemplos de imagens geradas pela *Generative Adversarial Network*

Fonte: Goodfellow et al. (2014)

Baseado no trabalho de Goodfellow et al. (2014), um grupo de pesquisadores da empresa Nvidia Corporation, divulgou um artigo que apresenta uma nova metodologia para o treinamento de Redes Adversariais Generativas, onde ao aumentar gradativamente a resolução das imagens de treinamento, foram capazes de reduzir o tempo de treinamento e aumentar a qualidade das imagens geradas. Essa nova técnica foi aplicada para geração de imagens de rostos humanos baseadas num *dataset* de artistas e também na geração de imagens de quartos. O resultado desta geração pode ser visto na Figura 2, que contém alguns exemplos de imagens de quartos geradas pelo modelo (KARRAS et al., 2017).

Também na área da criatividade computacional, o trabalho "Geração automatizada de imagens em *pixel art* utilizando redes neurais" pode ser citado. Neste estudo, o autor usa Redes Neurais Artificiais para a geração de poses de personagens em *pixel art* a partir de imagens prévias e do desenho da pose do personagem. Após treinar e avaliar as imagens geradas, o autor conclui que a técnica pode ser aplicada no contexto do desenvolvimento de software para facilitar e incrementar a produção artística de jogos computacionais em que se empregam esse tipo de arte digital (CAMARGO, 2019).

Figura 2: Imagens dos quartos gerados

Fonte: Karras et al. (2017)

No contexto deste novo paradigma, no qual algoritmos são utilizados para sintetização de artefatos criativos, pode-se intuir uma aplicação no campo do *design* de calçado. A partir disto, além de embasar a parte teórica/prática desta pesquisa, justifica-se a investigação no arranjo do vale dos sinos, visto que a região é referência no mercado coureiro-calçadista nacional, concentrando o maior *cluster* de empresas do gênero do mundo, além de responder por uma participação relevante da atividade industrial e da pauta de exportações brasileiras (RODRIGUES; SALOMÃO, 2018).

Além de fábricas terceirizadas que produzem calçados para empresas estrangeiras, a região possui muitas marcas fortes, principalmente com o foco voltado ao público feminino. Logo, as empresas da região precisam empreender esforços no desenvolvimento dos produtos e criação de modelos exclusivos de calçados.

Portanto, tendo em vista as necessidades da região do Vale dos Sinos, vocacionada ao campo do calçado, e a necessidade de uma pesquisa capaz de avaliar as contribuições da criatividade computacional para indústria no *design* de seu principal produto, foi definido como objetivo principal deste trabalho investigar um processo que seja capaz de criar imagens bidimensionais de modelos de calçados por meio do uso de Redes Neurais Artificiais.

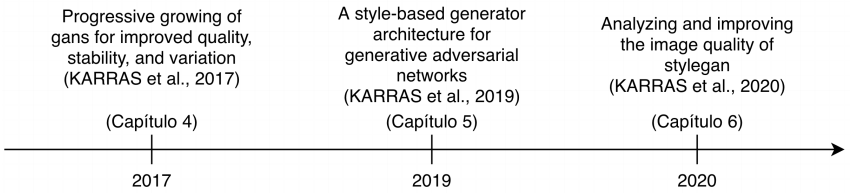
Para que o objetivo principal fosse alcançado, objetivos específicos foram definidos, sendo estes: investigar arquiteturas de Redes Neurais Artificiais bem como uma maneira otimizada de treinar o modelo; formar um *dataset* com imagens de calçados; investigar a implementação do modelo mais adequado a esse cenário; avaliar os resultados gerados.

Tendo em vista os objetivos citados acima, a estrutura deste trabalho foi consolidada de maneira a apresentar o caminho percorrido para a realização de tais objetivos. Sendo assim, o Capítulo 2 apresenta possíveis propostas de geração de imagens de calçados, bem como metodologias para a avaliação das imagens geradas. O Capítulo 3, por sua vez, faz uma introdução ao conceito de Redes Neurais artificias e *Deep Learning*, se aprofundando, então, numa arquitetura denominada GAN, que também é apresentada neste capítulo.

Os próximos três capítulos são análises de trabalhos complementares do mesmo autor principal, sendo o Capítulo 4, o primeiro trabalho apresentado por Karras et al. (2017), uma arquitetura de crescimento progressivo baseada na arquitetura GAN. O Capítulo 5, por sua vez, descreve a arquitetura Stylegan, criada por Karras, Laine e Aila (2019), acrescentando novos conceitos ao trabalho descrito no Capítulo 4.

Por fim, no Capítulo 6, é apresentada uma remodelação da arquitetura Stylegan, onde Karras et al. (2020a) analisa a arquitetura e apresenta melhorias de performance e correção de defeitos. Esta última é chamada de Stylegan2, e foi a arquitetura utilizada neste trabalho. A evolução desta arquitetura pode ser visualizada numa linha do tempo, representada na Figura 3, sendo cada ponto relacionado ao trabalho publicado correspondente.

Figura 3: Linha do tempo da Stylegan



Fonte: Produção do autor

No Capítulo 7 está descrito o desenvolvimento prático deste projeto, em que, inicialmente é proposto um processo para a geração de imagens, e em seguida a execução deste, juntamente com a descrição detalhada de cada uma das etapas. O Capítulo 8, por fim, apresenta os resultados gerados pela execução do processo definido, assim como a avaliação quantitativa e qualitativa destes resultados. Ademais, são apresentadas as considerações finais e trabalhos futuros no Capítulo 8.

2 TÉCNICAS DE GERAÇÃO E AVALIAÇÃO DE IMAGENS

Um método possível de ser utilizado para a geração de imagens é a Geração Procedural. Este método vem sendo utilizado pela indústria de jogos há muitos anos e consiste no uso algorítmico para a geração de dados, baseando-se em artefatos previamente gerados, combinados à aleatoriedade computacional (KELLY; MCCABE, 2007).

O trabalho de Greuter et al. (2003), por exemplo, usa a técnica de Geração Procedural para a criação de cidades pseudo-infinitas em tempo real. Neste trabalho o autor usa subpartes pré-modeladas de construções combinadas com a geração de números pseudo-aleatórios para a geração de novos prédios e construções. Tais construções são geradas automaticamente, baseadas na posição e extrusão nos planos base (GREUTER et al., 2003).

Neste contexto, entende-se que técnicas parecidas poderiam ser aplicadas para a geração de modelos de calçados, gerando diferentes modelos tridimensionais para cada parte do calçado e combinando estas partes aleatoriamente para a criação de novos modelos. Além disto, poderíamos aplicar ruído para deformação nas peças base e na geração das texturas. Esta abordagem, porém, não traz resultados de fato inovadores, pois, os modelos gerados serão uma combinação de componentes pré-modelados com algum grau de variação na textura e geometria. Ademais, é limitada à quantidade de componentes pré-moldados, fazendo com que o máximo de modelos gerados não ultrapasse a combinação de componentes criados.

Ao que concerne às questões citadas acima, compreende-se que uma solução mais adequada para tal problema, possa ser baseada em *Deep Learning*, um subcampo das Redes Neurais Artificiais, do inglês, *Artificial Neural Networks* (NNs). Uma Rede Neural padrão é constituída por muitos processadores simples e interconectados chamados neurônios, cada um produzindo uma sequência de ativações em que neurônios de entrada são ativados por meio de sensores que percebem o ambiente, enquanto outros neurônios são ativados por conexões com pesos, os quais são, por sua vez, ativados por neurônios da camada anterior. Além disso, alguns neurônios podem influenciar o ambiente ao disparar ações (SCHMIDHUBER, 2015).

O aprendizado se trata de encontrar os pesos que façam a Rede Neural exibir o comportamento desejado, tal como dirigir um carro ou gerar uma determinada imagem. Dependendo do problema e de como os neurônios são conectados, esse comportamento pode exigir um longa cadeia de estágios computacionais, em que cada estágio transforma as ativações agregadas da rede. *Deep Learning* se trata de precisamente configurar os pesos em cada um dos estágios da rede (SCHMIDHUBER, 2015).

LeCun, Bengio e Hinton (2015), por sua vez, afirmam que *Deep Learning* descobre a estrutura complexa em grandes *datasets* ao usar o algoritmo *Backpropagation* para indicar como uma máquina deve mudar seus parâmetros internos, estes que são usados para computar a representação em cada camada a partir da representação na camada anterior.

Mais especificamente no campo de geração de imagem, Goodfellow et al. (2014) propôs em 2014 em seu trabalho *"Generative Adversarial Networks"* uma arquitetura de redes neurais baseada em *Deep Learning*, capaz de gerar imagens ao tentar aproximar a distribuição de probabilidade de um determinado *dataset*. Tal arquitetura é chamada de GAN, sigla para *Generative Adversarial Networks*, traduzida para Redes Adversariais Generativas. Os conceitos de GAN e *Deep Learning* serão abordados mais profundamente no Capítulo 3.

2.1 AVALIAÇÃO DE IMAGENS

A avaliação de imagens geradas artificialmente de maneira automática é dificultada pelo fato de que o seu objetivo é criar imagens que pareçam ser reais ao olho humano, trazendo a aparente necessidade do uso de humanos para a avaliação das imagens geradas. Uma alternativa à avaliação de humanos é o uso *Deep Learning* para treinar um modelo que seja capaz de diferenciar imagens geradas de imagens reais e usar o resultado desse modelo como a probabilidade da imagem ser real ou gerada, dando assim, uma noção de quão realistas são as imagens geradas. Esta técnica é usada pelas Redes Adversariais Generativas (GAN) no processo de treinamento do modelo.

Para Borji (2019), alguns dos métodos para avaliar o resultado de modelos generativos tentam avaliar os modelos quantitativamente, enquanto outros enfatizam maneiras qualitativas, tais como estudos de usuários ou análise interna dos modelos e ambos os métodos tem prós e contras. Ainda para os autores, testar o modelo de maneira a fazer com que uma pessoa seja incapaz de distinguir uma imagem gerada pelo modelo de uma imagem real, não é o teste mais efetivo isoladamente, pois, somente com este, não é possível analisar medidas como a diversidade das imagens geradas ou a autenticidade destas em relação aos dados de treino. Medidas quantitativas, porém, enquanto menos subjetivas, podem não corresponder diretamente a maneira em que pessoas julgam as imagens geradas. Tais problemas, juntamente a outros, fazem com que a avaliação de modelos generativos seja notavelmente complicada. Apesar disto, alguns estudos têm procurado trabalhar medidas para se avaliar modelos generativos (BORJI, 2019).

No artigo *"Pros and Cons of GAN Evaluation Measures"*, Borji (2019) faz uma análise de alguns métodos de avaliação de modelos generativos. Para isto, o autor cita uma lista de propriedades a serem avaliadas, afirmando ainda que um bom modelo generativo deve contemplar tais propriedades:

- Favorecer modelos que gerem amostras de alta fidelidade, ou seja, que haja dificuldade ao distinguir imagens geradas de imagens reais;
- Favorecer modelos que gerem amostras diversas;
- Favorecer modelos com espaço latente sem enlace, assim como espaço contínuo (o conceito de espaço latente será apresentado no Capítulo 3);
- Ter limites bem definidos;
- Ser sensível a distorções e transformações nas imagens;
- Estar de acordo com a percepção e julgamento de humanos que avaliem os modelos;
- Ter baixa complexidade computacional.

Baseando-se nas propriedades descritas acima, Borji (2019) faz um estudo que apresenta uma série de métodos quantitativos e qualitativos para a avaliação de modelos generativos que contemplem a avaliação de algumas destas propriedades. Alguns métodos para avaliação de modelos generativos serão citados na sessão a seguir.

2.2 MÉTODOS PARA AVALIAÇÃO QUANTITATIVA E QUALITATIVA

Uma forma de se avaliar quantitativamente as imagens geradas é por meio do método *"Average Log-likelihood"*, ou Probabilidade Logarítmica Média em tradução literal. Este método foi por um tempo o padrão para treinar e avaliar modelos generativos, medindo a probabilidade dos dados verdadeiros sob a distribuição gerada. Porém, sua eficácia foi questionada por Theis, Oord e Bethge (2015) e desde então, vem sendo menos utilizado. Um dos problemas apresentados pelo autor é que a estimativa da probabilidade logarítmica não tem uma grande correlação com a qualidade das amostras, sendo que um modelo com uma probabilidade logarítmica baixa pode produzir amostras de alta qualidade, assim como um modelo com a probabilidade alta pode produzir amostras de baixa qualidade (THEIS; OORD; BETHGE, 2015).

Outro método de avaliação quantitativa muito utilizado se chama *Inception Score*. Proposto por Salimans et al. (2016), é um método amplamente adotado para avaliar Redes Adversariais Generativas, usando uma Rede Neural pré-treinada para capturar as propriedades desejadas das amostras geradas (BORJI, 2019). Com este método, uma grande quantidade de imagens são classificadas usando o modelo pré-treinado, e a partir disto, as probabilidades das imagens pertencentes a cada classe são preditas, sendo posteriormente sumarizadas no *Inception Score*.

Outra medida muito utilizada é a *Fréchet Inception Distance* (FID), a qual mede simultaneamente a qualidade das imagens geradas e a diversidade destas. Tal método foi proposto por Heusel et al. (2017) como melhoria do *Inception Score* (citado anteriormente), o qual os autores afirmam que tem uma boa correlação com a avaliação humana. Para calcular tal métrica, uma rede convolucional pré-treinada é utilizada para extrair características visuais relevantes, tanto das imagens geradas como das imagens reais. As distribuições das imagens geradas e das imagens do *dataset* real são, então, aproximadas em duas distribuições gaussianas. Posteriormente, a *Fréchet distance*, ou Distância Wasserstein-2, é calculada entre as duas distribuições e utilizada como medida de qualidade para o modelo (JIN et al., 2017). Quanto menor for o resultado do cálculo, menor é a *Fréchet distance*, indicando melhor qualidade e maior diversidade das imagens geradas pelo modelo.

No trabalho de Jin et al. (2017), a métrica FID foi utilizada para avaliar o modelo criado e descrito no trabalho "*Towards the Automatic Anime Characters Creation with Generative Adversarial Networks*". Neste trabalho, Redes Adversariais Generativas são usadas para gerar imagens de personagens de *Anime*.

Para o cálculo do FID, os autores selecionaram 12800 amostras do *dataset* real e então geraram amostras falsas usando condições correspondentes a cada amostra do *dataset* original. Posteriormente, alimentaram todas as imagens ao extrator de características em que obtiveram um vetor de características de 4096 dimensões para cada imagem. Finalmente, foi calculado o FID entre os vetores gerados sobre os dados reais e os dados gerados. Esse processo foi repetido 5 vezes para cada modelo, sendo no fim calculada a média destas notas. O resultado pode ser visto na Figura 4.

Figura 4: FID calculado

Model	Average FID	MaxFID-MinFID
DCGAN Generator+DRAGAN	5974.96	85.63
Our Model	4607.56	122.96

Fonte: (JIN et al., 2017)

Além do *FID score* para avaliar as imagens geradas, os autores usaram outro método para a avaliação destas. O método consistia em medir a precisão da imagem gerada no momento em que um determinado *label* era atribuído a esta. Para cada geração de imagem, um dos atributos de entrada era ativado, enquanto todos os outros 33 eram configurados aleatoriamente. Posteriormente, para cada atributo, 20 imagens foram geradas e então checadas manualmente pelos próprios autores. A checagem visava analisar se os *labels* atribuídos eram refletidos nas imagens geradas (JIN et al., 2017).

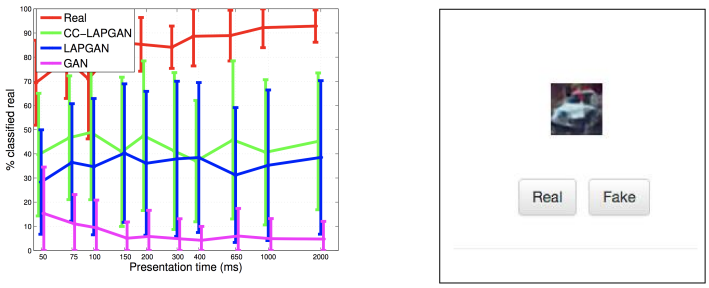
A avaliação das imagens geradas feita por humanos ajuda a inspecionar e ajustar os modelos. Entretanto, apresenta algumas desvantagens, sendo essa, cara, lenta, enviesada, difícil de reproduzir e incapaz de refletir totalmente a qualidade do modelo. Além disto, os avaliadores humanos podem ter uma alta variação, sendo necessária uma grande quantidade de avaliadores para que tais diferenças sejam amenizadas (BORJI, 2019). Por estes motivos, algumas maneiras mais automatizadas de se fazer esta análise qualitativa têm sido desenvolvidas.

Uma das técnicas automatizadas de análise qualitativa se chama *Nearest Neighbors* - traduzida para Vizinhos Mais Próximos - que visa comparar uma amostra gerada pelo modelo com a imagem mais próxima no *dataset* de treinamento, com o objetivo de detectar se não há *overfitting*, ou seja, se os dados gerados não estão simplesmente copiando os dados de treinamento (BORJI, 2019).

Outro método de análise qualitativa comum é o *Rapid Scene Categorization*, - traduzido para Rápida Categorização de Cena - o qual é baseado em estudos prévios. Tais estudos mostram que humanos são capazes de reportar determinadas características das cenas muito rapidamente. Para obter tais medidas qualitativas, Denton et al. (2015) solicitou a 15 voluntários diferentes para que distinguissem imagens geradas de imagens do *dataset* de treinamento. Os voluntários receberam uma interface, por meio da qual, lhes era pedido para pressionar o respectivo botão a fim de classificar a imagem mostrada como real ou gerada (BORJI, 2019). Sendo a precisão uma função do tempo de visualização da imagem, os autores aleatoriamente escolheram um tempo de apresentação de 11 durações variadas entre 50ms e 2000ms, o qual era sucedido pela ocultação da imagem.

Na Figura 5 à esquerda, podem-se ver os resultados das avaliações do trabalho de Denton et al. (2015) comparado a outros trabalhos realizados previamente. À direita está um exemplo de interface que foi apresentada aos usuários para a avaliação das imagens geradas.

Figura 5: Resultados da avaliação por *Rapid Scene Categorization*



Fonte: Denton et al. (2015)

3 GERAÇÃO DE IMAGENS ATRAVÉS DE DEEP LEARNING

Muitas tarefas de inteligência artificial podem ser resolvidas criando-se um conjunto de características a se extrair para essa tarefa, e então, provendo-se essas características para um algoritmo simples de *machine learning*. Para algumas tarefas, porém, é difícil saber exatamente quais características precisam ser extraídas. Por exemplo, se quiséssemos fazer um programa para identificar carros em uma imagem, poderíamos usar como heurística, a existência de rodas nessa imagem. Contudo, é difícil descrever uma roda no domínio de píxeis, pois, esta pode ter reflexos, sombras, estar obstruída por algum outro objeto e assim por diante. Tal dificuldade é conhecida como problema de aprendizado de representação, em que precisaríamos entender não só o resultado do mapeamento da representação fundamental da roda para a saída no formato de imagem, mas também a representação em si (GOODFELLOW; BENGIO; COURVILLE, 2016).

3.1 DEEP LEARNING

Deep Learning é uma técnica que se propõe a resolver o problema mencionado acima, de maneira a introduzir representações que são expressas em termos de outras mais simples. Um exemplo de modelo de *Deep Learning* é o *Multilayer Perceptron* (MLP), o qual é basicamente uma função que mapeia valores de entrada para valores de saída, composta por várias funções mais simples. Cada aplicação pode ser vista como uma função matemática diferente, sendo provedora de uma nova representação de entrada (GOODFELLOW; BENGIO; COURVILLE, 2016).

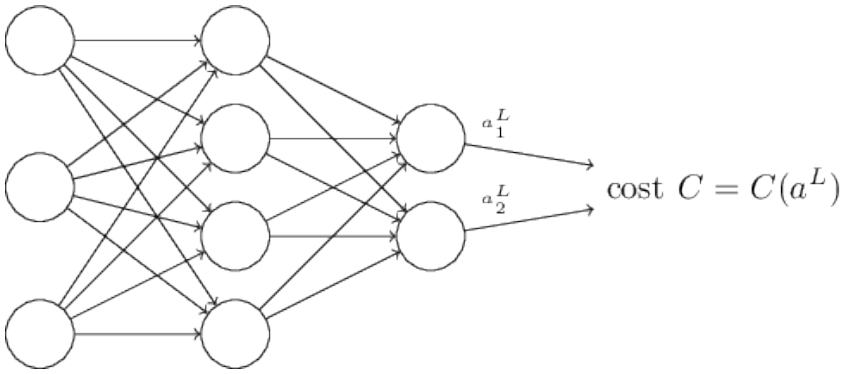
MLPs também são chamadas de *Feedforward Neural Networks* - em tradução literal chamada de Redes Neurais de Alimentação Unidirecional - as quais são denominadas assim, justamente porque a informação flui pela função, sendo avaliada desde a camada de entrada, passando pelas camadas intermediárias e terminando na camada de saída (GOODFELLOW; BENGIO; COURVILLE, 2016).

Para LeCun, Bengio e Hinton (2015), uma arquitetura *Deep Learning* é uma pilha de módulos simples com múltiplas camadas, em que praticamente todas estão submetidas ao aprendizado, e muitas podem calcular mapeamentos não lineares de entradas para saídas. Cada módulo dessa pilha transforma sua entrada com o objetivo de aumentar a seletividade e invariância da representação. Com múltiplas camadas não lineares, um sistema pode implementar funções muito complexas de suas entradas que são simultaneamente sensíveis a detalhes pequenos e insensíveis a grandes variações irrelevantes.

As arquiteturas MLPs podem ser treinadas por meio de um simples gradiente estocástico descendente. Se seus módulos forem funções relativamente suaves de suas entradas e seus pesos internos, tais gradientes podem ser computados a partir de um processo chamado *Backpropagation* (LECUN; BENGIO; HINTON, 2015). Tal processo, para calcular o gradiente de uma função objetivo em relação aos pesos de uma MLP, não é nada além de uma aplicação prática da regra de cadeias para derivadas. O principal ponto é que a derivada do objetivo em relação à entrada de uma camada possa ser calculada por trabalhar, de trás para frente, o gradiente em relação à saída daquela camada. A equação do *backpropagation* pode ser aplicada repetidamente para propagar os gradientes por todas as camadas, começando da camada de saída, até a camada de entrada. Uma vez que os gradientes foram computados, o cálculo destes em relação aos pesos de cada camada é simples (LECUN; BENGIO; HINTON, 2015).

A estrutura de um MLP pode ser observada na Figura 6, em que cada círculo representa um nó da rede e cada grupo vertical de círculos representa uma camada. A camada à extrema esquerda representa a entrada, e a camada à extrema direita representa a saída, tal qual é usada como base para o cálculo do custo a ser utilizado para a geração do gradiente com o processo de *backpropagation*.

Figura 6: Etrutura de um MLP



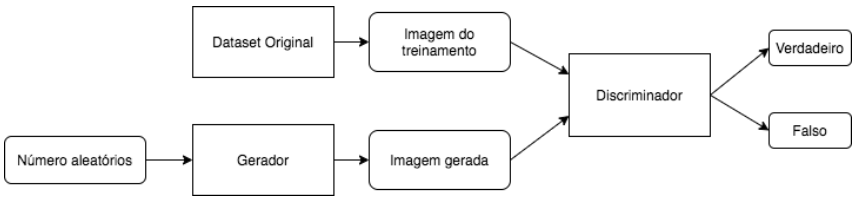
Fonte: (NIELSEN, 2015)

3.2 REDES ADVERSARIAIS GENERATIVAS (GAN)

Rede Adversarial Generativa (GAN) é a denominação de um *framework* que tem como objetivo estimar modelos generativos. Tal *framework* foi proposto no artigo *"Generative Adversarial Networks"*, escrito por Goodfellow et al. (2014), e foi responsável por uma ruptura no uso de *Deep Learning* no campo de modelos generativos. Este, pode ser usado em diversas áreas, para a geração de diferentes tipos de dados, porém, é especialmente interessante quando usado em tarefas de processamento de imagens como melhora de nitidez, aumento de resolução, preenchimento de partes apagadas, e até a geração de imagens novas (METZ et al., 2016).

O *framework* em questão define uma arquitetura composta por duas Redes Neurais separadas. Uma cumpre a função de geradora, enquanto a outra cumpre a função de discriminadora. A rede geradora recebe de entrada um vetor de ruído aleatório, tendo então, a responsabilidade de gerar imagens como saída, baseadas nesse ruído. A rede discriminadora, por sua vez, recebe a imagem gerada pela rede geradora ou um elemento do *dataset* de imagens originais como entrada, tendo como objetivo, categorizar a imagem gerada como sendo falsa ou verdadeira. Tal processo pode ser visualizado na Figura 7 (GOODFELLOW et al., 2014).

Figura 7: Arquitetura de uma GAN



Fonte: Produção do autor

O autor do trabalho em questão, faz uma analogia do processo generativo com um time de falsificadores que tenta produzir dinheiro falso sem serem identificados, assim como o processo discriminativo pode ser comparado à polícia, que tenta identificar o dinheiro falso gerado pelos falsificadores. A competição entre os dois grupos faz com que ambos aperfeiçoem seus papeis: os falsificadores procuram tornar suas cópias cada vez mais indistinguíveis de cópias reais, enquanto os policias, por sua vez, procuram aperfeiçoar a identificação do dinheiro falso (GOODFELLOW et al., 2014). Assim como o intuito do grupo de policiais, o objetivo do processo discriminativo é identificar se um determinado dado é real ou falsificado. O objetivo do processo generativo, por sua vez, é gerar dados de maneira que a rede discriminativa não consiga identificá-los como falsos.

Para que se treine simultaneamente a rede discriminativa e a rede generativa, estas são treinadas alternadamente usando uma regra de atualização baseada em gradientes, em que, durante cada passo do treinamento, tanto na rede discriminativa quanto na generativa, um grupo de amostras, advindos do *dataset* original ou da geração da rede generativa, é usado para calcular a perda e o gradiente para a rede correspondente. Os parâmetros da rede são, então, atualizados usando o gradiente (WINKLER, 2018).

O objetivo de modelo generativo é fazer com que a distribuição gerada pela rede generativa seja igual à distribuição das imagens reais. Portanto, minimizar a diferença entre as duas distribuições é um ponto crucial para o treinamento de modelos generativos (ALQAHTANI; KAVAKLI-THORNE; KUMAR, 2019). Este é o objetivo da rede discriminativa usada pela arquitetura tradicional da GAN.

A rede geradora, por sua vez, faz o papel de uma função que recebe as características necessárias para a representação da imagem gerada, e tem como saída a imagem em si. Estas características, que são passadas para rede generativa, são chamadas de espaço latente, e a rede precisa aprender o que cada valor neste espaço representa.

Espaço latente é o espaço onde a representação comprimida dos dados está. Se quisermos alterar alguns atributos da imagem (como pose, idade ou até um objeto, no caso da imagem de uma pessoa), modificar a imagem diretamente no espaço de píxeis seria muito difícil, pois, a variedade onde a distribuição se encontra é de alta complexidade e multidimensional. Ao contrário disto, manipular o espaço latente é mais rastreável, pois, este expressa características específicas da imagem de entrada de uma maneira comprimida. (ALQAHTANI; KAVAKLI-THORNE; KUMAR, 2019)

Para Metz et al. (2016), GANs sofrem com muitos problemas, particularmente durante o treinamento. Uma de suas falhas comuns envolve o colapso da rede generativa ao produzir apenas uma única amostra ou uma família pequena de amostras. Além deste, cabe ressaltar outro problema que envolve as redes discriminativa e generativa, que oscilam durante o treinamento, ao invés de convergir num ponto único. Além disto, se um dos agentes fica muito mais poderoso que o outro, o sinal de aprendizado para os outros agentes se torna inútil e o sistema não aprende.

GANs, como formuladas originalmente por Goodfellow et al. (2014) no artigo "Generative Adversarial Networks", produzem imagens nítidas, porém, numa resolução baixa e com um grau de variação relativamente limitado, como pode ser observado na Figura 1. Além disto, seu treinamento é instável, entretanto, está a frente dos outros modelos generativos baseados em *Deep Learning* (KARRAS et al., 2017). As otimizações propostas por Karras et al. (2017) serão exploradas no Capítulo 4.

4 OTIMIZAÇÃO DE GANS COM CRESCIMENTO PROGRESSIVO FUNDAMENTADO POR KARRAS ET AL. (2017)

Para Karras et al. (2017) parte dos problemas da arquitetura original das GANs se deve ao fato de que a rede discriminativa cumpre o papel de uma função de perda adaptativa. Quando é medida a distância das distribuições geradas e treinadas, se estas distribuições não tem uma sobreposição suficiente, os gradientes apontam para direções aleatórias, ou seja, diferenciá-las torna-se uma tarefa simples.

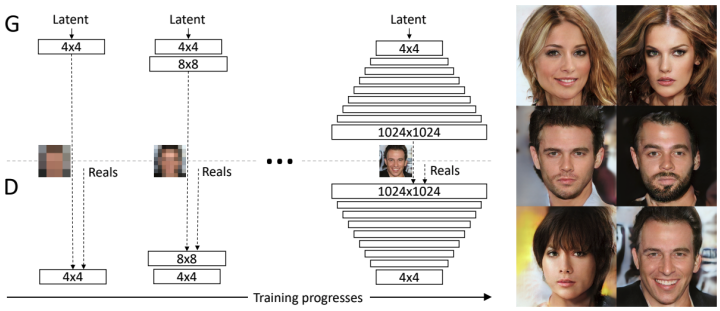
No artigo *"Progressive growing of GANs for improved quality, stability, and variation"*(KARRAS et al., 2017), os autores afirmam que a geração de imagens de alta resolução maximiza o problema dos gradientes citado anteriormente, fazendo com que a diferenciação entre as imagens geradas e de treinamento seja facilitada. A resolução alta também gera limitações de memória, o que traz a necessidade do uso de lotes de treinamento menores. A principal contribuição deste artigo, se dá justamente na questão da resolução, de tal modo, os autores propõem uma mudança na arquitetura da GAN.

A arquitetura proposta sugere o crescimento progressivo, tanto da rede discriminativa quanto da rede generativa, começando em imagens simples, de baixa resolução, e adicionando novas camadas que introduzem detalhes de mais alta resolução com o progresso do treinamento. Isto faz com que o treinamento fique mais rápido e melhore a estabilidade em resoluções altas, além de fazer com que a rede primeiro, descubra a estrutura em alto nível da distribuição da imagem, e posteriormente, mude a atenção para aperfeiçoar detalhes menores, ao invés de ter de aprender todos os níveis simultaneamente (KARRAS et al., 2017).

Os autores usam redes generativas e discriminativas que são imagens espelhadas entre si, sob o ponto de vista das camadas de resolução, e incrementam progressivamente as duas em sincronia. Todas as camadas são mantidas treináveis durante todo o processo de treinamento e quando novas camadas são adicionadas, estas são inseridas com uma transição suave para evitar mudanças abruptas nas camadas já treinadas de menores resoluções (KARRAS et al., 2017).

Tal processo pode ser observado na Figura 8, em que a rede generativa é representada por G e a rede discriminativa é representada por D, as quais, têm no inicio do treinamento, uma resolução baixa de 4x4 píxeis. No decorrer do treinamento, camadas são adicionadas incrementalmente às redes generativas e discriminativas, assim, aumentando a resolução das imagens geradas. À direita estão exemplos de imagens criadas, usando crescimento progressivo com a resolução de 1024 x 1024 (KARRAS et al., 2017).

Figura 8: Processo de treinamento de uma ProGAN



Fonte: Karras et al. (2017)

Com base no método de treinamento progressivo, pode-se notar alguns benefícios em relação aos métodos anteriores. No inicio do processo, a geração de imagens menores é consideravelmente mais estável e ao incrementar progressivamente a resolução, a rede precisa descobrir cada vez menos informações novas a cada etapa. Na prática, este processo estabiliza o treinamento de maneira suficiente para a geração confiável de imagens de alta resolução (KARRAS et al., 2017).

Para Karras, Laine e Aila (2019) num artigo chamado *"A Style-Based Generator Architecture for Generative Adversarial Networks"*, apesar dos avanços alcançados, tanto com a criação da arquitetura progressiva, quanto com outros estudos relacionados às Redes Adversarias Generativas, a rede generativa ainda agia como uma caixa preta, e a compreensão de diversos aspectos do processo de sintetização da imagem, como por exemplo, a origem de características estocásticas, ainda faltavam. Os autores também afirmam que as propriedades do espaço latente eram mal compreendidas e as demonstrações comuns de interpolação deste, não proviam nenhuma maneira quantitativa de distinguir geradores uns dos outros. Este trabalho será explorado no Capítulo 5.

5 GAN BASEADA EM ESTILOS FUNDAMENTADO POR KARRAS, LAINE E AILA (2019)

O capítulo atual, na sua totalidade, é a apresentação do estudo publicado por Karras, Laine e Aila (2019), em que os autores propõem uma reescrita na arquitetura da rede generativa de maneira a expor novas maneiras de controlar o processo de sintetização das imagens, motivados pela literatura de transferência de estilos (*style transfer*). Esta nova arquitetura começa a partir de uma entrada constante aprendida, e ajusta o estilo da imagem, baseada no espaço latente, em cada camada de convolução, fazendo assim, com que as características da imagem sejam controladas em diferentes escalas. Combinada com vetores aleatórios injetados diretamente na rede, essa arquitetura leva à separação automática e não supervisionada de atributos de alto nível (como pose ou cor de pele, em imagens de pessoas) da variação estocástica (como posição do cabelo ou rugas) nas imagens geradas. Tal arquitetura não modifica a rede discriminativa, nem a função de perda (KARRAS; LAINE; AILA, 2019).

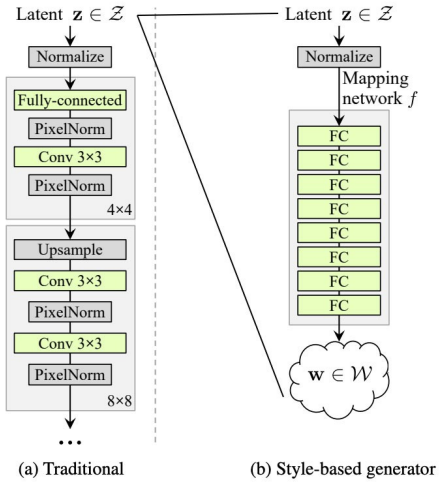
Na arquitetura original da GAN, a distribuição do espaço latente, representada pelo vetor aleatório de entrada da rede generativa, deveria representar a distribuição de imagens reais. Se ao invés disto, esse vetor for gerado a partir de uma distribuição normal ou uniforme, um modelo otimizado necessitará que esse vetor contenha informações extras, além do tipo ou estilo. Por exemplo, se quisermos gerar imagens de soldados e visualizar a distribuição dos dados de treinamento com dois fatores latentes, sendo estes, masculinidade e comprimento de cabelo, notaremos que uma parte da distribuição estará com valores zerados, pois, soldados homens não podem ter cabelo comprido, mostrando uma relação entre estas duas características. Se pegarmos o vetor de entrada da rede generativa baseado numa distribuição uniforme, a rede tentará produzir imagens de soldados homens com cabelo comprido. A rede discriminativa, então, deverá classificar tal imagem como falsa, pois, nos dados de treinos não há nenhum soldado homem de cabelos compridos (HUI, 2020). Esta propriedade é conhecida como enlace, pois, uma determinada característica está fortemente atrelada à outra.

A rede generativa proposta pelos autores encapsula o espaço latente de entrada num espaço latente intermediário, o que afeta profundamente os fatores de variação apresentados por tal gerador. Este espaço latente intermediário é livre, e portanto, é possível diminuir o enlace deste, o que para os autores não é possível com o espaço latente de entrada, por precisar seguir a densidade de distribuição dos dados de treino (KARRAS; LAINE; AILA, 2019).

Este encapsulamento é feito por meio de uma rede de mapeamento, que transforma o vetor de entrada nesse espaço latente intermediário, usando oito camadas totalmente conectadas. Através desta rede de mapeamento, o espaço latente de entrada de 512 dimensões é transformado em um espaço intermediário de mesmo tamanho (HUI, 2020).

Tal mapeamento pode ser visualizado na Figura 9, em que (a) representa uma GAN tradicional e (b) mostra o novo espaço latente proposto pela *StyleGAN* com a função de mapeamento da entrada para um espaço latente intermediário. Como nota-se na imagem, o espaço latente Z da GAN tradicional, cuja parte da arquitetura está representada à esquerda, na imagem, é substituído, na nova arquitetura, pelo espaço latente W, o qual está representado à direita, juntamente à função de mapeamento. Tal função consiste em uma Rede Neural com 8 camadas completamente conectadas, na imagem representada pelos blocos inscritos com (FC) - sigla para *Fully Connected*, traduzido para Completamente Conectada.

Figura 9: Mapeamento do espaço latente numa StyleGAN



Fonte: Fonte: (HUI, 2020)

Com base na análise de métricas qualitativas, os autores alegam que a Stylegan é de todas as maneiras superior a GAN tradicional. Ainda afirmam que novos métodos para moldar diretamente o espaço latente intermediário durante o treinamento podem vir a trazer grandes avanços em trabalhos futuros (KARRAS; LAINE; AILA, 2019).

6 ANALISANDO E MELHORANDO A QUALIDADE DA STYLEGAN POR KARRAS ET AL. (2020)

Para Karras et al. (2020a) o modelo definido pela arquitetura de GAN *Stylegan*, apresentada no trabalho citado no Capítulo 5, rendem resultados satisfatórios na geração não condicional de imagens dirigidas por dados. No artigo *Analyzing and Improving the Image Quality of StyleGAN*, os autores analisam os resultados gerados por tal modelo, expõem alguns artefatos característicos deste e propõem alterações no treinamento para mitigar tais artefatos.

Os autores apresentam um artefato no formato de bolha que aparece recorrentemente nas imagens geradas. Tais formações são atribuídas a uma falha de *design* na arquitetura causada pela camada de normalização (KARRAS et al., 2020a). Exemplos deste podem ser visualizados na Figura 10.

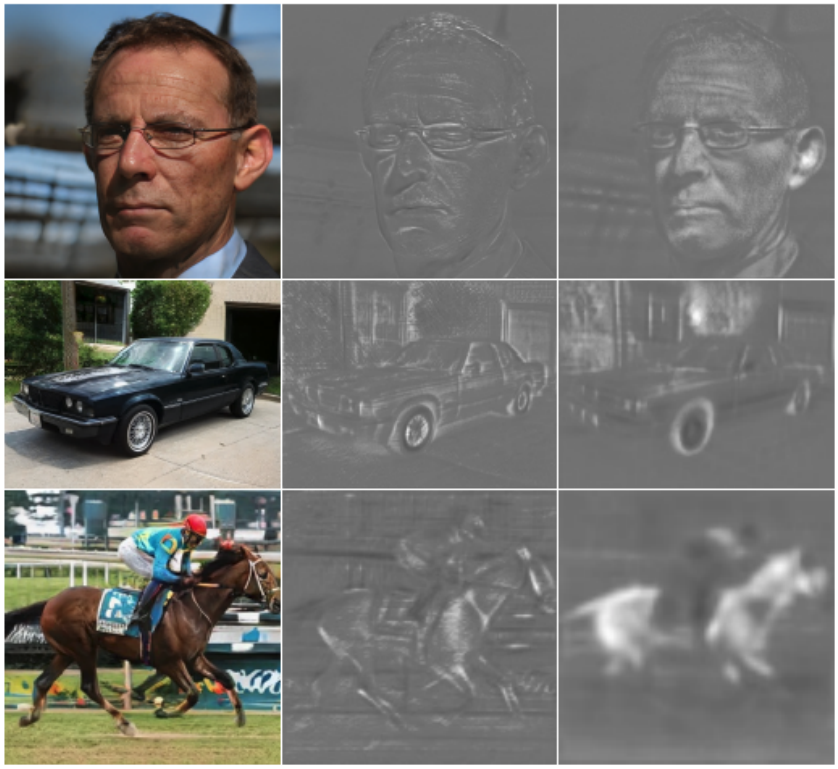
Figura 10: Artefatos parecidos com bolhas

Fonte: (KARRAS et al., 2020a)

Mesmo que por vezes não visíveis, estes artefatos foram observados pelos autores nas ativações da rede generativa em todas as imagens geradas pela StyleGAN, começando a aparecer em todos os *feature maps* desde a resolução 64x64. Como citados anteriormente, os autores atribuem este problema à camada de normalização que é alimentada à entrada de cada camada de convolução do modelo. A camada de normalização é vital para o controle das imagens geradas com a mistura de estilos (KARRAS et al., 2020a).

Os autores argumentam que poderiam simplesmente remover tal camada, mitigando, assim, os artefatos parecidos com bolhas e melhorando as métricas da arquitetura. Tal remoção, porém, faria com que o controle do estilo das imagens fosse removido, perdendo, assim, uma das grandes vantagens dessa arquitetura. Ao invés disto, é proposta uma alteração na camada de normalização, onde os autores trocam um dos subcomponentes desta camada chamado *instance normalization*, em português, normalização da instancia, por um subcomponente que executa uma operação de demodulação (KARRAS et al., 2020a). Tal alteração foi capaz de remover os artefatos das imagens geradas, como pode ser observado na figura Figura 11.

Figura 11: Imagens geradas após a alteração na camada de normalização



Fonte: (KARRAS et al., 2020a)

Ainda neste trabalho, os autores citam um problema com o crescimento progressivo. Eles afirmam que este método tem muito sucesso ao estabilizar a síntese de imagens de alta resolução, porém, gera algumas características indesejadas, sendo o ponto principal delas, a preferência do gerador para a localização de detalhes na imagem gerada. Os autores mostram que, na aplicação do modelo para a geração de imagens de rostos de humanos, características como dentes e direção dos olhos deveriam se mover suavemente nas imagens dependendo da direção em que o rosto está orientado, porém, tais características tendem a ficar fixas em locais específicos antes de pular para o próximo local de preferência da rede (KARRAS et al., 2020a).

Para mitigar tal problema, os pesquisadores investigaram em outras arquiteturas de GANs que não usam crescimento progressivo. Como resultado, criaram uma nova arquitetura, sem o crescimento progressivo, conseguindo manter os principais benefícios da ProGAN, que é o foco em características de baixa resolução no início do treinamento e a mudança deste foco para características mais detalhadas no fim do treinamento (KARRAS et al., 2020a).

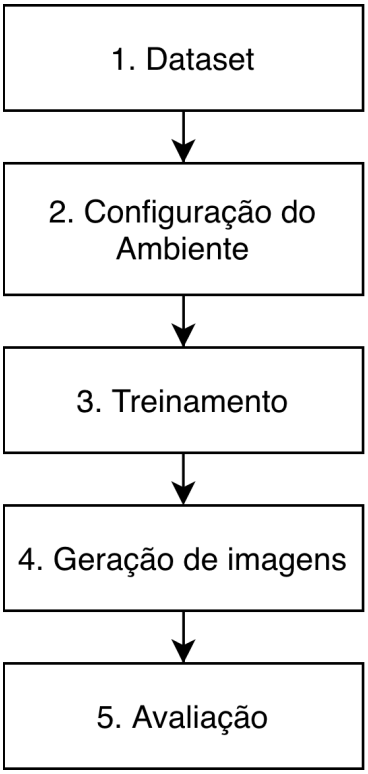
Os autores concluem afirmando que foram capazes de melhorar a qualidade ao resolver diversas falhas na primeira versão da arquitetura Stylegan, além de melhorar a performance no treinamento (KARRAS et al., 2020a).

7 CRIAÇÃO DE IMAGENS BIDIMENSIONAIS DE CALÇADOS

Com base no referencial teórico, foi definido um processo que foi implementado para cumprir os objetivos definidos no inicio deste trabalho. Este processo está sumarizado na Figura 12, contendo 5 etapas, as quais serão apresentadas em mais detalhes, juntamente com o ambiente, durante este capítulo, com exceção da avaliação, que será apresentada no Capítulo 8.

Assim, conforme a Figura 12, a etapa 1 representa a aquisição de um *dataset* com imagens bidimensionais de calçados. Enquanto, a etapa 2 se refere à configuração do ambiente necessário para o treinamento e geração das imagens. Na etapa 3, as imagens adquiridas são usadas como entrada para a rede StyleGAN, a qual é responsável pelo treinamento e criação do gerador de imagens. O gerador de imagens, por sua vez, está contido na etapa 4, que se refere ao processo de geração de imagens a partir de vetores aleatórios de entrada. Por fim, na etapa 5, está a avaliação das imagens, que utiliza a métrica FID, que foi apresentada no Capítulo 2.

Figura 12: Fluxo do processo de treinamento do Gerador de imagens de calçados



Fonte: Produção do autor

7.1 DATASET

Como pode ser visto na etapa 1 da Figura 12, é necessária a aquisição de um *dataset* a ser utilizado, posteriormente, pelo treinamento da rede. O *dataset* escolhido para a execução do processo foi o UT Zappos50K, o qual foi consolidado por Yu e Grauman (2014) e disponibilizado pela Universidade do Texas.

Tal *dataset* consiste em 50.025 imagens catalogadas, coletadas do site zappos.com. As imagens foram divididas em 4 categorias principais, sendo estas: sapatos, sandálias, chinelos e botas - seguidos por tipos funcionais e marcas individuais. Nas imagens, os sapatos estão centralizados num fundo branco e todos estão posicionados com a mesma orientação. A formatação das imagens do *dataset* pode ser observada na figura 13, a qual contém algumas imagens retiradas do *dataset* UT Zappos50K.

Figura 13: Amostra do *dataset* UT Zappos50K

Fonte: (YU; GRAUMAN, 2014)

Este *dataset* foi criado no contexto de tarefas de compras online, onde usuários põem uma atenção especial em detalhes refinados nas imagens apresentadas. Como os compradores podem ter diferentes características, necessidades e estilos, o site possui uma grande variedade de calçados com estilos diferentes, o que é consequentemente refletido no *dataset*. Adicionalmente, cada imagem tem 8 *labels* de metadados associados que foram usados para filtrar os sapatos no site citado.

Apesar de o *dataset* dispôr de alguns metadados, representados por meio de uma estrutura de diretórios, uma etapa de tratamento nos dados foi executada antes do treino removendo tais informações, pois, como um dos objetivos da arquitetura Stylegan é aprender as características do espaço latente, tais informações são desnecessárias para o treinamento.

Os metadados citados acima foram, então, suprimidos com a utilização de uma ferramenta chamada `flatten-directory` [1]. Esta ferramenta percorre recursivamente o diretório raiz, renomeando os arquivos com a adição dos diretórios percorridos como prefixo para evitar colisões e copiando os arquivos para um diretório de saída. O comando chamado está representado abaixo.

```
flatten-directory --rootdir="shoes" --outputdir="shoes-flattened"
```

Onde destacam-se os parâmetros:

- `--rootdir="shoes"`: Diretório onde se encontram as imagens na estrutura de diretórios original;
- `--outputdir="shoes-flattened"`: Diretório de saída, onde as imagens são copiadas.

Além de suprimir os metadados, foi necessária outra etapa de transformação nos dados, aumentando o tamanho das imagens de 188x188 para 256x256. Tal etapa foi necessária, pois, a Stylegan precisa que os dados de entrada tenham tamanhos que sejam potências de 2.

A mudança na resolução foi feita com um *script* em Python, que pode ser visualizado a seguir. No *script*, as imagens do diretório são listadas, em seguida ocorre a iteração sobre estas imagens, em que, a cada etapa da iteração, a imagem é redimensionada, salva e por fim, a imagem antiga é removida. Este script pode ser observado abaixo.

```
1  from PIL import Image
2  import glob
3  import os
4
5  path="/Users/usr/src/tcc/shoes"
6  image_filenames = sorted(glob.glob(os.path.join(path, '*.jpg')))
7  length = len(image_filenames)
8  count=0
9  for img_path in image_filenames:
10     if count%100 == 0:
11         print((count * 100/length))
```

[1] disponibilizada em https://www.npmjs.com/package/flatten-directory

```
12        img_path
13        im = Image.open(img_path)
14        new_size=(256,256)
15        resized = im.resize(new_size)
16        new_name = img_path.replace('jpg', 'png')
17        resized.save(new_name)
18        os.remove(img_path)
19        count+=1
```

7.2 CONFIGURAÇÃO DO AMBIENTE

Tendo o *dataset* configurado, a etapa 2 pôde então ser iniciada. Para o treinamento, foi utilizada a base de código oficial disponibilizada pela empresa NVIDIA, baseada na publicação do trabalho citado no Capítulo 6 por Karras et al. (2020b). Além disso, é importante mencionar que todas as configurações, incluindo a função de ativação, foram preservadas tal como se encontravam no código fonte disponibilizado em Karras et al. (2020b).

A base de código citada acima, tem como requerimento, além de diversas bibliotecas utilitárias, os seguintes itens:

- Sistema operacional Linux ou Windows (KARRAS et al., 2020b);
- Instalação da linguagem Python na versão 3.6 (KARRAS et al., 2020b);
- TensorFlow 1.14 ou 1.15 (KARRAS et al., 2020b);
- Uma ou mais placas de vídeo (GPUs) NVIDIA modernas, com drivers da NVIDIA, CUDA 10.0 assim como cuDNN (KARRAS et al., 2020b).

Tensorflow é uma sistema de aprendizado de máquina para a operação em larga escala em diversos ambiente diferentes, que usa fluxo de dados para representar operações que alterem estados, estados compartilhados e processos computacionais, ao mapear os vértices do fluxo de dados para diferentes grupos de dispositivos computacionais, como processadores com múltiplos núcleos ou, no caso desta aplicação, múltiplos núcleos de placas de vídeo (ABADI et al., 2015).

Compute Unified Device Architecture. (CUDA) - Arquitetura de dispositivos computacionais unificados - por sua vez, se trata de um conjunto de ferramentas que disponibiliza ao desenvolvedor um ambiente para a criação de aplicações aceleradas por GPU. Tal ambiente tem como objetivo abstrair linguagens de placas de vídeos complexas (*shader languages*), possibilitando ao desenvolvedor a utilização de primitivas gráficas (NVIDIA; VINGELMANN; FITZEK, 2020).

A biblioteca cuDNN, por fim, se trata de um de uma biblioteca de implementações de primitivas de *deep learning*. Assim como o objetivo da CUDA é disponibilizar ao desenvolvedor chamadas de primitivas gráficas, o objetivo da cuDNN é disponibilizar chamadas de primitivas de *Deep Learning*, abstraindo assim, a implementação da paralelização dos estágios necessários no treinamento e execução de modelos de *Deep Learning* (CHETLUR et al., 2014).

Dados todos os requerimentos citados acima, uma máquina virtual foi alugada contendo os requisitos de hardware necessários para a execução do treinamento. A máquina foi alugada no portal www.paperspace.com, uma empresa de *Infrastructure as a service*, que disponibiliza dois tipo de serviço, chamados *Gradient* e *Core* (PAPERSPACE, 2020).

O serviço *Gradient* se trata de ferramentas para Aprendizado de Máquina e Ciência de Dados, provendo funcionalidades como *Jupyter notebooks* - Uma ferramenta para execução interativa de *scripts* em python - e o Gerenciamento de modelos para *Machine Learning*. O CORE, por outro lado, disponibiliza *templates* de máquinas virtuais com acesso a GPU (PAPERSPACE, 2020).

Como a base de dados a ser utilizada já possui uma estrutura definida, foi contratada uma máquina virtual do serviço CORE. A máquina contratada é do tipo P5000, e fica na costa Leste dos Estados Unidos, contendo os seguintes atributos de *hardware*:

- Processador: Intel Xeon (PAPERSPACE, 2020);
- Número de CPUs: 8 vCPUs (PAPERSPACE, 2020);
- Velocidade de *Clock*: 2.60 GHz (PAPERSPACE, 2020);
- Memória principal: 30 GB (PAPERSPACE, 2020);
- Placa de vídeo: NVIDIA QUADRO P5000 (PAPERSPACE, 2020);
- Memória da placa de vídeo: 16 GB (PAPERSPACE, 2020).

Para contratar a máquina virtual, foi necessário o envio de uma mensagem para a empresa com uma justificativa pela qual estava sendo feita tal requisição, juntamente com a escolha do tipo de máquina e o *template* utilizado.

O *template* a ser utilizado, diz respeito à configuração inicial da máquina virtual, sendo a escolhida para o desenvolvimento deste trabalho, o *template* chamado *ML-in-a-Box*. Este *template* vem com a distribuição Ubuntu do sistema operacional Linux instalado, além dos *drivers* da NVIDIA, da linguagem Python e um conjunto de bibliotecas para *Machine Learning* (PAPERSPACE, 2020).

Depois de escolhidas as configurações necessárias para a máquina virtual, foi necessário cadastrar um cartão de crédito internacional, o qual foi utilizado para a cobrança do aluguel da máquina e armazenamento. O custo destes foi de $5 USD (Dólares Americanos) mensais para 50 GB de armazenamento, adicionados a $0.78 USD por hora em que a máquina fica ligada (PAPERSPACE, 2020). Tais valores foram obtidos na página www.paperspace.com/pricing, no dia 02 de novembro de 2020.

Feitas todas estas etapas, uma máquina nova pode ser criada pelo portal do Paperspace. Após criada, a máquina pode ser iniciada pela interface visual disponibilizada pelo portal, e acessada de três maneiras: (1) área de trabalho virtual, (2) terminal virtual, ou (3) conexão via SSH, sendo que a última necessita que seja configurado um IP público para a máquina com o custo de $3 USD mensais - valor obtido na página www.paperspace.com/pricing, no dia 02 de novembro de 2020.

Após iniciada a máquina virtual, esta foi acessada usando o terminal virtual. Com acesso a máquina, pôde-se clonar a base de código, citada no inicio desta sessão, utilizando Git - uma ferramenta de versionamento de software - e instalar as dependências do repositório.

Antes de se iniciar o treinamento, é necessária a conversão das imagens para `tfrecords`, formato utilizado pelo TensorFlow. Para isso, é necessário executar um comando disponibilizado no repositório como descrito abaixo:

```
python dataset_tool.py create_from_images datasets/shoes shoes
```

Onde se destacam:

1. `python dataset_tool.py`: Chamada do arquivo de ferramentas de *dataset*;
2. `create_from_images`: Comando para gerar um *dataset* de imagens customizadas;
3. `datasets/shoes`: Diretório onde estão localizadas as imagens;
4. `shoes`: Diretório destino dos `tfrecords`.

7.3 TREINAMENTO

Com o ambiente propriamente configurado, foi executado o inicio do treinamento da rede, usando como entrada o *dataset* UT Zappos50K apresentado anteriormente, através do comando descrito a seguir.

```
nohup python run_training.py --num-gpus=1 --data-dir=~/datasets
--dataset=shoes --config=config-f
```

Cada uma das partes está descrita abaixo:

1. `nohup`: Uma ferramenta que permite a escrita do console em um arquivo de texto, fazendo com que os dados não sejam perdidos caso a sessão seja encerrada;

2. `python run_training.py`: Chamada do arquivo que contém o código responsável pelo treinamento;

3. `--num-gpus=1`: Configuração de quantas placas de vídeo devem ser usadas no treinamento. Foi configurada como 1, pois, foi contratada apenas uma GPU;

4. `--data-dir=./datasets`: Diretório onde estão contidos os *datasets*;

5. `--dataset=shoes`: *Dataset* a ser utilizado no treinamento;

6. `--config=config-f`: Configuração de utilização de técnicas diferentes usadas pela rede, em que `config-f` é a padrão. Outras configurações podem ser observadas na Figura 14, retirada do código do treinamento. Estas configurações não são apresentadas no artigo original nem na documentação do repositório, porém, o código possui alguns comentários que indicam a diferença entre as configurações.

Figura 14: Configurações disponíveis para treinamento da Stylegan

```
_valid_configs = [
    # Table 1
    'config-a', # Baseline StyleGAN
    'config-b', # + Weight demodulation
    'config-c', # + Lazy regularization
    'config-d', # + Path length regularization
    'config-e', # + No growing, new G & D arch.
    'config-f', # + Large networks (default)

    # Table 2
    'config-e-Gorig-Dorig',   'config-e-Gorig-Dresnet',   'config-e-Gorig-Dskip',
    'config-e-Gresnet-Dorig', 'config-e-Gresnet-Dresnet', 'config-e-Gresnet-Dskip',
    'config-e-Gskip-Dorig',   'config-e-Gskip-Dresnet',   'config-e-Gskip-Dskip',
]
```

Fonte: (KARRAS et al., 2020b)

O treinamento foi executado de maneira que, a cada determinado intervalo de tempo chamado *tick*, um *snapshot* da rede foi salvo juntamente com um conjunto de imagens geradas pela rede até o momento. Estes *snapshots* são arquivos que contém todo o estado da rede.

Por causado do tamanho dos arquivos, o treinamento foi interrompido na primeira execução, pois, o limite de armazenamento da máquina virtual de 50 GB foi atingido, sendo necessária, então, a limpeza recorrente dos *snapshots* mais antigos. Porém, o treinamento foi resumido do último *snapshot* gerado, fazendo com que possíveis problemas na execução do treinamento não resultassem na perda de progresso. A evolução das imagens geradas pela rede serão apresentadas no Capítulo 8

O tempo de treinamento da primeira execução foi de aproximadamente 10 horas, enquanto a segunda execução durou aproximadamente 25 horas. Os dados detalhados do treinamento estão na Tabela 2 e na Tabela 3, localizadas no apêndice deste trabalho.

7.4 GERAÇÃO DE IMAGENS

Após a etapa anteriormente apresentada, como resultado do treinamento, um arquivo com a extensão `pkl` foi gerado. Esta extensão representa arquivos binários da biblioteca *Pickle*, do Python, a qual serve para serializar dados num formato binário. O arquivo foi, então, usado pelo gerador para carregar a rede com seus pesos em tempo de execução. Cada um dos *snapshots* gerados pôde ser carregado pelo gerador, permitindo que imagens fossem geradas, baseadas no estado da rede naquele momento.

O comando usado para gerar imagens tem o seguinte formato:

```
python run_generator.py generate-images \
    --network=gdrive:networks/stylegan2-ffhq-config-f.pkl \
    --seeds=66,230,389,1518 \
    --resut-dir=./result-images
```

Onde se destacam as seguintes partes:

1. `python run_generator.py`: Chamada do arquivo que contém o código para a geração das imagens;
2. `generate-images`: Comando passado para o arquivo para a geração de imagens;
3. `--network=gdrive:networks/stylegan2-ffhq-config-f.pkl`: Arquivo gerado no treinamento, contendo as informações da rede;
4. `--seeds=66,230,389,1518`: Vetores aleatórios de entrada separados por vírgula. Cada inteiro representa uma entrada no espaço latente, que consequentemente é responsável pela geração de uma imagem diferente;
5. `--resut-dir=./result-images`: Diretório no qual as imagens geradas serão escritas.

Como resultado de execução deste comando, uma imagem no formato `PNG` é gerada para cada uma das *seeds* passadas como parâmetro no diretório especificado, podendo, então, ser observada e avaliada.

8 RESULTADOS

O resultado pode ser observado na Figura 15, onde foram geradas 9 imagens baseadas em 9 *seeds* diferentes, através da chamada do comando citada ao final do Capítulo 7. A *seed* representa o vetor aleatório de entrada, que então será mapeado para o espaço latente intermediário. Apesar de a *seed* não ser a representação no espaço latente em si, o mapeamento é sempre o mesmo, fazendo com que imagens de sapatos geradas com a mesma *seed* representem o mesmo espaço latente, resultando assim no mesmo calçado.

Figura 15: Imagens geradas

Fonte: Produção do Autor

Como citado no Capítulo 7, a métrica utilizada para avaliação foi a FID. Como descrito no Capítulo 2, o resultado do cálculo desta métrica reflete a qualidade das imagens geradas, assim como a diversidade destas, se comparadas às imagens do *dataset* original, sendo que quanto mais alta for a pontuação, menos diversas ou de menos qualidade são as imagens geradas, ao passo que, quanto mais baixa a pontuação, mais alta a qualidade e maior a diversidade dessas imagens.

A métrica FID foi calculada 4 vezes durante o período de treinamento, sendo a primeira destas após 1 minuto de treinamento da rede, enquanto a última pontuação foi calculada após aproximadamente 31 horas de treinamento.

Os resultados podem ser visualizados na Tabela 1, na qual a coluna *Tick*, corresponde à quantidade de *ticks* executados até então, sendo que em cada *tick* um *snapshot* foi salvo. A coluna "Tempo de treinamento", por sua vez, indica o tempo de treinamento agregado da rede até o momento do cálculo, enquanto a coluna "Tempo de Cálculo"corresponde à quantidade tempo que o FID levou para ser calculado na respectiva iteração. A coluna FID, por fim, representa a pontuação calculada para a rede naquele determinado *tick*.

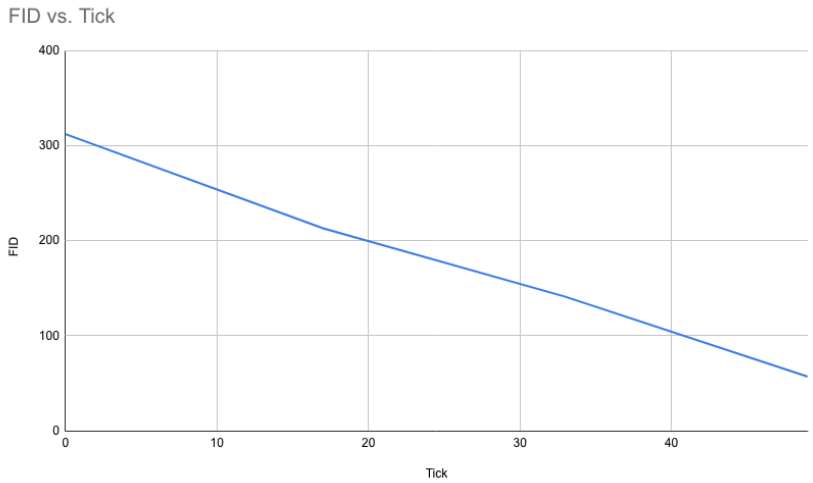
Tabela 1: FIDs calculados

Tick	Tempo de treinamento	Tempo de duração do cálculo	FID
0	1m 03s	30m 58s	311.5757
17	10h 17m 04s	26m 11s	212.6930
33	20h 37m 49s	26m 08s	140.5475
49	30h 48m 44s	26m 13s	57.1253

Fonte: Produção do Autor

Pode-se notar que o resultado do cálculo do FID é mais alto no *tick* 0, que ocorreu no primeiro minuto, em comparação à pontuação calculada no *tick* 49, que ocorreu no final do treinamento, indicando que houve uma melhora de aproximadamente 6 vezes na qualidade das imagens geradas pela rede no final do treinamento. A evolução deste indicador pode ser visualizada no gráfico apresentado na Figura 16.

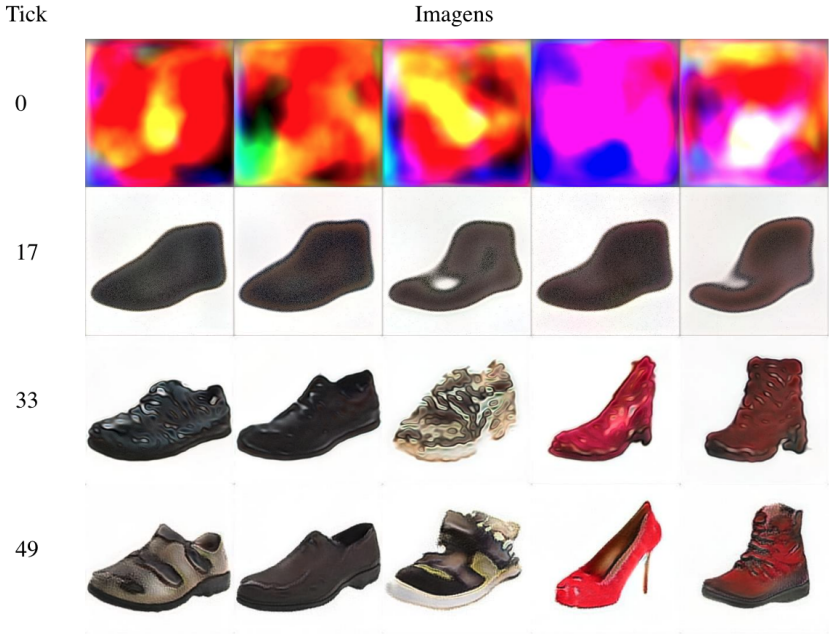
Figura 16: Gráfico de evolução do FID



Fonte: Produção do Autor

Na Figura 17, foram geradas imagens baseadas em *snapshots* tirados no momento de cada cálculo de FID mostrado na Tabela 1. Cada linha na imagem corresponde a uma linha na tabela, permitindo a visualização da qualidade das imagens como demonstrada pelo cálculo do FID. Cada coluna representa a imagem de um calçado gerada com base na mesma *seed*, ou seja, possuindo a mesma entrada no espaço latente em diferentes estágios do treinamento da rede.

Figura 17: Evolução das imagens geradas



Fonte: Produção do Autor

Como se pode notar, no *tick* 0, as imagens não possuem nenhuma forma clara. Todas as imagens geradas são compostas de borrões coloridos distribuídos de maneira indefinida. Já no *tick* 17, após 10 horas de treinamento, a rede aprendeu o formato em alto nível de um calçado, porém sem muita variedade, sendo que todos os calçados gerados possuem uma cor muito similar, e os contornos possuem uma forma genérica.

Do *tick* 17 para o 33 já note-se uma grande mudança em relação à variedade dos calçados gerados. Nas amostras produzidas, percebe-se que há uma variação de cor entre os calçados, além de uma variação de formato e gênero. Percebe-se, também, que a rede neste estado gera calçados com uma indefinição na parte traseira, como é possível observar na Figura 18, onde os sapatos selecionados apresentam uma leve concavidade na região do calcanhar.

Figura 18: Tendência de gerar sapatos com salto

Fonte: Produção do Autor

Ainda na Figura 18, percebe-se também que os sapatos possuem algumas manchas parecidas com bolhas, além de não possuírem detalhes finos como cadarços ou o encaixe para o pé. No *tick* 33, porém, o encaixe para os pés e cadarços já aparecem, assim como as manchas diminuem e a tendência de criar modelos com concavidade no calcanhar parece ser mitigada, como se pode ver na Figura 19, onde os sapatos gerados possuem mais definidamente a presença ou não de salto. Nota-se também, mais distintamente, qual é o modelo de cada calçado gerado, sendo o primeiro um tênis casual, o segundo um *Scarpin*, e o terceiro uma bota casual.

Figura 19: Sapatos com detalhes mais aparentes

Fonte: Produção do Autor

Apesar do progresso aparente nas imagens, ainda percebe-se muitas falhas nas imagens geradas pela rede baseada no *snapshot* mais recente. Porém, ao se observar o gráfico da Figura 16, nota-se uma tendência até então linear no progresso do FID, dando um indício de que a qualidade das imagens geradas tende a melhorar com mais tempo de treinamento, possivelmente diminuindo ainda mais as manchas nas imagens geradas.

Destaca-se ainda, que os dados qualitativos apresentados na Figura 17 corroboram com os dados quantitativos apresentados na Tabela 1, em que pode-se notar que houve uma melhora na qualidade das imagens geradas em cada um dos *ticks* apresentados em ambas as avaliações. Logo, pode-se inferir que a métrica quantitativa utilizada resulta em um bom indicador de avaliação.

9 CONSIDERAÇÕES FINAIS

Este trabalho compreende, tal como Colton, Wiggins et al. (2012), que a criatividade computacional é uma area da computação, que se utiliza de técnicas de Inteligência Artificial, na qual se constrói e trabalha com sistemas computacionais que criam artefatos.

Neste mesmo sentido, a pesquisa desenvolvida e apresentada neste volume buscou investigar um processo que seja capaz de criar imagens bidimensionais de modelos de calçados por meio do uso de Redes Neurais Artificiais.

Quanto à estrutura deste trabalho, foram apresentadas técnicas de geração e avaliação de imagens, como visto no Capítulo 2. Posteriormente, no Capítulo 3, foi feita uma introdução ao conceito de Redes Neurais artificias e *Deep Learning*, assim como à arquitetura GAN. Nos capítulos seguintes (Capítulo 4, Capítulo 5 e Capítulo 6), foram apresentados os trabalhos que resultaram na criação da arquitetura Stylegan2 por Karras et al. (2020a), que foi a utilizada para o desenvolvimento deste projeto. No Capítulo 7 foi apresentado o processo definido para a geração das imagens de calçados, assim como a execução deste processo. Por fim, no Capítulo 8, pudemos analisar os resultados da execução do processo definido, bem como a avaliação destes resultados.

Os objetivos específicos foram alcançados e serão apresentados a seguir. Inicialmente foi feita uma investigação das arquiteturas existentes de GANs, sendo escolhida a Stylegan. Para treinar o modelo de maneira otimizada, foi alugada uma máquina virtual dedicada para processos de *Machine Learning*. A formação de um *dataset* customizado não foi necessária, pois já existia um *dataset* consolidado, o qual foi utilizado. Este *dataset* foi, então, adaptado para o formato de entrada esperado pela arquitetura escolhida e para o treinamento. A avaliação dos resultados foi feita usando a métrica FID, além de uma análise qualitativa.

Pode-se concluir que o objetivo geral deste trabalho, que era gerar imagens bidimensionais de calçados, foi atingido. Mesmo que as imagens ainda possuam algumas falhas, muitos dos resultados gerados tem uma boa qualidade e podem servir de inspiração para a geração de calçados reais.

Como limitações, pode-se mencionar, por exemplo, o fato de o treinamento ter sido executado numa máquina virtual, com os custos que foram descritos no Capítulo 7, fazendo com que o tempo de treinamento fosse limitado pelo orçamento, o que consequentemente, limitou também a qualidade das imagens geradas.

Como trabalho futuro, espera-se empregar uma estatística com teste de hipótese para avaliar se as diferenças entre os FIDs em diferentes tempos de treinamento são significativas. Também propõe-se como trabalho futuro, o treinamento da rede por um período de tempo maior, para que a qualidade das imagens geradas seja superior, pois através de uma análise qualitativa, foi perceptível que há um potencial de melhora na qualidade dos resultados com mais tempo de treinamento.

Por fim, propõe-se um estudo do espaço latente descoberto, de maneira a manipular tal espaço, com o intuito de gerar imagens de sapatos com as características desejadas. Também espera-se gerar a interpolação deste espaço, a fim de melhorar a compreensão das características descobertas pela rede.

Assim, a pesquisa proposta foi concluída com os objetivos atingidos. Destaca-se como contribuição deste trabalho a criação e execução de um processo para a geração de imagens bidimensionais de calçados. Ademais, a jornada de pesquisa dentro do tema da computação criativa pode ser replicada em outros trabalhos do mesmo tema.

REFERÊNCIAS

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow-.org/>. Acesso em: 07 dec. 2020. Citado na página 32.

ALQAHTANI, H.; KAVAKLI-THORNE, M.; KUMAR, G. Applications of generative adversarial networks (gans): An updated review. *Archives of Computational Methods in Engineering*, Springer, p. 1–28, 2019. Citado 2 vezes nas páginas 21 e 22.

BORJI, A. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, Elsevier, v. 179, p. 41–65, 2019. Citado 3 vezes nas páginas 15, 16 e 18.

CAMARGO, M. A. GeraÇÃo automatizada de imagens em pixel art utilizando redes neurais. 2019. Citado na página 11.

CHETLUR, S. et al. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014. Citado na página 33.

COLTON, S.; WIGGINS, G. A. et al. Computational creativity: The final frontier? In: MONTPELIER. *Ecai*. [S.l.], 2012. v. 12, p. 21–26. Citado 2 vezes nas páginas 10 e 42.

DENTON, E. L. et al. Deep generative image models using a laplacian pyramid of adversarial networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 1486–1494. Citado na página 18.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>. Acesso em: 07 dec. 2020. Citado na página 19.

GOODFELLOW, I. et al. Generative adversarial nets. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 2672–2680. Citado 6 vezes nas páginas 10, 11, 15, 20, 21 e 22.

GREUTER, S. et al. Real-time procedural generation ofpseudo infinite'cities. In: *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*. [S.l.: s.n.], 2003. p. 87–ff. Citado na página 14.

HEUSEL, M. et al. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017. Disponível em: <http://arxiv.org/abs/1706-.08500>. Acesso em: 07 dec. 2020. Citado na página 17.

HUI, J. *GAN - StyleGAN and StyleGAN2*. 2020. Disponível em: <https://medium-.com/@jonathan_hui/gan-stylegan-stylegan2-479bdf256299>. Acesso em: 07 dec. 2020. Citado 2 vezes nas páginas 25 e 26.

JIN, Y. et al. *Towards the Automatic Anime Characters Creation with Generative Adversarial Networks*. 2017. Citado na página 17.

KARRAS, T. et al. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. Citado 6 vezes nas páginas 11, 12, 13, 22, 23 e 24.

KARRAS, T.; LAINE, S.; AILA, T. A style-based generator architecture for generative adversarial networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 4401–4410. Citado 4 vezes nas páginas 13, 24, 25 e 26.

KARRAS, T. et al. Analyzing and improving the image quality of stylegan. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 8110–8119. Citado 4 vezes nas páginas 13, 27, 28 e 42.

KARRAS, T. et al. *StyleGAN2 — Official TensorFlow Implementation*. GitHub, 2020. Disponível em: <https://github.com/NVlabs/stylegan2>. Acesso em: 07 dec. 2020. Citado 3 vezes nas páginas 32, 35 e 50.

KELLY, G.; MCCABE, H. Citygen: An interactive system for procedural city generation. In: *Fifth International Conference on Game Design and Technology*. [S.l.: s.n.], 2007. p. 8–16. Citado na página 14.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Citado 3 vezes nas páginas 15, 19 e 20.

LOLLER-ANDERSEN, M.; GAMBÄCK, B. Deep learning-based poetry generation given visual input. In: *ICCC*. [S.l.: s.n.], 2018. p. 240–247. Citado na página 10.

METZ, L. et al. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016. Citado 2 vezes nas páginas 20 e 22.

NIELSEN, M. A. Neural networks and deep learning. 2015. *URL http://neuralnetworksanddeeplearning.com*, v. 62, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com>. Acesso em: 07 dec. 2020. Citado na página 20.

NVIDIA; VINGELMANN, P.; FITZEK, F. H. *CUDA, release: 10.2.89*. 2020. Disponível em: <https://developer.nvidia.com/cuda-toolkit>. Acesso em: 07 dec. 2020. Citado na página 32.

PAPERSPACE. *Paperspace - An IaaS provider*. 2020. Disponível em: <https://www.paperspace/>. Acesso em: 02 nov. 2020. Citado 2 vezes nas páginas 33 e 34.

RODRIGUES, H. F.; SALOMÃO, I. C. O setor calçadista do vale do sinos (rs) no âmbito do mercosul: desafios e potencialidades. *Cadernos de Campo: Revista de Ciências Sociais*, n. 24, p. 169–186, 2018. Citado na página 12.

SALIMANS, T. et al. Improved techniques for training gans. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2016. p. 2234–2242. Citado na página 16.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015. Citado na página 14.

THEIS, L.; OORD, A. v. d.; BETHGE, M. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015. Citado na página 16.

TOIVONEN, H.; GROSS, O. Data mining and machine learning in computational creativity. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 5, n. 6, p. 265–275, 2015. Citado na página 10.

WINKLER, F. *Image Representation Learning with Generative Adversarial Networks*. Tese (Doutorado) — Hochschule für Angewandte Wissenschaften Hamburg, 2018. Citado na página 21.

YU, A.; GRAUMAN, K. Fine-grained visual comparisons with local learning. In: *Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2014. Citado na página 30.

10 APÊNDICES

10.1 APÊNDICE A

Os cabeçalhos de cada uma das colunas estão descritos abaixo:

- Tick: Contagem do *tick*, sendo o *tick* o intervalo no qual um *snapshop* da rede é salvo
- kimg: Quantidade de imagens alimentadas para a rede sendo que 1 kimg = 1000 imagens
- Tempo: Tempo de execução do treinamento até o *tick* em questão
- seg/tick: Tempo de duração em segundo do *tick* em questão
- seg/kimg: Temp ode de duranção em segundo do consumo de kimg
- Manutenção: Tempo gasto em manutenção da rede

Tabela 2: Resultados da primeira execução

Tick	kimg	Tempo	seg/tick	seg/kimg	Manutenção
0	10000.1	1m 03s	62.7	489.97	0.0
1	10008.2	1h 08m 43s	2161.8	268.08	1898.2
2	10016.3	1h 45m 10s	2166.4	268.66	20.7
3	10024.3	2h 21m 38s	2167.1	268.73	20.7
4	10032.4	2h 58m 05s	2166.5	268.67	20.6
5	10040.4	3h 34m 29s	2163.3	268.27	21.0
6	10048.5	4h 10m 57s	2167.4	268.78	20.6
7	10056.6	4h 47m 25s	2167.4	268.78	20.6
8	10064.6	5h 23m 53s	2167.3	268.77	20.6
9	10072.7	6h 00m 19s	2164.9	268.47	21.1
10	10080.8	6h 36m 47s	2167.5	268.79	20.7
11	10088.8	7h 13m 15s	2166.9	268.72	20.7
12	10096.9	7h 49m 45s	2169.1	268.98	20.9
13	10105.0	8h 26m 17s	2171.0	269.22	21.4
14	10113.0	9h 02m 56s	2177.6	270.04	21.0
15	10121.1	9h 39m 30s	2173.1	269.49	20.9
16	10129.2	10h 16m 03s	2172.3	269.39	20.9

Fonte: Produção do Autor

10.2 APÊNDICE B

Os cabeçalhos de cada uma das colunas estão descritos abaixo:

- Tick: Contagem do *tick*, sendo o *tick* o intervalo no qual um *snapshop* da rede é salvo
- kimg: Quantidade de imagens alimentadas para a rede sendo que 1 kimg = 1000 imagens
- Tempo: Tempo de execução do treinamento até o *tick* em questão
- seg/tick: Tempo de duração em segundo do *tick* em questão
- seg/kimg: Temp ode de duranção em segundo do consumo de kimg
- Manutenção: Tempo gasto em manutenção da rede

Tabela 3: Resultados da segunda execução

Tick	kimg	Tempo	seg/tick	seg/kimg	Manutenção
0	10096.1	1m 01s	60.8	475.27	0.0
1	10104.2	1h 04m 00s	2167.6	268.80	1611.2
2	10112.3	1h 40m 32s	2171.5	269.28	20.7
3	10120.3	2h 19m 25s	2311.5	286.64	21.3
4	10128.4	2h 58m 17s	2310.6	286.54	21.7
5	10136.4	3h 37m 07s	2307.9	286.19	22.1
6	10144.5	4h 16m 00s	2311.2	286.61	21.5
7	10152.6	4h 52m 35s	2174.3	269.63	20.9
8	10160.6	5h 29m 10s	2174.5	269.65	20.5
9	10168.7	6h 05m 53s	2180.2	270.36	22.7
10	10176.8	6h 42m 32s	2177.0	269.96	22.1
11	10184.8	7h 19m 05s	2172.1	269.36	20.7
12	10192.9	7h 55m 37s	2172.2	269.37	20.6
13	10201.0	8h 32m 08s	2169.4	269.02	21.2
14	10209.0	9h 08m 41s	2172.1	269.36	20.7
15	10217.1	9h 45m 14s	2172.4	269.39	20.6
16	10225.2	10h 21m 46	2171.8	269.32	20.7
17	10233.2	11h 24m 25	2168.6	268.93	1589.6
18	10241.3	12h 00m 58	2172.3	269.39	20.7
19	10249.3	12h 37m 30	2172.4	269.39	20.6
20	10257.4	13h 14m 04	2172.5	269.41	20.6
21	10265.5	13h 50m 34	2169.6	269.05	21.3
22	10273.5	14h 27m 08	2172.3	269.39	20.7
23	10281.6	15h 03m 40	2172.1	269.36	20.6
24	10289.7	15h 40m 13	2172.1	269.35	20.7
25	10297.7	16h 16m 44	2169.4	269.02	21.2
26	10305.8	16h 53m 17	2172.3	269.38	20.6
27	10313.9	17h 29m 50	2172.6	269.42	20.8
28	10321.9	18h 06m 23	2172.3	269.39	20.8
29	10330.0	18h 42m 53	2169.2	269.00	21.2
30	10338.0	19h 19m 28	2173.9	269.59	20.8
31	10346.1	19h 56m 06	2176.5	269.90	20.9
32	10354.2	20h 32m 41	2174.1	269.60	20.9
33	10362.2	21h 35m 24	2169.0	268.98	1594.6
34	10370.3	22h 11m 57	2172.2	269.37	20.7
35	10378.4	22h 48m 30	2171.9	269.33	20.6
36	10386.4	23h 25m 02	2171.7	269.31	20.7
37	10394.5	1d 00h 01m	2169.1	268.99	21.2
38	10402.6	1d 00h 38m	2174.6	269.66	20.7
39	10410.6	1d 01h 14m	2172.9	269.46	21.0

Fonte: Produção do Autor

10.3 APÊNDICE C

Este apêndice contém um compilado com as imagens geradas pela rede no último *snapshot*.

Figura 20: Imagens geradas

Fonte: (KARRAS et al., 2020b)