UNIVERSIDADE FEEVALE

MATHEUS ADAMS CAMARGO

GERAÇÃO AUTOMATIZADA DE IMAGENS EM *PIXEL ART* UTILIZANDO REDES NEURAIS

Novo Hamburgo

2019

MATHEUS ADAMS CAMARGO

GERAÇÃO AUTOMATIZADA DE IMAGENS EM *PIXEL ART* UTILIZANDO REDES NEURAIS

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

2019

MATHEUS ADAMS CAMARGO

GERAÇÃO AUTOMATIZADA DE IMAGENS EM *PIXEL ART* UTILIZANDO REDES NEURAIS

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
2019

AGRADECIMENTOS

Gostaria de registrar minha gratidão àqueles que contribuíram de alguma forma para a realização desse trabalho de conclusão, em especial:

Primeiramente ao Criador, sem fé em mim mesmo nada disso seria possível;

Ao meu orientador, João Batista Mossmann, que com sua experiência e organização prestou todo apoio na condução deste projeto;

A minha família, que esteve ao meu lado durante todo o processo, ao Lucas especialmente pelas ideias e dicas valiosas;

A todos que despertaram e mantiveram em mim a paixão pela ciência, pela computação e pela arte;

Muito obrigado a todos!

RESUMO

Neste projeto é abordada a relação da engenharia de software com a indústria de jogos, especificamente no contexto do desenvolvimento de imagens para jogos 2D que utilizam como estilo o *pixel art*. Embora existam várias ferramentas que auxiliam na construção de imagens bidimensionais, ainda há uma demanda grande da capacidade criativa do artista no que diz respeito ao primeiro estágio de concepção de uma figura. Um exemplo dessa dificuldade inicial ocorre ao desenhar novas poses para personagens existentes. Nesse sentido, nota-se que redes neurais têm sido usadas para resolver vários problemas envolvendo geração de imagens, sendo possível estudar e desenvolver uma forma de simplificar o desenvolvimento de novas posições para personagens utilizando este recurso computacional. A partir desse propósito, este trabalho tem como objetivo principal investigar, propor e implementar um processo de geração de imagens bidimensionais por meio de uma rede neural, aplicado no desenvolvimento de jogos com personagens em *pixel art*. O método para atingir essa proposta inicia com um estudo teórico das tecnologias disponíveis e das formas de integrá-las. A partir dessa investigação, é definido um processo gerador de imagens em *pixel art* utilizando redes neurais. Para validar essa proposta, o processo foi construído na prática e foram estabelecidas duas técnicas de comparação de imagens para verificar o grau de semelhança das imagens geradas em relação ao resultado ideal esperado. O processo se mostrou capaz de gerar imagens semelhantes ao esperado, com características de *pixel art*. Os resultados são promissores como versão inicial de imagem sob a qual o artista pode efetuar o desenvolvimento, e o processo definido poderá ser reproduzido em outros estudos a partir das diretivas que foram apresentadas.

Palavras-chave: Estimativa de pose humana. Geração de imagens. *Pixel art*. Redes neurais. Tradução imagem-para-imagem.

ABSTRACT

In this project the relationship between software engineering and the gaming industry is discussed, specifically in the context of images for 2D games that choose pixel art as aesthetics. Although there are several tools that help in the construction of two-dimensional images, there is still a great demand for the creative capacity of the artist, in particular regarding the initial stage of conception of a figure. An example of this initial difficulty occurs when drawing new poses for existing characters. In this sense, it is well-known that neural networks have been used to solve several problems involving image generation, therefore being possible to study and develop a way to simplify the development of new positions for characters using this computational resource. Thus, this work aims to investigate, propose and implement a two-dimensional image generation process through a neural network, applied in the development of games with pixel art characters. The method to achieve this proposal begins with a theoretical study of available technologies and ways of integrating them. Through this investigation, a pixel art image generator process is defined using neural networks. To validate this proposal, the process was built in practice and two image comparison techniques were established to verify the similarity between generated images and the expected ideal result. The process proved itself to be able to generate images close to the expected ones, with pixel art characteristics. The results are promising as an initial version of the image under which the artist can make adjusts, and the defined process can be reproduced in other studies from the directives that were presented.

Keywords: Human pose estimation. Image generation. Image-to-image translation. Neural networks. Pixel art.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RGB	*Red-Green-Blue*
SMC	*Simple Matching Coefficient*
RNA	Rede Neural Artificial
cGAN	*Conditional Generative Adversarial Networks*
CNN	*Convolutional Neural Network*
GAN	*Generative Adversarial Network*
CPM	*Convolutional Pose Machine*
GEE	*Generalized Equations Estimating*

LISTA DE EQUAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A engenharia de software é uma área abrangente da computação que define processos e ferramentas envolvidos na criação de um software que ofereça soluções para problemas do cotidiano que podem ser interpretados por sistemas digitais. Dentro dessa área, o processo de desenvolvimento de software é caracterizado por criar um produto imaterial, sendo mais semelhante a um serviço. Essa alta abstração e a existência de diversas linguagens de programação e padrões de projeto contribuem para que o desenvolvimento de software seja aplicado em diversas áreas, como por exemplo medicina, gerenciamento de empresas ou desenvolvimento de jogos digitais. (PRESSMAN; MAXIM, 2016)

Especificamente na indústria de jogos, é comum as equipes de desenvolvimento serem compostas por profissionais de diferentes áreas. A integração destas equipes costuma ser uma tarefa desafiadora devido às características próprias de trabalho de cada setor (FREITAS *et al.*, 2017). Enquanto que os responsáveis pelo desenvolvimento de software se preocupam com as tarefas de programação e automatização de processos, os artistas que cuidam do design e da parte gráfica dos jogos ainda passam por muitos processos criativos executados de forma manual ao desenvolver conteúdo.

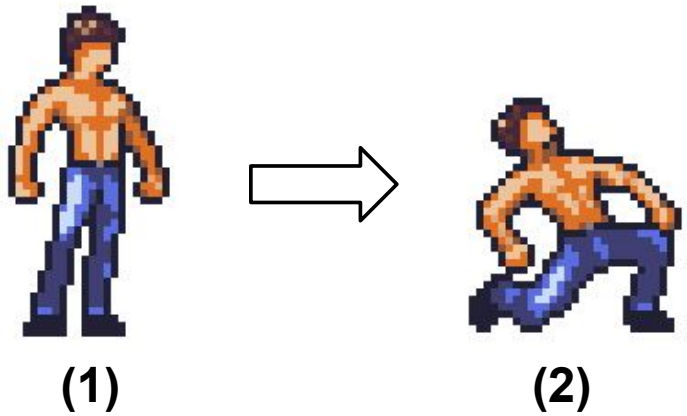
Embora neste campo da arte gráfica digital existam várias ferramentas que auxiliem na construção de imagens bidimensionais, como softwares ou técnicas de desenho, existe uma demanda grande da capacidade criativa do artista, que deve considerar questões como iluminação, movimento e composição. No livro *Creating the Art of the Game* (OMERNICK, 2004), o autor destaca que essa capacidade de imaginar e criar é o que diferencia um artista de outros profissionais. Segundo ele, um bom artista é aquele que consegue agregar de forma convincente a fantasia com os elementos pertinentes da realidade ao desenvolver seu estilo (OMERNICK, 2004).

Como visto, o artista pode adotar vários estilos visuais ao desenvolver seu trabalho, utilizando por exemplo desenhos em 2D, desenhos vetoriais, *pixel art* ou modelos em 3D. Neste projeto, será abordado em específico o *pixel art*, que é caracterizado por utilizar baixas resoluções para representar formas complexas e ter uma paleta de cores reduzida, exigindo a capacidade de síntese de informação (KUO; YANG; CHU, 2016). Além disso, este trabalho terá como contexto as animações em *pixel art*, onde é necessário gerar vários quadros do mesmo personagem em diferentes posições. Para o desenvolvimento de personagens animados, os conceitos citados anteriormente sobre a construção de ilustrações também se aplicam: o personagem possui uma anatomia, proporções e iluminação que devem ser respeitados para gerar realismo.

Neste contexto das animações em 2D, o artista frequentemente precisa criar novas

posições corporais para os personagens. Diferente do que ocorre em projetos feitos em 3D, esta é uma tarefa complexa, pois não é possível simplesmente manipular um modelo tridimensional e obter a imagem desejada. Para fazer isso em imagens bidimensionais, normalmente o artista fará um esboço inicial à mão livre da nova posição desejada, gerando apenas um esqueleto da posição. Depois, tendo o referencial do personagem em mente, adiciona as proporções corretas em cada parte do corpo e vai aprimorando os detalhes até que fique suficientemente coerente com as demais imagens do personagem. Como foi visto, esse processo de desenho é trabalhoso em todas as etapas, porém, no estágio inicial há mais incerteza das formas a desenhar e com isso a chance de o artista se equivocar é maior. Pensando nisso, este projeto se propõe a estudar e desenvolver uma forma de simplificar o desenvolvimento de novas posições para personagens em *pixel art*, utilizando redes neurais. Tomando como exemplo a Figura 1, o personagem na imagem (1) está ereto. Para criar a posição vista na imagem (2), em que o personagem está abaixado e inclinado para trás, seria necessário desenhá-la manualmente desde o início. Com este projeto, deseja-se ser capaz de gerar automaticamente uma versão inicial da imagem (2) com base em um treinamento prévio que assimile as características da imagem (1). Dessa forma, o artista poderá trabalhar já partindo de um estágio mais avançado da produção da imagem.

Figura 1: Exemplo de mudança da pose de um personagem



Fonte: elaborado pelo autor, com base na arte “Hero spritesheets (Ars Notoria)” (OpenGameArt.org)

Tal como citado, uma das abordagens possíveis para geração automatizada de imagens é o uso de redes neurais, como pode ser visto no artigo *Image-to-Image Translation with Conditional Adversarial Networks* (ISOLA *et al.*, 2017). Neste projeto, os autores aplicaram redes neurais para definir um processo de transformação de imagens: a partir de um treinamento prévio que vincula uma série de entradas e saídas conhecidas, a rede aprende a gerar saídas para novas entradas. Essa lógica já foi usada para várias aplicações, como por exemplo gerar fachadas de prédios ou modelos de bolsas ou sapatos realistas a partir de um desenho simples. Outro uso de redes neurais para geração de imagens advém

da empresa NVIDIA, que criou uma rede neural capaz de combinar de forma parametrizável a face de duas pessoas quaisquer, gerando fotos realistas (KARRAS; LAINE; AILA, 2018).

Além dos exemplos citados, outra pesquisa que se relaciona com o objetivo deste projeto é a transformação de pose em vídeos demonstrada no artigo *Everybody Dance Now* (CHAN *et al.*, 2018). Neste caso, a rede neural foi treinada para ler dois vídeos: o primeiro, com um dançarino executando seus passos; e o segundo, com um sujeito qualquer se movimentando livremente. A rede neural desenvolvida neste projeto consegue produzir um terceiro vídeo, onde se vê os passos do dançarino, porém executados pelo sujeito do segundo vídeo. Ou seja, a rede aprende a transferir somente os movimentos de um vídeo para o outro, semelhante ao que se pretende fazer neste trabalho.

Tendo em vista essas aplicações de redes neurais para geração de imagens, observa-se que o segmento do *pixel art* escolhido como tema deste trabalho exige atenção especial em relação às imagens geradas: a resolução da imagem é reduzida e a paleta de cores deve ser igual a da imagem original para manter a coerência. Atendendo a estes critérios, pretende-se simplificar o processo criativo do artista, disponibilizando uma forma de gerar rapidamente novas poses de personagens. Reduzindo esse esforço inicial para conceber a forma geral do personagem, é possível trabalhar diretamente nos detalhes e aperfeiçoamento do desenho. O artista poderá também fazer testes de posições, visualizando rapidamente um resultado parcial, que permitirá melhorar a pose antes de investir tempo no desenho final. Com isso, tem-se como objetivo principal deste projeto desenvolver um processo de geração de imagens, utilizando rede neural, que esteja alinhado com as necessidades específicas do contexto da criação de personagens em *pixel art*.

Nos próximos capítulos, serão retomados e abordados em mais detalhes os conceitos que se relacionam com este estudo. Para apresentar esta proposta, este trabalho está organizado da seguinte forma: nos capítulos 2 até 4 são abordados os conceitos aplicados neste projeto, detalhando características de imagens em *pixel art* e das redes neurais. A seguir, nos capítulos 5 e 6 são discutidas implementações práticas destes conceitos, para que no capítulo 7 seja definida a proposta principal do trabalho. Os capítulos 8 e 9 abordam a implementação prática e análise de resultados. Por fim, o capítulo 10 apresenta a conclusão.

2 APLICAÇÃO DE IMAGENS NO CONTEXTO DIGITAL

Neste capítulo serão apresentados conceitos que se relacionam com a área do desenvolvimento de jogos. Na seção 2.1 este projeto será contextualizado, abordado de forma mais detalhada este processo de desenvolvimento e a problemática envolvida em comparação com outras áreas da computação. Enquanto isso, na seção 2.2 são vistos detalhes sobre o desenvolvimento da parte visual dos jogos, relacionando com os conceitos apresentados anteriormente e destacando os processos que se pretende simplificar com este projeto.

2.1 PROCESSO DE DESENVOLVIMENTO EM *GAME DESIGN*

A engenharia de software busca resolver problemas do cotidiano usando meios computacionais (PRESSMAN; MAXIM, 2016). No contexto do desenvolvimento de jogos digitais, assim como em outras áreas de negócio que se fundamentam na computação, a questão final a ser resolvida é de que forma obter um retorno financeiro através desses meios digitais. A abordagem escolhida para resolver este problema utilizando jogos normalmente envolve gerar algum tipo de satisfação no usuário, de forma que ele se sinta motivado a comprar o jogo ou investir seu tempo jogando. Se o usuário estiver suficientemente motivado, vai optar por investir seu dinheiro com um jogo específico e abrir mão de outros jogos da concorrência.

Essa fidelização do usuário pode ser gerada de diversas formas, como por exemplo trabalhando as sensações que o jogador experimenta durante a partida ou utilizando sistemas de recompensa que gerem satisfação quando o jogador completa objetivos (CRAWFORD, 2012). Outra técnica que torna o jogo mais atrativo e que está sempre presente em algum grau diz respeito ao visual do jogo, que deve ser coerente com o tipo de público-alvo para gerar a identificação do jogador com o game. Para trabalhar essas questões e organizar o processo de criação de um jogo, a equipe de desenvolvimento deve aplicar conceitos de *game design*. (SCHELL, 2008)

Qualquer processo de *design* é caracterizado pela tomada de decisões, ou seja, levantando diversas opções e escolhendo aquela que leva mais próximo ao objetivo do produto desenvolvido e melhor se encaixa nas restrições estabelecidas. Conforme visto anteriormente, neste contexto do desenvolvimento de jogos o objetivo é criar uma experiência valiosa ao jogador, portanto o *game design* define métodos para avaliar o quanto desse valor um determinado aspecto do jogo será capaz de entregar. (SCHELL, 2008)

Segundo Cook (2019), a equipe de desenvolvimento era tradicionalmente reduzida e fazia as escolhas de design baseadas nas experiências próprias de cada um, ou seja, com base no conhecimento empírico dos elementos que geram satisfação. Além disso, o

processo se caracterizava por ter etapas bem demarcadas, passando pelo processo de desenvolvimento, depois executando os testes e então a liberação do jogo. Ao atingir essa etapa de testes, dificilmente haveria grandes mudanças nas características visuais ou de jogabilidade. Por fim, só ao liberar o jogo é que se descobria se ele seria um sucesso ou fracasso. Essa abordagem se assemelha muito a um modelo em cascata, e funcionava bem para criar jogos semelhantes aos já existentes, que contivessem pequenas mudanças nas mecânicas ou visual, mas que teriam chances de sucesso parecidas. Apesar de adequada para o cenário de equipes de desenvolvimento menores, essa sistemática acabava desfavorecendo a inovação, não promovia uma melhora contínua do jogo e gerava garantia de sucesso baseada em critérios próprios de cada desenvolvedor. (COOK, 2019)

Com a evolução tecnológica, surgiram novas ferramentas e modelos de desenvolvimento para a construção de jogos, além de estudos que identificaram com mais clareza quais elementos geram satisfação no jogador. As equipes se tornaram cada vez maiores e multidisciplinares para conseguir trabalhar com a complexidade crescente do desenvolvimento. Há profissionais responsáveis por desenvolver somente os personagens, os efeitos visuais, a interface, técnicos para pensar na jogabilidade, sonoplastas, programadores e animadores. Cada um destes grupos realiza a construção de um produto específico, mas que tem o mesmo objetivo de cativar o jogador de alguma forma. Para dar conta dessa realidade, o processo de desenvolvimento também se modernizou e atualmente contém muitas características de metodologias ágeis. Ao invés de etapas delimitadas, o processo como um todo se baseia em ciclos de testes prévios e ajustes, garantindo que eventuais falhas de concepção sejam eliminadas antes de comprometer a qualidade do jogo entregue. Pensando por exemplo no processo da interface de usuário, a equipe vai definir quais são as informações mais importantes e como exibi-las e organizá-las de modo geral. Depois, são criados elementos cada vez mais específicos da interface, identificando a cada rodada de desenvolvimento se foi mantido o padrão entre todas as telas do jogo.

Com relação aos elementos que cativam o jogador, o conhecimento empírico foi substituído pelo entendimento dos mecanismos de recompensa do cérebro humano. Além disso passou-se a utilizar métricas que sejam capazes de quantificar o envolvimento de um jogador, como por exemplo o total de minutos jogando ou quantidade de vezes que determinada funcionalidade foi utilizada durante os testes. A partir da análise de impacto das mudanças sobre métricas como essas, a equipe consegue prever de forma mais segura quais abordagens vão ajudar a melhorar o resultado e quais elementos podem estar quebrando o balanço do jogo. (FREITAS *et al.*, 2017)

Como foi visto, o processo se tornou mais iterativo, focando em técnicas que permitem agilizar a descoberta de conhecimento, fazendo testes prévios e adequando o desenvolvimento antes que o custo da mudança se torne alto. Dessa forma, a complexidade atual do desenvolvimento é quebrada em etapas menores que podem ser aprimoradas in-

dividualmente. Este projeto busca portanto aprimorar o uso destas técnicas inseridas no processo de desenvolvimento da parte artística do jogo, que é abordado na próxima seção.

2.2 ASPECTOS ARTÍSTICOS DE IMAGENS DIGITAIS

Como foi visto anteriormente, o jogo é constituído por artefatos que cativam o jogador de alguma forma. Um dos primeiros contatos que há com o jogo e que também é o aspecto mais evidente do *design* durante o *gameplay* é sem dúvidas o aspecto visual. É através da imagem que são transmitidos conceitos importantes do jogo, como por exemplo o público-alvo, gênero e efetivamente o que está ocorrendo durante a partida. Segundo Schell (2008), há quatro elementos básicos que constituem um jogo: estética, mecânica, história e tecnologia. Cada um destes agrega diferentes valores ao jogo e se relaciona com os demais, podendo ser interpretado tanto como início de raciocínio ou como detalhe a ser considerado ao desenvolver algum elemento. Por exemplo, ao desenvolver as mecânicas, é necessário considerar de que forma elas se relacionam com o visual ou com a história do jogo. Como nota-se, uma aparência bem planejada e adequada ao público-alvo é um dos pilares para construir um jogo de sucesso.(SCHELL, 2008)

Na seção anterior, verificou-se que o processo de desenvolvimento é baseado em fazer escolhas que levem cada vez mais perto dos objetivos que o jogo pretende atingir, e isso normalmente é feito através de várias iterações para levantar possibilidades e testar as mais efetivas. No campo da arte visual, a mesma estratégia se aplica: os artistas levantam várias possibilidades de desenho para cada aspecto do jogo e vão selecionando as ideias mais promissoras para serem aprimoradas.(FREITAS *et al.*, 2017). Os critérios aplicados geralmente estão relacionados com a necessidade de comunicar ao jogador rapidamente através do visual a história por trás de cada elemento do jogo. Pensando por exemplo no processo de criação de um personagem, a equipe vai levantar quais são as suas características mais importantes, e a partir dessa lista identificar de quais formas elas podem ser representas. Podem ser elencadas diversas possibilidades, mas a equipe vai escolher apenas uma antes de seguir para o desenvolvimento em detalhes, que estará sujeito à mesma sistemática. Nesse sentido, uma das escolhas do artista diz respeito ao estilo de desenho, e neste projeto será abordado o estilo do *pixel art*.

Embora qualquer arte gráfica digital seja intrinsecamente composta por *pixels*, nem todas podem ser consideradas *pixel art*. Segundo Silber (2015), o *pixel art* ocorre quando cada pixel visível da imagem foi colocado intencionalmente pelo artista naquela posição específica. Além disso, normalmente há uma limitação clara da paleta de cores utilizada, pois isso também ajuda a tornar cada pixel mais evidente, em contraste com outros tipos de arte digital que não possuem uma paleta limitada e apresentam uma transição de cores mais suave. Na Figura 2 o personagem é composto por poucas cores e possui baixa resolução, ilustrando as características típicas desse tipo de arte.

Figura 2: Exemplo de imagem em *pixel art* com a respectiva paleta de cores destacada

Fonte: produção do autor, com base na arte “Hero spritesheets (Ars Notoria)” (OpenGameArt.org)

Nas primeiras décadas em que se começou a desenvolver jogos digitais a memória disponível era muito limitada e por isso era importante reduzir tanto a quantidade de informações armazenadas quanto o tamanho dos arquivos de imagem gravados. O *pixel art* se tornou praticamente um consenso neste período pois se mostrou efetivo ao atender simultaneamente essas duas necessidades. Por ser naturalmente desenhado em resoluções menores, já há uma diminuição no tamanho dos arquivos. Além disso, muitos jogos aplicam o uso de paletas de cores. Uma paleta de cores, que também pode ser vista na Figura 3, nada mais é do que um conjunto limitado de cores disponíveis para uso na imagem. Ela permite uma abordagem diferente para efetuar a coloração, que faz diminuir o tamanho dos arquivos e é detalhada a seguir. (SILBER, 2015)

Em uma imagem que não usa paleta, cada pixel tem sua cor descrita por três parâmetros: um para a intensidade do canal de cor vermelha, um para cor verde e o último para cor azul. Ao sobrepor estes três canais, surge a cor desejada. Este sistema denominado RGB, do inglês *Red-Green-Blue*, pode ser representado digitalmente por três bytes: um para cada um dos canais. Cada byte pode representar até 256 valores diferentes, dessa forma cada pixel pode conter milhões de cores distintas devido às combinações possíveis dos três canais. Enquanto isso, imagens que usam paleta de cores tem como objetivo reduzir o tamanho final dos arquivos e como premissa o uso de poucas cores. Para fazer isso, utilizam um arquivo adicional que descreve essa paleta limitada de cores. Neste arquivo ficam elencadas as cores disponíveis, que podem ser descritas em um formato qualquer, como por exemplo o RGB. Além disso, para cada cor há um índice único associado. No arquivo da imagem, ao invés de utilizar três bytes de informação para cada pixel, é apenas indicado qual o índice da paleta que deve ser buscado para colorir aquele pixel. Desta forma, pode ser utilizado apenas um byte de informação para cada pixel e ainda assim ter 256 cores distintas para serem referenciadas na paleta de cores. Isso gera uma diminuição notável no tamanho das imagens, mesmo considerado a existência dessa paleta em separado, pois a mesma paleta pode ser utilizada entre várias imagens distintas.

O uso de paletas gera outra vantagem, que é a possibilidade de recolorir uma imagem inteira sem ter que passar por cada pixel efetuando a substituição: basta alterar a cor na paleta associada e isso refletirá em todas as imagens que utilizam essa paleta. Este exemplo está representado na Figura 3, onde foi alterado para verde as cores da calça do personagem. A imagem deste personagem foi obtida através de um repositório *online* de uso livre (OpenGameArt.org), e será utilizada neste trabalho quando necessário exemplificar imagens em *pixel art*.

Figura 3: Exemplo de *pixel art* com a paleta de cores alterada

Fonte: adaptado de "Hero spritesheets (Ars Notoria)" (OpenGameArt.org)

Contrário ao que se poderia esperar, o estilo de *pixel art* ainda é bastante utilizado na atualidade, mesmo que as limitações técnicas que originalmente motivaram seu uso tenham sido resolvidas e aberto espaço para outras abordagens, como por exemplo a modelagem 3D. Verificou-se que o *pixel art* provém uma estética diferenciada e estabelece uma sensação de nostalgia nos jogadores mais velhos, sendo ambos fatores que aumentam a chance de sucesso comercial de um jogo, como foi visto na seção 2.1. Além disso, em comparação com outras técnicas de desenvolvimento que envolvem criação de modelos em 3D ou vetorização para criar componentes do jogo, o *pixel art* não exige softwares especializados para edição. Isso possibilita não só uma simplificação do processo, pelo fato de trabalhar com arquivos de complexidade menor, como também torna o desenvolvimento mais acessível ao permitir que os técnicos escolham a ferramenta com que preferem desenvolver.(SILBER, 2015)

Os artistas que desenvolvem desenhos em *pixel art* também utilizam como método o processo de *game design* visto na seção 2.1. Ou seja, também fazem a arte de forma iterativa e levantando algumas possibilidades para selecionar as melhores. Por exemplo, se um artista precisa criar uma animação, ele fará um esboço inicial de cada pose necessária e vai fazendo melhorias até que o resultado final esteja satisfatório. Conforme descrito no livro *The Illusion of Life: Disney Animation* de Thomas e Johnston (1981) e ilustrado de forma prática por Becker (2017), o artista pode adotar duas sistemáticas ao desenvolver animações:

1. ***Straight Ahead*:** o artista desenha cada quadro da animação sequencialmente até o final. É adequado para animações mais dinâmicas, pouco previsíveis ou que contenham muitos elementos da Física em ação. Como exemplo de aplicações pode ser imaginada a forma como seria uma fogueira queimando, fumaça se dispersando ou ainda uma roupa balançando.

2. ***Pose to Pose*:** o artista desenha somente os quadros principais do começo até o final da animação, sem muitos detalhes. Depois de estarem coerentes entre si, começa a conectá-los com mais quadros intermediários para só então trabalhar os detalhes de cada quadro. Essa abordagem é a mais indicada para desenvolver o movimento de personagens, pois dá maior controle sobre o andamento da animação do começo até o final e o artista consegue identificar mais cedo se alguma pose não está adequada.

Por fim, o último conceito a ser destacado é referente à terminologia específica desta área de desenvolvimento de jogos. Ao construir gráficos em *pixel art* é comum denominar como *sprite* (do inglês, espírito ou fada) cada quadro da animação de um personagem. Este termo surgiu devido ao fato de que essas imagens, quando utilizadas nos jogos, são renderizadas em separado do cenário ou outras imagens de fundo, como se estivessem flutuando sobre a tela. A partir disso surgiu também o termo *sprite sheet*, que se refere à imagem única que contém todos os *sprites* de um personagem organizados sequencialmente em linhas e colunas. (GUTTAG, 1993)

Com tudo isso, entende-se que o desenvolvimento das imagens no contexto do *pixel art* está fundamentado em um processo iterativo que faz parte da metodologia padrão do *game design*. Além destes fundamentos, o artista também pode aplicar conceitos mais gerais de avaliação de imagens durante o seu processo criativo, que são discutidos no próximo capítulo e embasam as técnicas para avaliação de resultados deste projeto.

3 TÉCNICAS DE AVALIAÇÃO DE IMAGENS

Ao trabalhar com imagens é importante ter definida uma forma de avaliar a qualidade do que está sendo produzido, seja para analisar o resultado final ou para identificar problemas durante o desenvolvimento. Para este projeto que propõe a geração de imagens, isso é essencial para verificar a efetividade da proposta. Uma das formas de avaliar imagens é a comparação com uma referência, e essa abordagem é adequada para este estudo uma vez que é possível gerar novas imagens já tendo um resultado ideal esperado. Quando mais semelhantes forem as imagens geradas e as esperadas, tanto melhor será o processo de geração. Portanto, é necessário fundamentar algumas técnicas que pretende-se aplicar para isso no desenvolvimento deste projeto. Nas próximas seções, serão apresentadas e discutidas formas de fazer comparação entre as imagens, inicialmente apresentando critérios quantitativos e posteriormente alguns métodos qualitativos para extrair informações.

3.1 AVALIAÇÃO QUANTITATIVA COM IMAGEM BINARIZADA

Uma forma de avaliação quantitativa de imagens envolve o uso de conceitos da ciência de dados. O *Simple Matching Coefficient* (SMC, coeficiente simples de correspondência, tradução nossa) permite verificar a similaridade de um parâmetro numérico entre um conjunto de dados e uma referência conhecida, sob a condição de que a informação analisada seja um dado binário. Criada por William M. Rand, essa técnica também é conhecida como coeficiente de Rand e produz um valor padronizado entre todas as comparações, ou seja, é gerado sempre um resultado entre 0 e 1 mesmo que a grandeza da diferença seja variável. A aplicação neste projeto pode ser feita verificando se cada pixel possui ou não alguma característica (informação binária) nas imagens, para então aplicar o cálculo dimensionando essa diferença. (RAND, 1971)

A formulação original desta técnica pode ser observada na equação 3.1, onde o resultado SMC é obtido a partir de quatro parâmetros: FP (falso positivo), FN (falso negativo), VP (verdadeiro positivo) e VN (verdadeiro negativo). Como nota-se, estes quatro parâmetros classificam cada dado de acordo com o acerto ou erro em relação à referência, sendo que o somatório de todos os parâmetros corresponde à quantidade de elementos analisados. Como no dividendo se encontram os acertos e no divisor o total, percebe-se que está fórmula calcula o grau de precisão, sendo 0 o valor que representa o erro total e 1 o valor que indica acerto completo entre a referência e o objeto analisado, sendo possível aplicar esta lógica para diversos contextos. (RAND, 1971)

$$SMC = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.1}$$

Para aplicação neste projeto, deseja-se ter um parâmetro que varie entre 0 e 1 para medir o quão diferentes são as imagens geradas e as imagens esperadas pelo processo. Portanto, foi feita uma alteração nesta fórmula para adequar a esta proposta, além de simplificar algumas características do cálculo para aplicação neste cenário da análise de imagens. Pretende-se criar uma comparação que considere informações pixel-a-pixel nas imagens e sabe-se que as imagens comparadas terão sempre as mesmas dimensões. A partir disso, pode ser visto no SMC adaptado da equação 3.2 que passou a ser considerado como parâmetro as dimensões $R_{largura}$ e R_{altura} das imagens, que corresponde à quantidade de pixels existentes, ou seja, exatamente a quantidade de elementos analisados. Além disso, o dividendo $VP + VN$ deve ser substituído por $FP + FN$ para calcular a diferença ao invés do grau de acerto. Para simplificar essa parte, percebeu-se que não é necessário discriminar os casos de falso positivo e falso negativo separadamente, basta calcular a diferença entre valor do parâmetro escolhido entre as imagens comparadas, deste modo identificando a quantidade de elementos divergentes entre as duas. Isso é expresso nesta fórmula adaptada através dos parâmetros P_g e P_r que representam o valor do parâmetro P na imagem gerada e na imagem de referência, respectivamente. Com isso o $SMC_{adaptado}$ mantém o comportamento de variar exatamente entre 0 e 1, porém sendo 0 o valor que indica igualdade total do parâmetro comparado e o valor 1 denotando que houve diferença no parâmetro para todos os pixels existentes na imagem. Desta forma tem-se um meio de quantificar a diferença entre imagens, sendo necessário ainda definir qual será o parâmetro P comparado entre as imagens.

$$SMC_{adaptado} = \frac{|P_g - P_r|}{R_{largura} \times R_{altura}} \tag{3.2}$$

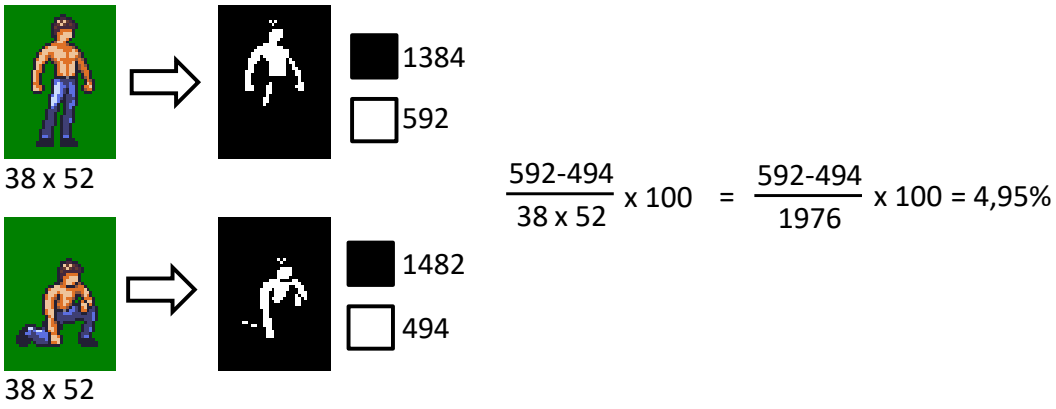
Propõe-se então comparar a imagem em modo binário, ou seja, considerando os pixels somente como pretos ou brancos. Com isso se uma imagem for totalmente escura e a outra completamente clara elas são consideradas diferentes e pontuarão 1 na equação 3.2. Um método para binarizar a imagem consiste em verificar o grau de claridade de cada pixel e aplicar sobre este um limiar para decidir se deve ser transformado em preto ou branco. O nível de luminosidade percebida pelo olho humano pode ser obtido através de uma média ponderada de cada canal de cor RGB. A cor verde, por exemplo, é mais notável do que a cor azul na visão humana, portanto exercendo uma influência maior na luminosidade. A fórmula que faz este processo pode ser vista na equação 3.3, e retorna um valor entre 0 (tom mais escuro) e 255 (tom mais claro). (RIDPATH; CHISHOLM, 2000)

$$Brilho_{rgb} = \frac{(Red \times 30) + (Green \times 59) + (Blue \times 11)}{100} \tag{3.3}$$

Aplicando esta fórmula e um limiar escolhido nas duas imagens a comparar, é

possível gerar uma representação binária composta somente de pixels claros ou escuros, possibilitando a contagem e cálculo da diferença. Com isso, pode ser visto no exemplo da Figura 4 duas imagens comparadas através desta técnica, demonstrando o resultado em termos percentuais. Para fins de exemplo foram colocadas duas imagens nitidamente diferentes, porém em uma situação prática essa comparação é mais adequada quando se sabe que as imagens serão semelhantes, pois neste contexto permite identificar as diferenças que uma avaliação superficial pode não perceber. Para complementar esta técnica, a próxima seção apresenta outra modalidade de comparação com enfoque nas cores da imagem.

Figura 4: Exemplo de comparação de imagens utilizando SMC adaptado



Fonte: elaborado pelo autor

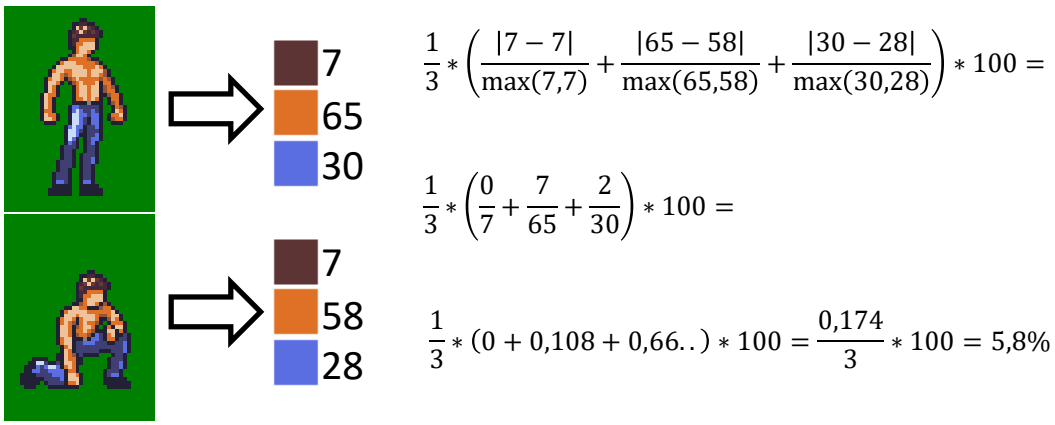
3.2 AVALIAÇÃO QUANTITATIVA DA PALETA DE CORES

A partir do estudo da técnica de comparação da imagem binária, este projeto propõe outra forma de comparação que aplica os mesmos conceitos, porém destinados à comparação da distribuição de cores da imagem. Como viu-se na seção 2.2, as imagens em *pixel art* normalmente possuem uma paleta de cores reduzida, com isso sendo possível contabilizar individualmente informações sobre cada cor que pode existir na imagem. Então, além de fazer o comparativo dos tons claros e escuros, é planejado comparar cada tonalidade presente na paleta de cores do personagem verificando a eventual diferença na distribuição. Para esta tarefa, é proposto criar mais uma variação do método SMC, desta vez ajustando a fórmula para agrupar o resultado individual de cada cor. A equação 3.4 demonstra essa proposta, onde a operação do somatório acumula o resultado individual para em seguida obter a média simples considerando as N cores que houver na paleta. Quanto ao cálculo do SMC, deseja-se tomar como parâmetro a quantidade de ocorrências de cada cor na imagem gerada e na imagem de referência. Considera-se a mesma necessidade de variar os resultados entre 0 e 1, porém neste contexto da separação por cores não é possível assumir as dimensões visto que cada cor representa apenas uma parcela da

imagem. Então, é considerado a quantidade de ocorrências que for maior entre a imagem gerada e a imagem de referência, desta forma mantendo o mesmo comportamento desejado de assumir valor 0 quando a informação for igual, progredindo até 1 quando a diferença for total. Ou seja, se uma imagem apresentou ao menos 1 pixel com determinada cor e a outra não obteve nenhum com aquela tonalidade, será considerada diferença total. Para complementar esse entendimento, a Figura 5 demonstra essa técnica aplicada utilizando somente três cores para simplificar o exemplo. Tendo definido métodos quantitativos de avaliação de imagens, pode ser abordado na seção seguinte a técnica qualitativa.

$$SMC_{paleta} = \frac{1}{N} \times \sum_{i=1}^{N} \frac{|m_1 - m_2|}{max(m_1, m_2)} \tag{3.4}$$

Figura 5: Exemplo de comparação da paleta utilizando SMC adaptado



Fonte: elaborado pelo autor

3.3 AVALIAÇÃO QUALITATIVA

Tendo visto o método quantitativo, convém discutir também o método com viés mais qualitativo para comparação de imagens. Nesse sentido, o método científico demanda embasamento em critérios que direcionem as percepções sobre a imagem para que essa análise não fique vinculada à impressão pessoal de quem estiver avaliando. Para este fim, o livro Desenhando com o lado direito do cérebro, de Edwards (2012), aborda diversas técnicas do desenho de observação, ou seja, aquele em que a ilustração é elaborada com base em uma referência que é consultada durante todo o processo de criação. Para poder efetuar o desenho de forma que efetivamente represente a realidade, a autora elabora um conjunto de métodos e procedimentos para visualizar de forma técnica a estrutura a ser esboçada, os quais foram desenvolvidos e aprimorados a partir dos estudos da autora e das percepções que ela obteve com seus alunos enquanto professora de desenho. Todo este

conhecimento é de interessante aplicação neste projeto para permitir uma análise mais completa e criteriosa dos *sprites* que serão estudados.

O cérebro humano, ao tentar representar a realidade através do desenho, naturalmente recorre a símbolos e simplificações que acabam comprometendo a exatidão. Isso ocorre por exemplo ao desenhar um olho através de um círculo simples ou o nariz utilizando um triângulo. A proposta principal do livro de Edwards (2012) é enxergar as coisas como elas efetivamente são, sem utilizar essas abstrações. Para trabalhar essa habilidade, um dos exercícios propostos pela autora a seus alunos envolvia fazer uma cópia, utilizando como referência um desenho de cabeça para baixo. Desta forma os alunos ficavam incapazes de interpretar completamente a gravura, muitas vezes sem conseguir entender o que estavam desenhando e isso os forçava a analisar os traços em sua essência, livres de qualquer pré-concepção. A autora observou uma melhora na qualidade da representação dos alunos, e a partir disso identificou quais são os métodos para enxergar as imagens em sua composição técnica, sem este artifício de virar a imagem de ponta cabeça. Sendo assim, podem ser destacadas cinco habilidades básicas do desenho que englobam a capacidade geral de desenhar algo utilizando um referencial externo, ou seja, aspectos que podem ser observados deste modo técnico. Estas habilidades são analisadas brevemente a seguir, indicando como elas se relacionam com o desenho de observação. (EDWARDS, 2012)

- **Percepção das bordas:** capacidade que trabalha a percepção dos contornos das formas, analisando o direcionamento das linhas que compõem uma determinada forma complexa.

- **Percepção dos espaços:** capacidade de perceber que os espaços entre determinadas formas, bem como os espaços no entorno destas formas que se busca desenhar, são também formas a serem analisadas. Chamados espaços vazios ou negativos, eles determinam outras formas que nascem da relação entre os limites (bordas) de uma forma com outra e das formas com o seu contexto.

- **Relacionamento:** capacidade de perceber a relação entre as partes e o todo em uma determinada estrutura ou a relação entre uma forma e seu contexto. Esta habilidade apreendida permite analisar o dimensionamento das formas nas direções básicas vertical e horizontal e também na direção diagonal. A percepção do relacionamento entre as formas e seu entorno permite a definição das proporções e da ocupação de espaços em um determinado contexto.

- **Luz e sombra:** a percepção de luz e sombra permite compreender a qualidade das superfícies de uma determinada forma. Tais qualidades podem indicar a tipologia da forma analisada e ressaltar a impressão de profundidade. A partir da correta

representação de luz e sombra, o desenhista pode retratar a volumetria geral da imagem, bem como as qualidades materiais de todas as superfícies.

- **Gestalt:** segundo Edwards (2012), esta habilidade não se aprende e tão pouco pode ser ensinada, pois ela seria o resultado da aprendizagem e combinação das outras quatro habilidades básicas. Quando apreendida ela permite a percepção e compreensão geral do todo em uma determinada situação de analise formal objeto, espaços e superfícies. Conceitualmente, é tido como estado em que o todo e as partes se relacionam de forma tão intrínseca e coerente que se tornam uma unidade completa e crível.

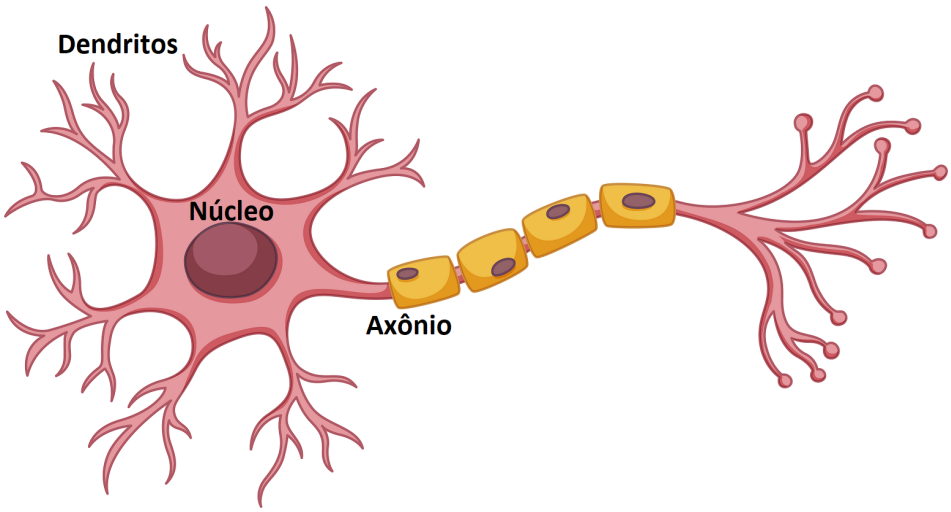
Por meio destes conceitos nota-se que existem técnicas para comparar imagens de forma qualitativa, empregando critérios técnicos bem definidos que advém do desenho de observação. Aplicando estes conhecimentos poderão ser feitas análises com mais rigor na comparação entre as imagens que são estudadas neste projeto. Com tudo isso, percebe-se que todas as técnicas vistas neste capítulo e no anterior realmente auxiliam no desenvolvimento criativo, porém este processo ainda é bastante manual para o artista, pois ele começa cada nova imagem do zero e vai criando os detalhes com base na sua experiência e no conhecimento da arte que pretende criar. Muito disso ocorre porque as ferramentas amplamente utilizadas para criação da arte digital são focadas em facilitar as tarefas do artista, mas não trabalham a automatização de processos, que é uma característica de áreas que têm uma relação mais forte com o desenvolvimento tecnológico, como é o caso da engenharia de software. Neste sentido, já existem diversos estudos e abordagens para o processamento digital de imagens que permitem trabalhar essas questões de automatização. No próximo capítulo, são abordadas essas tecnologias em mais detalhes.

4 REDES NEURAIS

Redes neurais artificiais são um resultado da aplicação de conceitos da biologia na área do aprendizado de máquina. Observando a capacidade do cérebro humano de se adaptar a problemas desenvolvendo uma maneira própria de interpretá-los, começou a ser criado um modelo de processamento de dados que trabalhasse de forma semelhante. Algumas características da rede neural natural foram compreendidas e encontrou-se maneiras de reproduzir artificialmente os mesmos comportamentos. A estrutura que foi desenvolvida a partir disso se mostrou muito eficiente para classificar dados e detectar padrões mediante um processo de aprendizado prévio, sendo atualmente aplicada para resolução dos mais variados problemas.

O elemento fundamental de uma rede neural artificial (RNA) é chamado neurônio. Ele possui duas características principais: a capacidade de armazenar dados e o método de aprendizado que permite moldar essas informações ao contexto necessário. Assim como no cérebro humano, o neurônio é a unidade que se associa a várias outras para efetuar processamentos complexos. Como pode ser visto na Figura 6, um neurônio humano possui como componentes principais: os dendritos, que são as conexões de entrada; o núcleo, que faz a tomada de decisões desta célula; e o axônio, que se estende até as conexões de saída. A função desta célula é basicamente receber um sinal elétrico, avaliá-lo e passá-lo adiante caso entenda que deve ser acionada por ele (HAYKIN, 2007). Definida esta origem das redes neurais artificias, pode ser abordado em mais detalhes quais elementos caracterizam sua estrutura e posteriormente seu modelo de aprendizado, os quais estão descritos nas seções a seguir.

Figura 6: Representação tradicional de um dos tipos de neurônio

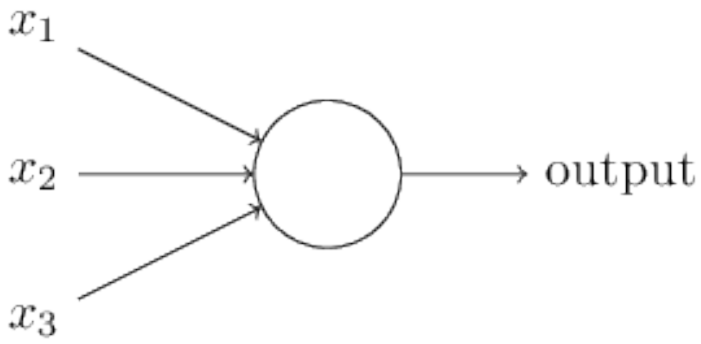


Fonte: adaptado da arte criada por brgfx (Freepik.com)

4.1 ESTRUTURA DE INFORMAÇÕES EM UMA REDE NEURAL ARTIFICIAL

As RNA se inspiraram fortemente na estrutura básica de um neurônio descrita anteriormente, tentando criar um modelo artificial com comportamento parecido. Um dos primeiros modelos criados ficou conhecido como *perceptron*. Nele há uma série de entradas binárias e uma saída binária, como pode ser visto na Figura 7, onde há três parâmetros *x1*, *x2* e *x3*. Além disso, cada uma destas entradas possui um peso associado, ou seja, contribui em diferente grau para compor o resultado, podendo inclusive contribuir de forma negativa. Para calcular a saída, o *perceptron* simplesmente faz a soma dos parâmetros com os pesos aplicados e compara com um limiar (do inglês *threshold*), que é um número real qualquer que serve de referência para calibrar a resposta. O neurônio artificial é acionado ou não dependendo da composição da entrada, semelhante ao que ocorre no cérebro humano. (NIELSEN, 2015)

Figura 7: Representação de um *perceptron*

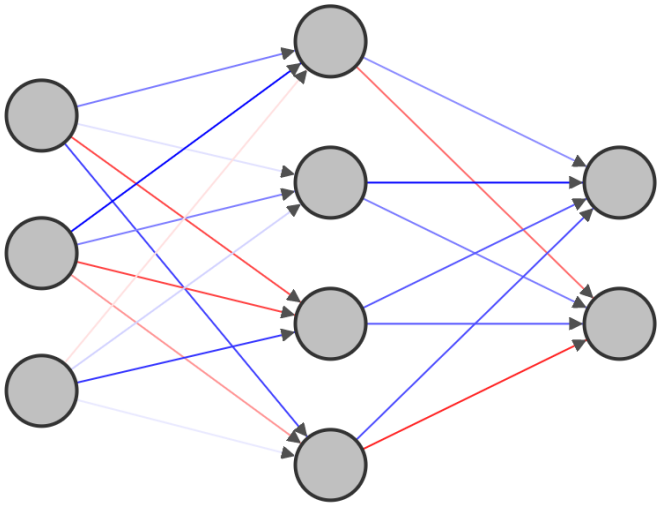


Fonte: Nielsen (2015)

Um neurônio único é capaz de avaliar linearmente vários parâmetros, entretanto os problemas que as redes neurais buscam resolver envolvem a existência de variáveis que se relacionam de forma desconhecida. Portanto, em uma RNA completa há diversos neurônios conectados e compondo juntos a saída, de forma que possam trabalhar relações mais complexas entre os parâmetros. Para fazer isso, os neurônios são organizados em camadas, onde a primeira delas corresponde à identificação dos atributos de entrada, e a última relaciona as saídas geradas, como pode ser visto na Figura 8. A camada central é onde efetivamente se encontram os neurônios da rede.(HAYKIN, 2007)

Entre a entrada e a saída, os neurônios são posicionados em um ou mais níveis ocultos, nos quais cada neurônio recebe como entradas todas as saídas da camada anterior. Estas camadas são denominadas ocultas pois o usuário da RNA não precisa visualizar de forma detalhada como elas ficaram estruturadas, basta que o resultado final descreva corretamente o comportamento dos parâmetros de entrada. Conforme são utilizados mais neurônios em uma mesma camada, mais análises distintas a rede consegue fazer sobre os mesmos parâmetros; enquanto isso, maior quantidade de etapas ocultas provém à RNA

Figura 8: Exemplo de uma rede neural artificial (RNA) simples



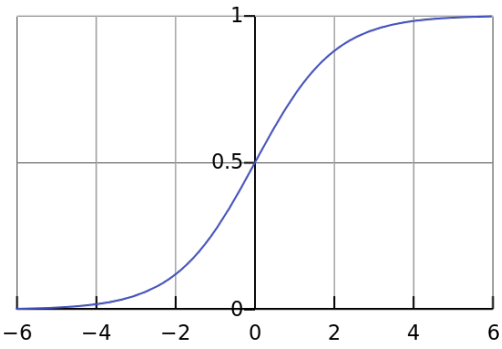
Fonte: produção do autor

maior abstração, já que cada novo processamento vai trabalhar com dados que já foram interpretados e alterados anteriormente. Vem desta ideia de empilhamento de várias etapas o conceito de aprendizado profundo (do inglês *Deep Learning*). A definição da quantidade de neurônios em cada camada e a quantidade de camadas é tarefa de quem estiver projetando a rede neural artificial e é passível de experimentação até achar a distribuição mais adequada ao problema. (HAYKIN, 2007)

Um problema verificado no neurônio *perceptron* é que ele se tornava muito suscetível às variações nos parâmetros de entrada e aos pesos aplicados a cada um. Pelo fato de a saída ser binária, eventuais ajustes nos pesos poderiam inverter o resultado da saída em vários cenários e não somente no caso desejado, degradando a performance da rede ao invés de melhorá-la. Outro problema é tratamento linear dos parâmetros, o que torna universo de situações que a rede pode processar limitado. Finalmente, cada neurônio de saída fica totalmente ativo ou totalmente inativo, dificultando uma análise comparativa de cada saída em relação às demais. Por tudo isso, entendeu-se que havia um novo parâmetro a ser definido nos neurônios, que ficou conhecido como função de ativação (NIELSEN, 2015). Essa função é o que define o comportamento do neurônio com relação ao método para transformar os parâmetros de entrada no parâmetro de saída, além de definir se o neurônio ficará ativo ou não para determinado conjunto de entradas. Segundo Sharma (2017), as funções de ativação mais utilizadas na atualidade são:

1. **Função Sigmoid:** Esta função transforma a saída em um intervalo não linear entre 0 e 1, de forma que os parâmetros ainda influem significativamente na saída, mas haverão variações mais suaves. Além disso resolve a limitação do domínio linear dos problemas.

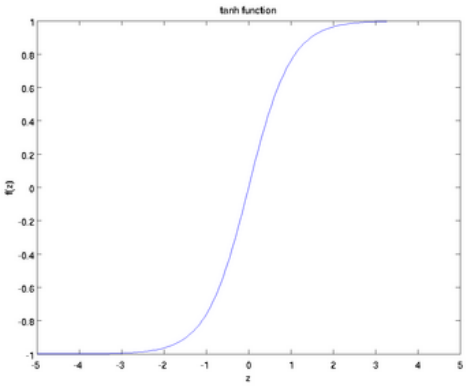
Figura 9: Função Sigmoid



Fonte: Sharma (2017)

2. **Função Tanh:** Esta função é quase idêntica à função Sigmoid, porém trabalha com domínio negativo na saída. Assim como ocorre na função Sigmoid, possui uma curva tão suave nas extremidades que pode causar um aprendizado mais lento da RNA quanto muitos neurônios atingem estes pontos da função.

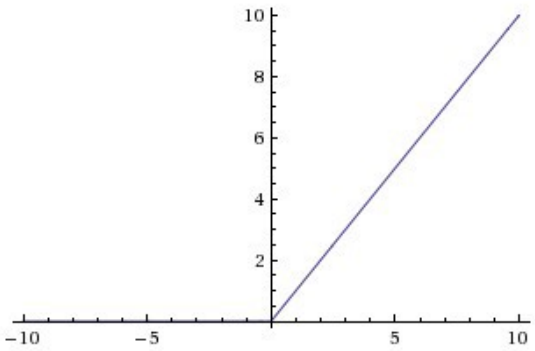
Figura 10: Função Tanh



Fonte: Sharma (2017)

3. **Função ReLU:** Do inglês *Rectified Linear Unit*, esta função faz com que valores positivos sejam transportados diretamente para saída, enquanto que valores negativos são zerados. Apresenta como vantagem o fato de não possuir curva suave nas bordas, o que ajuda a manter a taxa de aprendizado da rede. Além disso, utiliza uma operação matematicamente mais simples que as demais apresentadas e portanto gera um desempenho melhor. A região zerada nessa função de ativação traz tanto o benefício de aliviar a carga da RNA ao desativar alguns neurônios, como também pode ser um problema caso deixe muitas partes da rede inoperantes. Ambas as situações são melhoradas com pequenas variações neste modelo que não usam uma reta horizontal nessa região zerada.

Figura 11: Função ReLU



Fonte: Sharma (2017)

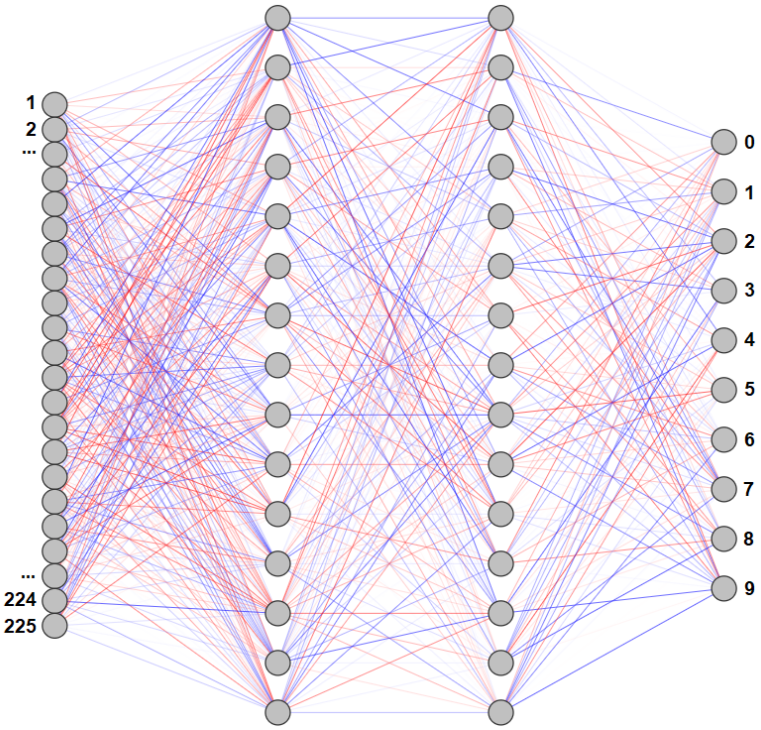
4.2 MODELO DE APRENDIZADO DE UM REDE NEURAL ARTIFICIAL

No começo desta seção foram destacadas duas características dos neurônios de uma rede neural artificial: a capacidade de armazenar informação e a capacidade de aprendizado. Até agora foi abordado somente a forma como a RNA armazena as informações e a partir delas compõe o resultado. Para entender o processo de aprendizado, convém utilizar um dos exemplos mais clássicos de aplicação de redes neurais: o reconhecimento de dígitos escritos à mão livre.

O objetivo desta RNA hipotética é ler uma imagem na entrada da rede e na saída indicar qual número de 0 até 9 que está representado na imagem. Essa tarefa se mostra complexa pois há várias formas de representar o mesmo número sem perder a sua essência, já que a forma geral do dígito depende da caligrafia do indivíduo que o escreveu. Isso torna uma abordagem com algoritmos tradicionais quase impraticável, pois seria necessário prever muitas situações variáveis. Entretanto, o reconhecimento de imagem acaba sendo adequado ao tipo de problema que uma rede neural artificial consegue trabalhar pelo fato de haver uma correspondência definida entre os dados de entrada e a resposta desejada.

Para tentar interpretar a imagem utilizando uma RNA, pode ser considerado cada pixel dessa imagem como uma entrada. Supondo que são processadas imagens com tamanho 15x15, haveriam 225 parâmetros. Na saída da rede, são necessários 10 neurônios, cada um representando a possibilidade de a imagem de entrada corresponder ao respectivo número de 0 até 9. Como foi visto, dependendo da função de ativação utilizada, a saída poderá ser por exemplo um número real de 0 até 1, onde 1 seria o indicativo de certeza da RNA em um determinado dígito. Nas camadas ocultas este exemplo vai utilizar 2 camadas com 15 neurônios em cada uma delas. Na Figura 12 há um modelo de como esta RNA ficaria estruturada. Nesta imagem foram omitidos alguns neurônios de entrada para simplificação e os pesos aleatórios estão exemplificados com as cores azul para valores positivos e vermelho para os negativos.

Figura 12: RNA para reconhecimento de dígitos



Fonte: produção do autor

Dessa forma, o processo de aprendizado ocorre mediante um treinamento prévio em que são fornecidos à rede exemplos de entradas e os respectivos resultados esperados. No contexto do reconhecimento de dígitos escritos à mão livre, o treinamento seriam imagens dos números e a resposta de qual dígito está representado na imagem. Inicialmente é aplicando um peso aleatório nas conexões de todos os 15 neurônios de cada uma das duas camadas da rede. Com isso, ao fazer a execução inicial cada neurônio da rede aplica seus pesos e o cálculo interno e entrega a resposta para a próxima camada oculta da rede até que os neurônios de saída sejam carregados. Tendo como base estes pesos aleatórios, provavelmente a resposta inicial será bastante incorreta, mas é neste momento que começa o aprendizado.

A RNA faz uma comparação entre a saída que foi gerada e o resultado esperado para cada um dos exemplos fornecidos. A partir disso, caracteriza uma função que descreve o custo (do inglês *loss*) geral da rede, que é um termo utilizado para indicar o grau de erro nas respostas que a rede oferece. Através de processos matemáticos que buscam minimizar essa função de custo, a rede vai identificar em média quais pesos precisam ser ajustados para melhorar o desempenho geral. A técnica que identifica os ajustes é chamada descida de gradiente (do inglês *gradient descent*) e permite com que, após processar uma vez todos os exemplos de entradas e saídas conhecidas, a rede possa calibrar os pesos que inicialmente eram aleatórios de forma que eles se aproximem da distribuição ideal para resolver todos os exemplos. A sequência de ajustes que percorrem a rede neural ocorre

através de retropropagação (do inglês *backpropagation*), pois começa pelo resultado final e vai se repetindo em cada camada até atingir a entrada da RNA. (SANDERSON, 2017)

O processo de leitura de todos os exemplos e calibragem dos pesos da RNA constitui uma época de aprendizado (do inglês *epoch*) e a quantidade exata de épocas a serem executadas também é um parâmetro a ser definido por quem estiver projetando ou aplicando a rede neural artificial. Quando mais épocas a RNA cumprir durante o processo de treinamento, mais apta ela estará a resolver os problemas propostos. Entretanto, não é indicado executar uma quantidade alta de épocas de forma indiscriminada, pois podem surgir alguns problemas. O primeiro deles é a queda na taxa de aprendizado da RNA, quando a função custo já está se aproximando do mínimo e não haverá muitos ganhos para o desempenho em função do tempo investido. Essa situação é denominada desaparecimento de gradiente (do inglês *vanishing gradient*), pois o processo de descida de gradiente deixa de apresentar avanços significativos. Outro problema é o chamado *overfitting*, que é termo em inglês para o estado em que a RNA se torna muito viciada nos exemplos fornecidos e torna-se pouco adaptável para as situações reais nunca treinadas anteriormente. Ou seja, é uma situação em que a rede apenas memorizou todas as correspondências entre entradas e saídas ao invés de generalizar uma solução. No exemplo de reconhecimento de dígitos, haveria alta eficiência em reconhecer os dígitos que já foram vistos, porém o desempenho seria fraco ao inserir novas imagens. No que diz respeito ao uso de camadas ocultas, estes dois problemas tendem a demorar mais para ocorrer para cada camada oculta que for adicionada. Isso ocorre pois novas camadas aumentam o uso da retropropagação e com isso o aprendizado é mais lento nas camadas ocultas iniciais, aumentando a quantidade de épocas de treino até que a RNA apresente estes problemas. (NIELSEN, 2015)

Ainda com relação ao processo de aprendizado, há dois tipos principais: supervisionado e não supervisionado. Em um modelo de aprendizado supervisionado, a RNA recebe a indicação se a sua previsão sobre os dados está correta e a partir disso aprimora a sua estrutura interna, ou seja, o exemplo abordado nesta seção se enquadra nesta categoria. Enquanto isso, um aprendizado não supervisionado não fornece essa indicação do acerto, ao invés disso a rede tenta localizar relações entre os parâmetros de entrada conforme são apresentados novos exemplos. Dessa forma a própria rede neural vai criar rótulos nos dados conforme for identificando similaridades entre os diferentes dados recebidos. Esse tipo de aprendizado é interessante de ser aplicado quando há muitos parâmetros de entrada e não é possível identificar quais são os mais relevantes para categorizar os registros (BARROS, 2016).

Com tudo isso, nota-se como uma rede neural pode fazer o reconhecimento de informações em uma imagem. Apesar de ser um processo já bastante eficiente, existem modelos de redes neurais que se adaptam melhor a processamentos com imagens ou ou-

tras estruturas de dados que possam ser representadas na forma de uma matriz. Tendo entendido como uma rede neural tradicional está estruturada, na próxima seção será visto como essa estrutura pode ser aprimorada para trabalhar com imagens.

4.3 ARQUITETURAS DE REDES NEURAIS ARTIFICIAIS

Anteriormente foi destacada a estrutura de uma RNA básica além do seu comportamento no que diz respeito ao processo de aprendizado. Foi visto como é possível, inclusive, resolver problemas que envolvem reconhecimento de imagens. Esse tipo de análise gráfica é complexa para um algoritmo tradicional pelo fato de envolver muitos cenários distintos a serem previstos, enquanto que uma RNA consegue ser mais dinâmica e se adaptar ao contexto. Apesar deste ganho, pode ser observado que a solução proposta utilizando uma RNA simples também pode se exigir muitos recursos computacionais. Considerando o exemplo abordado, a imagem de apenas 15x15 pixels de dimensão acaba gerando uma entrada de 225 parâmetros na RNA. Considerando as duas camadas ocultas com 15 neurônios em cada e sabendo que cada uma delas se conecta totalmente com a camada seguinte, calcula-se que a rede possui 3750 conexões até atingir a saída. Este número aumenta proporcionalmente quanto maior for a estrutura interna da RNA ou o tamanho da entrada. Mesmo utilizando funções de ativação como o ReLU que são matematicamente mais simples, ainda torna-se oneroso calibrar tantas conexões a cada época de aprendizado. Em uma situação prática nem sempre é possível trabalhar com imagens tão pequenas, e portanto surgiram modelos mais avançados de RNA que são aplicados em processamentos de imagens. Nesta seção serão abordados dois tipos específicos que serão aplicados no decorrer deste projeto.

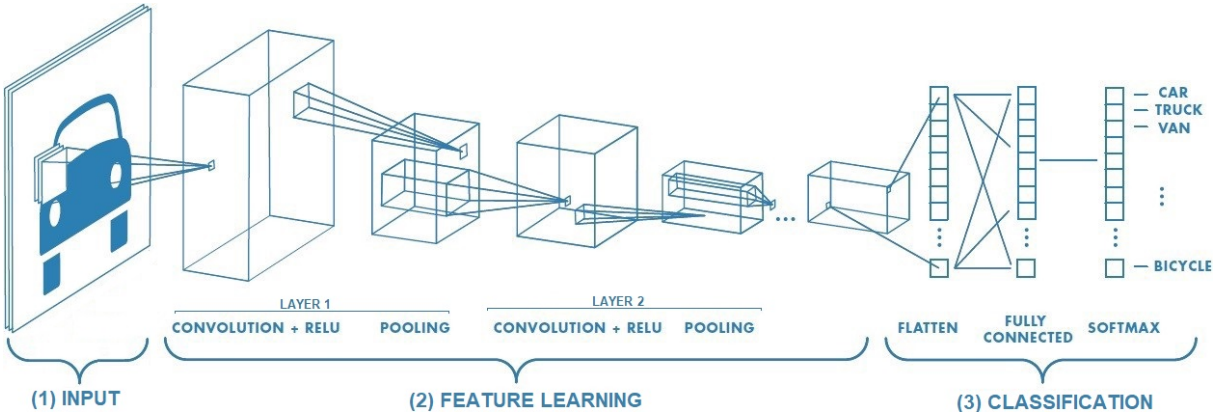
4.3.1 Redes Neurais Convolucionais

Para permitir o uso de redes neurais artificias nas situações em que os dados estão distribuídos em forma de tabelas, com linhas e colunas definidas, começou-se a estudar conceitos matemáticos que trabalham com matrizes. Devido a essa fundamentação, a solução proposta pelas redes neurais convolucionais acabou sendo adequada também para qualquer dado que possa ser representado neste formato. Um exemplo disso são os estudos de sinais sonoros, onde que cada faixa de frequências pode ser representada em um eixo da tabela e a variação conforme o tempo no outro eixo.(SAHA, 2018)

Uma rede neural convolucional é chamada em inglês *convolutional neural network*, e muitas vezes é referida apenas como CNN. Normalmente é implementada como uma RNA híbrida, logo, mesclando características próprias com modelos tradicionais. Como pode ser visto na Figura 13, há três grandes etapas de processamento: entrada de dados (*input*), aprendizado de características (*feature learning*) e classificação (*classification*). Para ilustrar este exemplo, será considerado uma aplicação típica de CNNs que é o reco-

nhecimento de imagens. Este tipo de CNN recebe como entrada uma imagem e atribui a ela um de seus rótulos de saída conhecidos, buscando descrever o que está representado na imagem. Ou seja, neste caso em que há um veículo representado, espera-se que a CNN consiga indicar que se trata de um carro. Apesar de ser um exemplo específico para facilitar o entendimento, ele ilustra as etapas que todas as CNN possuem.

Figura 13: Exemplo de um processo possível em uma CNN



Fonte: Adaptado de Saha (2018)

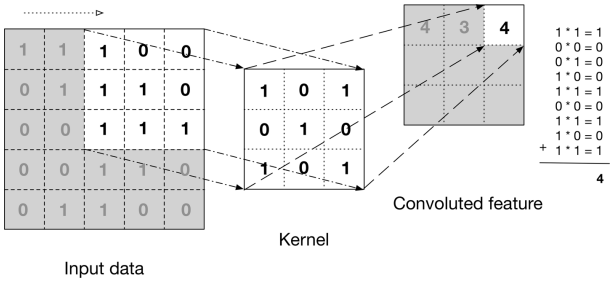
Como se vê, o processo de entrada dos dados indicado no bloco (1) da Figura 13 começa com a matriz que será analisada, que neste exemplo é a figura de um carro. Supondo que é uma imagem colorida, haverá três canais de cores RGB, conforme discutido na seção 2.2. Isso implica que a matriz de entrada da CNN possui uma terceira dimensão que é a profundidade, que descreve os diferentes canais de cores. Essa informação individualizada dos canais de cores poderá ser usada ou não, dependendo de como ficar estruturada a rede neural ao final do processo de aprendizagem.

A próxima etapa, indicada no bloco (2), é o aprendizado de características (do inglês *feature learning*), onde se encontram os neurônios da CNN. Eles concentram três mecanismos que distinguem uma CNN de outros tipos de RNA, que são: a convolução, o ReLU e o *pooling*. (PATTERSON; GIBSON, 2017) Apesar de estarem representados com uma organização específica, é importante ressaltar que em uma construção prática é possível alterar as ordem em que eles ocorrem ou até mesmo repetir um mesmo processo mais de uma vez.

1. **Convolução:** Este é o processo matemático que dá nome a CNN. É uma operação amplamente utilizada no processamento de imagens que é capaz de extrair padrões específicos que ocorram em qualquer ponto dentro da figura. Isso é feito definindo uma máscara, muitas vezes também chamada de *kernel* ou filtro, que é uma matriz de tamanho menor que contém o padrão específico a ser localizado. A convolução faz um cálculo sobre toda a imagem e gera uma nova figura que destaca os pontos

onde este padrão estiver presente. O cálculo, que pode ser visto na Figura 14, é feito sobrepondo este *kernel* em um ponto específico da imagem de entrada, multiplicando cada par de pixels que ficaram empilhados e somando todos os valores. O resultado é atribuído à posição que corresponder ao pixel central da máscara.

Figura 14: Cálculo de uma convolução



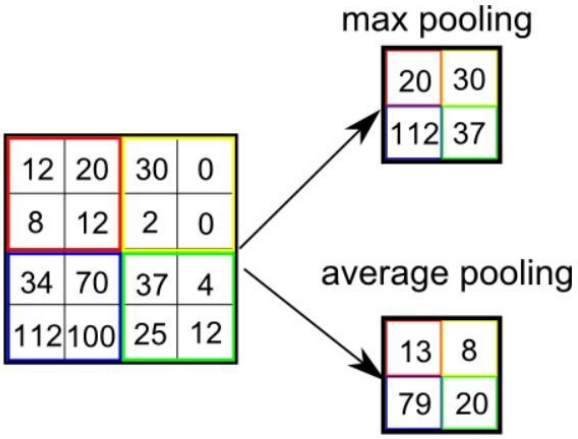
Fonte: Saha (2018)

Para gerar a nova imagem, a convolução repete este processo em cada ponto da imagem em que for possível aplicar a matriz, ou seja, varre pixel-a-pixel a largura e altura da imagem. No caso de haver os canais de cores RGB, este processo pode ser feito simultaneamente nos três canais, e nesta situação o *kernel* é uma matriz tridimensional. A operação de convolução também possui a característica de reduzir o tamanho da imagem dependendo do tamanho do filtro utilizado. Na operação que foi vista na Figura 14, utilizando um *kernel* de 3x3 pixels, não será possível atingir de forma completa 1 pixel da borda da imagem pois a máscara ficaria sobreposta parcialmente para fora da imagem. Existem técnicas de cálculo para prever essa situação, mas normalmente se opta por perder os pixels da borda da imagem por dois fatores: estes pixels já foram considerados por aplicações da máscara nos pixels adjacentes, além de que a redução no tamanho da imagem é benéfica para a CNN, como será visto no tópico *pooling*.

2. **ReLU:** Este conceito já foi visto anteriormente como um dos tipos de função de ativação que pode ser implementado em um neurônio. Foi apresentado como sendo uma operação que mantém os números positivos e altera os demais para zero. No contexto de uma CNN que trabalha com imagens, essa operação serve para eliminar quaisquer valores negativos nos pixels após a operação de convolução, já que normalmente não se representa cores com escala negativa. Adicionalmente, os pixels que ficam com valor zerado tendem a serem calculados mais rapidamente do que aqueles que possuem algum valor específico. Desta forma o ReLU deixa a imagem resultante mais fácil de interpretar nas etapas seguintes, evidenciando os pontos que geraram alguma informação (valores positivos) enquanto que normaliza todos os demais como zerados.

3. **Pooling:** Este é um processo que melhora dois aspectos da CNN com um mesmo procedimento. O *pooling* varre toda a imagem passando uma máscara, de forma semelhante à convolução, porém ao invés de aplicar um filtro o objetivo é reduzir o tamanho da imagem. Para isso, a máscara seleciona determinada quantidade de pixels e escolhe somente um valor para representá-los na imagem de saída. Normalmente esse processo se encontra implementado de duas formas, que podem ser vistas na Figura 15: ou é selecionando o pixel com maior valor, ou então é feita uma média dos valores de todos os pixels sob a máscara. Essa operação é vantajosa para a CNN porque reduz a quantidade de dados que precisam ser processados nas etapas seguintes, o que melhora o desempenho, além de que ajuda a reduzir o *overfitting*, que foi explicado na seção 4.2. Ou seja, ao desconsiderar alguns segmentos da imagem acaba sendo aumentada a capacidade da CNN de lidar com situações que sejam semelhantes, sem que o treinamento seja proveitoso somente para caso específico de cada imagem. (LEE *et al.*, 2009)

Figura 15: Exemplo de uma operação de pooling



Fonte: Saha (2018)

Como foi visto nesta segunda etapa, representada pelo bloco (2) na Figura 13, a CNN utiliza estes três processos para gradativamente reduzir o tamanho da imagem e aplicar filtros que buscam padrões específicos. Há muitos filtros distintos que uma CNN pode aprender, como por exemplo detectores de bordas ou de padrões de cores. Assim como ocorre em uma RNA simples, o objetivo de combinar estes processos de convolução, ReLU e *pooling* é aumentar a abstração da rede neural. Ou seja, fazer com que a cada nova operação a rede crie filtros que reconheçam características mais complexas da imagem. Retornando ao exemplo do reconhecimento de imagens citado no começo desta seção, uma CNN poderia detectar bordas nas primeiras camadas da rede, para então combinar os resultados dessas operações e reconhecer padrões de círculos ou curvas que definem certos objetos. Ao final desta etapa, há então um conjunto de pixels abstratos que descrevem

de forma mais sucinta e mais significativa as características da imagem e que servem de entrada para a etapa final da CNN. (PATTERSON; GIBSON, 2017)

Por fim, na última etapa que foi representada pelo bloco (3) na Figura 13, está o processo de classificação. Tendo a imagem já representada pelo resultado de vários filtros que abstraem de suas características, esta parte da rede neural convolucional é normalmente implementada como uma RNA simples que tem como objetivo enquadrar os dados recebidos em uma das opções de saída disponíveis, assim como foi visto na seção 4.2. O que acaba diferenciando esta RNA é o contexto em que ela está inserida, pois ela necessita de algum processo para adequar o formato de saída dos filtros aplicados na etapa anterior com o formato esperado pelos neurônios de entrada da rede. Este processo está representado pela legenda *flatten* no bloco (3) da Figura 13 e representa o "achatamento" das matrizes de saída para que sirvam de entrada para a RNA, muito semelhante ao que foi comentado na seção 4.2 sobre transformar cada pixel da imagem em um parâmetro de entrada, a diferença é que neste caso os pixels contém muito mais informação do que somente a cor, pois são o resultado dos processos cada vez mais abstratos das etapas anteriores. Além disso, neste caso específico de uma CNN para reconhecimento de imagens, também há um processo muito comum de ser aplicado que é o chamado *softmax*. Esse é um procedimento que calcula os valores de distribuição probabilísticos entre classes que sejam mutuamente excludentes, ou seja, exatamente o tipo de saída que uma RNA de classificação costuma gerar.

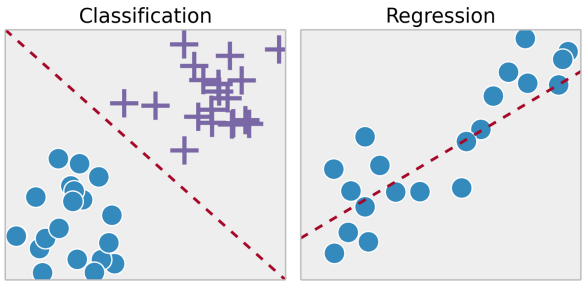
Finalmente, o processo de aprendizado de uma CNN é análogo a uma RNA. Entretanto, ao invés de calibrar pesos que conectam os diferentes neurônios, esta rede neural vai utilizar a retropropagação para carregar as máscaras de convolução de forma a descrever cada vez melhor as imagens de treinamento. Desta forma os filtros específicos que a rede vai utilizar não precisam ser definidos pelo projetista, pois serão criados durante o processo de treinamento. Há outro diferencial importante chamado *dropout*, que é um processo que desativa os neurônios que tenham fraca conexão em uma rede neural artificial. No contexto de uma CNN esse procedimento é amplamente aplicado para evitar que sejam criados filtros pouco relevantes ou que sejam previstas conexões entre filtros que não são significativos quando combinados. Ou seja, enquanto que uma RNA geralmente possui todos os neurônios conectados entre cada camada, uma CNN implementa o mecanismo de *dropout* para reduzir a quantidade de conexões da rede.

Com tudo isso, nota-se que as CNN possuem aplicação para análise de imagens e muitos mecanismos que visam aprimorar seu desempenho neste contexto em que a quantidade de dados de entrada tende a ser grande. Assim como as RNA, possui grande capacidade de classificar informações sobre os dados de entrada. Na próxima seção, será visto mais um modelo de rede neural artificial que é muito utilizado com imagens, mas que possui um viés de geração de dados ao invés de reconhecimento.

4.3.2 Redes Neurais Adversárias Geradoras

Muitas RNA podem ser definidas como redes de classificação ou de regressão. Redes de classificação têm como objetivo principal interpretar um conjunto de dados e atribuir um rótulo aos mesmos, gerando uma saída discretizada. Enquanto isso, redes de regressão trabalham com dados contínuos na saída. Em ambos os casos, o objetivo principal destas RNA é mapear uma função que descreva corretamente a transformação dos parâmetros de entrada nas saídas especificadas, como pode ser visto na Figura 16 onde o conjunto de dados é discriminado por uma rede de classificação ou então descrito por uma aproximação de uma rede de regressão. Com a aplicação de redes neurais artificias nos mais variados contextos, estas técnicas de aprendizado de máquina encontraram aplicação também para resolver situações em que é necessário criar dados de forma automatizada ao invés de apenas analisá-los. Ou seja, redes neurais que são capazes de utilizar este conhecimento adquirido sobre os dados de entrada para gerar novas informações que sejam semelhantes (BARROS, 2016).

Figura 16: Exemplo de operações de classificação e regressão



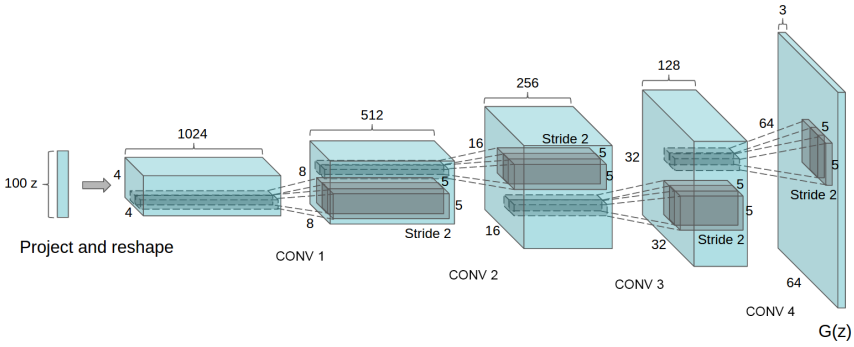
Fonte: Barros (2016)

Redes neurais adversárias geradoras são comumente referidas por sua sigla em inglês GAN (*generative adversarial networks*) e provêm uma solução para este tipo de problema de geração de dados. Foram apresentadas inicialmente por Goodfellow *et al.* (2014) e dentre outros tipos de redes neurais artificiais capazes de gerar dados, esta se destaca por utilizar tanto características de aprendizado supervisionado quanto não supervisionado, que foi visto na seção 4.2. Assim como ocorre em qualquer rede neural geradora, o funcionamento básico de uma GAN é descobrir direta ou indiretamente as distribuições probabilísticas de cada parâmetro de entrada que serviu de treinamento. A partir disso, é possível selecionar um valor aleatório para cada um dos parâmetros de entrada de forma que o resultado final ainda seja reconhecido como válido. Alguns modelos fazem essa seleção buscando diretamente valores dos exemplos conhecidos através do treinamento, enquanto que outras implementações podem gerar uma função de regressão para cada um dos parâmetros. Neste último caso, é selecionado algum ponto randômico da função descoberta e adiciona-se um ruído aleatório para gerar as novas entradas. Com isso, o processo de aprendizado das redes artificiais geradoras se realiza identificando me-

lhor a cada época de treinamento quais são as características semelhantes no universo de dados de entrada e quais podem ser variadas sem comprometer a consistência dos registros gerados em relação aos que são reais.(GOODFELLOW; BENGIO; COURVILLE, 2016)

Assim como as RNA foram aplicadas para extração de características de imagens, as GAN também são amplamente aplicadas para geração de novas figuras. Ou seja, este tipo de rede geradora consegue aprender quais são os traços gerais de todas as imagens de treino e quais são as diferenças irrelevantes, sendo capaz de gerar novas saídas que são visualmente semelhantes ao conjunto de dados de entradas mas que contenham características únicas. Por exemplo, uma GAN treinada com centenas de fotos de salas de estar poderia gerar novas figuras que se encaixam neste padrão mas que não são iguais a nenhuma imagem vista anteriormente. O processo capaz de criar tais imagens é essencialmente o oposto ao que foi visto na subseção 4.3.1, onde as CNN usam convoluções para extrair características das imagens. Ao invés disso, as GAN utilizam operações de deconvolução para gerar matrizes de dados cada vez mais significativas até chegar a uma saída que é semelhante ao universo de treinamento, como pode ser visto na Figura 17. Enquanto que uma CNN elimina parte das informações durante as convoluções, uma GAN cria matrizes com novos dados a partir dos filtros, garantindo que o resultado ficou coerente com o conteúdo filtrado.(RADFORD; METZ; CHINTALA, 2015)

Figura 17: Processo de deconvolução para geração de dados



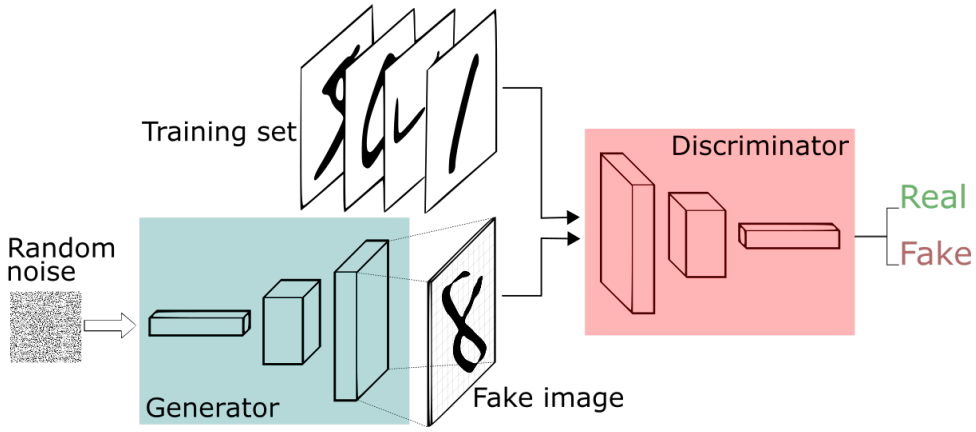
Fonte: Radford, Metz e Chintala (2015)

A partir dessa contextualização sobre os fundamentos de uma rede neural adversária geradora, pode ser definida sua estrutura de funcionamento. O processo é caracterizado por ter duas redes neurais artificiais funcionando simultaneamente, mas que se comportam como uma rede única para atender ao objetivo de aprender sobre os dados e gerar novas entradas. Enquanto que as duas redes vão entre si utilizar um aprendizado supervisionado que é descrito a seguir, o processo como um todo é classificado como não supervisionado pelo fato de que a GAN precisa receber somente os próprios dados de entrada para efetuar o aprendizado, sem necessidade de indicar rótulos ou relações entre os dados previamente. (SILVA, 2018)

Dentro de uma rede adversária geradora (GAN), como pode ser visto na Figura 18,

a primeira sub-rede é nomeada gerador e utiliza as operações de deconvolução descritas anteriormente para, a partir de um ruído aleatório, gerar uma imagem potencialmente pertencente ao conjunto de imagens de treino. Portanto, esta primeira rede tem como objetivo gerar imagens cada vez mais coerentes com o conjunto de entrada. A segunda sub-rede é denominada discriminador e é uma CNN que tem como objetivo classificar as imagens como reais ou geradas artificialmente. Além destas duas redes, há um seletor que aleatoriamente fornece como entrada ao discriminador uma imagem do conjunto de treino ou uma imagem gerada pela rede deconvolucional do gerador. Como a função do discriminador é classificar as imagens corretamente, essa rede neural vai trabalhar seu aprendizado para se tornar cada vez mais eficiente em diferenciar imagens reais e falsas. Enquanto isso, o objetivo da rede geradora é enganar o discriminador de forma que ele interprete como real uma imagem que na verdade foi gerada, e portanto esta rede vai se tornar cada vez mais apta a reproduzir as características imagens do conjunto de treino. Ao final do processo, o que poderá ser utilizado efetivamente é a rede neural geradora, para produzir novas imagens. Pelo fato de conter duas sub-redes com propósitos opostos é que este processo foi denominado de rede adversária. (GOODFELLOW *et al.*, 2014)

Figura 18: Processo típico de uma implementação de GAN



Fonte: Silva (2018)

Como nota-se, as redes neurais adversárias geradoras possuem uso para aplicações com imagens, tendo sido utilizas nos últimos anos em diversos projetos de estudo e servindo de base para construção de redes neurais artificiais ainda mais complexas que se fundamentam nos seus conceitos. No próximo capítulo, serão vistas algumas técnicas de processamento computacional de imagens, dentre elas aplicações práticas dos conceitos de RNA vistos até agora.

5 PROCESSOS COMPUTACIONAIS APLICADOS A IMAGENS

Neste capítulo, serão abordadas áreas de interesse deste projeto com relação a aplicações práticas de técnicas computacionais no processamento de imagens. Serão vistos processos capazes de reconhecer a pose humana a partir de imagens 2D, alterar a escala de imagens melhorando a resolução e técnicas para transformação de imagens utilizando redes neurais artificiais.

5.1 DETECÇÃO DE POSE

A estimativa de pose humana (do inglês *human pose estimation*) é um problema que envolve extrair informações sobre a posição de cada parte do corpo a partir de uma imagem bidimensional. Esta área possui diversas aplicações práticas, como, por exemplo, captura de movimentos para videogames ou animações, análise de postura e interpretação automatizada de linguagem de sinais. No meio científico, este tópico têm sido estudado mais extensivamente após os primeiros experimentos que utilizaram CNN e obtiveram resultados mais viáveis de serem aplicados em cenários reais de uso. Como foi visto anteriormente, o reconhecimento de imagens é uma tarefa complexa que é simplificada com o uso deste tipo de redes neurais artificiais. (RAJ, 2019)

Diferente do exemplo de reconhecimento de dígitos escritos à mão livre que foi explorado na seção 4.2, a análise de fotografias com pessoas abre ainda mais espaço para variações nos padrões a serem reconhecidos pela CNN. Neste contexto, a iluminação influi na imagem que será analisada, bem como o tipo de roupa que o indivíduo estiver utilizando, pois pode acrescentar volume ou alterar o formato do corpo. Outro fator que aumenta a dificuldade para obter a pose é oclusão de partes do corpo. Ou seja, o fato de podem aparecer somente alguns membros do corpo em função da posição da pessoa em relação à câmera, por haver partes da roupa ou de outras pessoas em frente. A partir desta problemática, foram criadas diversas soluções para detecção de pose com diferentes táticas de análise. Nesta seção será abordado as técnicas mais consolidadas nesta área, destacando como funciona um processo típico. (BABU, 2019)

A lógica de detecção mais adotada trabalha o problema de forma holística, ou seja, busca interpretar a pose como um todo e não cada parte individualmente. Desta forma, consegue deduzir com mais coerência a posição e frequentemente definir também as partes ocultas do corpo a partir das demais que estão visíveis (TOSHEV; SZEGEDY, 2014). Para extrair a pose, o processamento começa com várias CNN que são previamente treinadas com fotos de pessoas e a respectiva indicação de onde está cada ponto de interesse. Estas redes buscam inicialmente regiões mais abrangentes do corpo e com isso segmentam a

imagem em vários blocos, como por exemplo cabeça, torso, braços e pernas. Após isso essas regiões vão recebendo filtros cada vez mais específicos para refinar a segmentação. Nas primeiras modelagens essa etapa já definia numericamente os pontos específicos a serem entregues como resposta, porém determinar isso na imagem se mostrou bastante complexo mesmo para uma CNN.

A partir disso, os próximos estudos promoveram uma melhoria no formato de saída da rede. Ao invés de tentar regredir as informações da imagem diretamente até uma coordenada numérica específica, passou-se a gerar mapas de calor que indicam a probabilidade de o ponto específico da pose estar em determinado pixel. Desta forma, o pixel mais intenso dentro de cada mapa de calor vai indicar a coordenada mais provável e ainda será possível extrair um ponto numérico a partir disso, mas o processo para chegar nessa conclusão é simplificado pelo fato de aproveitar os recursos de cálculo de matrizes de uma CNN (TOMPSON *et al.*, 2015). Outra melhoria foi a criação do conceito de CPM (*Convolutional Pose Machine*), que significa máquina de pose convolucional, tradução nossa. Este é um componente adicionado para supervisionar e gerenciar as convoluções durante a extração de características. Contém mecanismos para direcionar o aprendizado considerando questões práticas, como por exemplo, o fato de que o ombro não pode estar muito distante do cotovelo. Esse tipo de lógica permite descartar falsos positivos e acelerar o processo de mapeamento. A implementação específica da CPM é variável, mas seu papel é sempre este de acelerar e otimizar a geração dos mapas de calor (WEI *et al.*, 2016).

Inicialmente a detecção de pose não era aplicável em muitos cenários reais, por exigir que houvesse apenas uma pessoa de cada vez na imagem. Isso era preciso porque do contrário ocorria frequentemente oclusão de partes ou sobreposição de membros de um sujeito com outro, dificultando a detecção e gerando previsões distantes da realidade. As próximas implementações resolveram esta limitação utilizando duas principais abordagens que ficaram conhecidas como *top-down* e *bottom-up* (NEWELL; YANG; DENG, 2016):

1. ***Top-down*:** É adicionando um componente na entrada da CNN que primeiramente classifica os diferentes indivíduos através de técnicas de segmentação de objetos para só então aplicar a lógica principal que processa cada um deles. Ou seja, faz primeiro uma análise global da imagem para deduzir dentro de cada bloco os dados da pose.
2. ***Bottom-up*:** É adicionado um componente na saída da CNN que agrupa os diferentes mapas de calor gerados para compor a pose se cada indivíduo presente. Ou seja, faz primeiro a análise refinada dos pontos de interesse e somente após agrupa eles para formar a pose e definir a qual corpo cada ponto pertence.

Com relação aos pontos que são mapeados por estas coordenadas em 2D, há algum consenso em mapear algumas regiões como ombros, cotovelos e joelhos. Entretanto, a distribuição exata dos pontos de interesse da pose é bastante variável, pois depende do

modelo de treinamento utilizado e da implementação específica. Como pode ser visto no exemplo da Figura 19, há modelos como o COCO que destacam características adicionais como olhos e nariz, enquanto que outros optam por extrair somente o topo da cabeça.

Figura 19: Comparação de dois modelos de estimativa de pose

Fonte: (GUPTA, 2018)

5.2 MUDANÇA DE ESCALA PARA *PIXEL ART*

Como foi visto na seção 2.2 o estilo de *pixel art* fornece um visual único capaz de cativar o jogador e provocar sentimento de nostalgia. Entretanto, é possível encontrar aplicações que trabalham o *pixel art* não como aparência final mas como meio para criar outra estética mais elaborada. Essa necessidade se encaixa no estudo de algoritmos para aumentar a resolução de imagens. Nesta seção serão visto alguns fatores motivadores para isso e exemplos de processos que podem ser utilizados.

O uso de *pixel art* somente como base da arte inicial pode ocorrer em alguns cenários distintos. Um deles ocorre quando um jogo que foi criado para determinada plataforma é executado em outro console com arquitetura diferente, por exemplo ao utilizar um computador atual para executar algum artefato que foi desenvolvido nas décadas iniciais dos jogos digitais. Nestas situações, é utilizado um emulador para prover ao sistema atual as funções que o jogo necessita para rodar e que não estão disponíveis de forma nativa. Além do aspecto estrutural, muitos emuladores disponibilizam recursos que não existiam no console original, como, por exemplo, tradução do texto ou salvar o estado do jogo a qualquer momento para retomar depois. Além disso podem trabalhar também as características visuais, transformando a imagem vista pelo jogador. No caso de jogos mais antigos ou para dispositivos móveis, que são projetados para telas menores, a execução em outra plataforma pode tornar os pixels evidentes demais e comprometer a satisfação gerada pelo *game*, como por exemplo se um jogo feito para uma tela de 16x16 pixels fosse executada em uma tela Full HD, que tem 1920x1080 pixels. Desta forma o emulador pode

aplicar técnicas para adaptar também a resolução para a plataforma de destino e resolver este problema. (CONLEY *et al.*, 2003)

Outra possibilidade de mudança de escala ocorre quando é necessário analisar ou processar imagens que foram obtidas com baixa resolução. Por exemplo, fotografias de satélite capturadas a altas distâncias, podem receber, além da ampliação óptica, um processamento digital para facilitar a análise pelos cientistas. No contexto da indústria de jogos, aumentar a escala do *pixel art* ocorre como preparação ao utilizá-lo em algoritmos de processamento digital de imagens, e este é o contexto onde este projeto está inserido. Neste caso, a dificuldade ocorre porque cada pixel existente foi posicionado manualmente pelo artista e portanto há muita informação agregada. Assim, mudanças feitas por um algoritmo podem exercer um impacto mais significativo do que em imagens que tenham resolução maior, evidenciando quaisquer erros que houve. Para trabalhar esse problema uma das soluções é aumentar a resolução do *pixel art*, para então efetuar os processamentos e após reduzir o tamanho novamente. Com isso os defeitos pontuais que forem inseridos podem ser eliminados ao reduzir o tamanho da imagem e manter somente os aspectos mais evidentes. (JURIO *et al.*, 2011)

Dito isso, existem diversos estudos que verificam formas de transformar a resolução de imagens, desde processamentos tradicionais direcionados a resoluções maiores que não têm muita preocupação em preservar detalhes, até lógicas específicas para *pixel art* que controlam tanto a resolução quanto a paleta de cores. O primeiros algoritmos deste gênero fazem a varredura da imagem de forma semelhante a uma convolução (vista na subseção 4.3.1) e geram a saída amplificada. Entre as implementações mais conhecidas que utilizam este modelo estão os filtros 2xSaI (FA, 1999), Scale2x (MAZZOLENI, 2001) e a família HQX (STEPIN, 2001).

Enquanto isso, há outros filtros utilizam técnicas mais sofisticadas que não operam diretamente sobre os pixels da imagem. Um exemplo disso é o projeto *Feature-Aware Pixel Art Animation* (KUO; YANG; CHU, 2016) em que os autores aplicaram técnicas de vetorização buscando extrair as linhas principais da imagem. Além disso utilizaram malhas para mapear as diferentes regiões coloridas e com isso não precisam trabalhar diretamente na imagem mas sim nessa representação da mesma. A partir disso desenvolveram operações de rotação e deformação do *pixel art* sem distorcer os traços originais.

Outra implementação deste conceito de transformação de escala foi proposta no projeto *Depixelizing Pixel Art* (KOPF; LISCHINSKI, 2011), que utilizou uma abordagem bastante original. O algoritmo criado começa gerando uma malha virtual que conecta cada pixel com os 8 adjacentes. Analisando as cores da imagem, identifica-se os pares de pixel onde há maior contraste e são eliminadas as suas conexões. Com isso já é possível extrair alguns padrões como linhas de contorno principais e regiões que tem cores semelhantes. Com mais alguns procedimentos que limpam ainda mais a malha virtual e eliminam am-

biguidades dessa representação, é criado praticamente um desenho vetorizado que poderá ser escalado para qualquer resolução. Além disso é feito um procedimento para controlar as cores da imagem que ficaram nas regiões semelhantes citadas anteriormente. Para melhorar a resolução nestas áreas o algoritmo proposto faz uma transição suave de cores.

Como foi possível observar, existem aplicações práticas que se beneficiam da mudança de escala de uma imagem em *pixel art* e diferentes técnicas para cumprir essa tarefa. Enquanto que as implementações mais modernas que se encontram no estado da arte desta área ainda estão sendo aprimoradas e são disponibilizadas de forma mais restrita, outras opções mais consolidadas são de uso livre e já foram aplicadas em diversos tipos de emuladores de jogos.

5.3 TRADUÇÃO IMAGEM-PARA-IMAGEM UTILIZANDO REDES NEURAIS

A transformação ou tradução de imagens (do inglês *image-to-image translation*) é o nome dado à aplicação prática dos conceitos de redes neurais artificiais geradoras vistas na subseção 4.3.2, onde foi destacado em específico a arquitetura de uma GAN. Foi visto que este tipo de RNA faz uma descoberta sobre a distribuição estatística dos parâmetros de entrada, permitindo que esta rede gere novas saídas que repetem este padrão e se assemelham às entradas originais. Como viu-se as GAN podem ser usadas para gerar qualquer tipo de dado na saída, seja uma série de números, texto, áudio ou imagens. Enquanto isso, as técnicas de tradução de imagens são específicas para lidar com imagens, tendo como objetivo principal descobrir uma função que mapeie uma imagem de entrada para uma imagem de saída diferente.

É interessante observar que esse objetivo é uma junção dos conceitos que fundamentam as CNN (subseção 4.3.1) e as GAN (subseção 4.3.2), visto que a primeira delas consegue extrair dados a partir de imagens, enquanto que a segunda consegue gerar imagens a partir de um treinamento prévio e dados aleatórios de entrada. Ou seja, os dados que a primeira delas consegue extrair podem ser usados como entrada para geração de novas imagens, ao invés de informações aleatórias. As aplicações de tradução de imagens são variadas, servindo para fazer transferência de estilo, melhoria de fotos e segmentação automatizadas (AMBERER, 2019).

As aplicações que fazem essa transformação de imagens utilizam um processo composto intrinsecamente pelos componentes que foram vistos anteriormente no Capítulo 4 que trata de redes neurais, porém é válido destacar a lógica e as aplicações possíveis que diferenciam essa área de estudo. As implementações que fazem essa conversão gráfica podem ser classificadas em dois tipos de aprendizado: pareado e não-pareado. No primeiro destes modelos é necessário que a imagem de entrada e a de saída sejam explicitamente vinculadas em pares para que a rede neural possa fazer suas validações internas e gerar o aprendizado. Enquanto isso, o aprendizado não-pareado precisa apenas da separação

entre qual é o conjunto de imagens de entrada e quais são as imagens de saída desejadas. (TRIPATHY; KANNALA; RAHTU, 2018)

Uma implementação que aplicou o conceito de aprendizado não-pareado foi demonstrada no artigo *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks* (ZHU *et al.*, 2017), onde os autores explicam a lógica por trás deste modelo. No aprendizado não-pareado não existe vinculação direta entre as imagens de entrada, e pode parecer impossível extrair qualquer informação desta forma. Porém torna-se possível visualizar esse processo pensando num problema de tradução linguístico explicado a seguir. Português e inglês são duas línguas com características próprias que as diferenciam. Uma frase em português pode ser traduzida para o inglês seguindo uma série de regras e padrões gramáticos, gerando uma nova frase válida em inglês. Espera-se que esta frase possa ser traduzida de volta para o português e que o resultado disso seja exatamente igual à frase em português original. Desta forma as operações de tradução não alteram o sentido da frase mesmo que sejam aplicadas uma após a outra.

Desta forma, os autores modelaram o aprendizado não-pareado trabalhando os conjuntos de entrada e saída da rede como se fossem duas linguagens distintas. Assim cria-se duas funções: uma que converte a imagem de entrada para saída, e outra que converte uma saída para entrada. Assim como explicado no exemplo de tradução de frases, é colocada uma restrição de que essas duas funções devem se anular quando aplicadas em sequência. Ou seja, traduzir a imagem de um conjunto para o outro e depois retorná-la ao conjunto original não pode alterar sua aparência. Além disso, a função deve gerar uma imagem válida no contexto do grupo de destino. Desta forma, a rede aprende a calibrar essas funções de mapeamento de imagem para atender às restrições propostas e recebeu o nome de ciclo-consistente pelo fato de permitir tradução da imagem nos dois sentidos. (ZHU *et al.*, 2017)

Outra aplicação de transformação de imagens está no artigo *Few-shot Unsupervised Image-to-Image Translation* (LIU *et al.*, 2019), em que é explicado outro método de aprendizado utilizado para tradução de imagens. Neste projeto foram desenvolvidos modelos que transformam um animal qualquer aplicando a ele características de outra criatura. Por exemplo, é possível apresentar uma imagem de um cachorro Border Collie em determinada posição e na saída obter praticamente a mesma imagem porém com um cachorro da raça São Bernardo. Foram propostos três componentes para conseguir fazer isso: um codificador de conteúdo, um codificador de classes e um decodificador que recebe dados de ambos e gera uma saída. O codificador de conteúdo extrai informações sobre a aparência geral da imagem, como posição e expressão do animal, enquanto que o codificador de classes extrai as características que são próprias de cada espécie como cor e tipo de pelagem. O decodificador de saída recebe essas informações e gera uma imagem, sendo incentivado pela lógica de aprendizado a reconstruir exatamente a mesma imagem

de entrada quando ela for usada tanto na extração de conteúdo quanto na extração de classes. A forma de aplicação após o treinamento consiste em substituir a entrada do codificador de classes por imagens do animal de destino, de forma que o decodificador faz a mistura das novas classes com o conteúdo da imagem de entrada.

Outro exemplo que avançou nessa abordagem foi criado pela NVIDIA (KARRAS; LAINE; AILA, 2018). Neste projeto o objetivo era misturar faces humanas de forma semelhante ao processo que foi comentado acima, ou seja, aplicando a aparência de uma pessoa ao rosto de outra. Porém neste caso a rede proposta conseguia tratar de forma independente e parametrizável as diferentes características a serem aplicadas na imagem de saída. Dessa forma, permite definir por exemplo quanto da posição, estilo de rosto e cores da imagem de referência serão afetados pela transformação, bem como utilizar diferentes imagens como referência para cada uma dessas transformações. Isso abre a possibilidade para maiores controles no processo de transformação de imagens.

Com tudo isso, nota-se que a tradução imagem-para-imagem possui aplicações práticas variadas e fundamentação em conceitos de inteligência artificial, bem como lógicas de processamento próprias que diferenciam esta área de outras do aprendizado de máquina. A seguir serão vistas implementações específicas dos conceitos abordados neste capítulo que imagina-se serem possíveis de aplicar neste projeto.

6 IMPLEMENTAÇÕES PRÁTICAS PARA PROCESSAMENTO DE IMAGENS

Neste capítulo será visto em mais detalhes os trabalhos que se relacionam com este projeto, os quais aplicam os conceitos explicados até agora. Enquanto que até então houve enfoque na teoria geral, aqui serão abordadas as ferramentas específicas OpenPose, os filtros HQX e o tradutor imagem-para-imagem Pix2Pix, que pretende-se utilizar durante o desenvolvimento prático deste projeto.

6.1 TÉCNICA APLICADA NA DETECÇÃO DE POSE

Anteriormente foi visto, na seção 5.1, que existem diversas técnicas consolidadas para detecção de pose e há modelos variados de mapeamento do corpo humano disponíveis. Nesta seção será vista uma aplicação prática destes conceitos, que ficou conhecida como OpenPose, apresentando suas características específicas e ao final justificando a escolha para ser usada neste projeto.

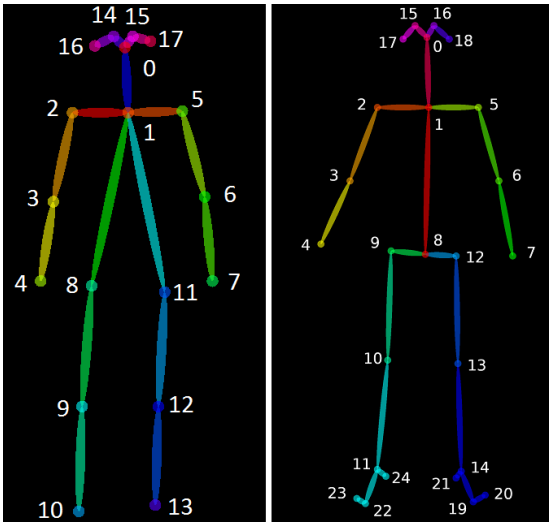
O OpenPose é um sistema de detecção de pose desenvolvido por Cao *et al.* (2018) que adota a sistemática *bottom-up*, ou seja, parte do conjunto de todos os pontos detectados para efetuar a montagem das poses individuais. A escolha dessa abordagem foi explicada pelo fato de que há alguns problemas com a opção *top-down*. Por ser necessário inicialmente segmentar a imagem entre os diferentes indivíduos, esta etapa torna-se crítica do processo, pois a qualidade geral da pose dependerá do quão bem for feito este recorte da imagem. Além disso, há um tempo gasto nessa segmentação e, por fim, ela não resolve totalmente o problema de haver oclusões ou sobreposição de membros. Cao *et al.* (2018)

A partir dessa escolha do modelo *bottom-up*, foi criada uma versão própria de uma CPM, que conforme foi visto na seção 5.1 é o componente que otimiza o processo de mapeamento dos pontos. Esta implementação introduziu o conceito dos chamados *pair affinity fields* (campos de afinidade de pares, tradução nossa). Essa técnica avalia o fluxo de cores da imagem para verificar quais partes estão mais relacionadas e com isso melhora a previsão da posição e conexão de cada parte do corpo com os membros adjacentes. Isso acaba sendo importante para no final conectar corretamente os membros de um mesmo indivíduo sem misturar o corpo de uma pessoa com o de outra. (CAO *et al.*, 2016)

Assim como ocorre com outros modelos, os padrões de pose gerados pelo OpenPose dependem da base de dados que for usada para geração dos mesmos. Ou seja, o conjunto de pontos de saída pode ser escolhido e há três opções que são utilizadas: MPII, COCO e BODY_25. Com relação às diferenças entre estes modelos, o MPII mapeia somente um ponto para o topo da cabeça e possui um ponto central para a articulação da coluna, enquanto que o modelo COCO não possui este ponto mas mapeia as orelhas, olhos e nariz.

A terceira opção ainda não recebeu um nome definitivo mas foi denominada BODY_25. Trata-se de uma extensão do modelo COCO feita para mapear também os pés e o centro do quadril. Desta forma, contém 7 pontos adicionais aos 18 do modelo original. Uma comparação entre o modelo COCO e essa extensão pode ser vista na Figura 20. (CAO *et al.*, 2018)

Figura 20: Modelos de mapa de pose COCO e BODY_25



Fonte: (CAO *et al.*, 2018)

Por fim, como o nome da ferramenta indica, o OpenPose é um sistema de detecção de pose que teve o código fonte liberado publicamente através do GitHub, que é um repositório online para projetos de desenvolvimento de software. Além disso, é compatível com o sistema operacional Windows e pode ser executado mesmo em máquinas sem placa gráfica, apesar de que com menor desempenho. Por tudo isso e pelo fato de ter suporte a um modelo de saída que mapeia tanto pontos do rosto quanto dos pés, escolheu-se esta ferramenta para ser utilizada no decorrer deste projeto, entendendo que estes diferenciais serão importantes no desenvolvimento prático.

6.2 TÉCNICA APLICADA NA MUDANÇA DE ESCALA PARA *PIXEL ART*

De acordo com o que foi apresentado na seção 5.2, a mudança de escala de imagens possui várias aplicações práticas, dentre elas a possibilidade de facilitar o tratamento do *pixel art* em algoritmos de processamento de imagens. Nesta seção será visto uma ferramenta que faz esta tarefa e que se chama HQX. Inicialmente é apresentado seu funcionamento e vantagens para então elencar os motivos que fundamentam sua aplicação neste projeto.

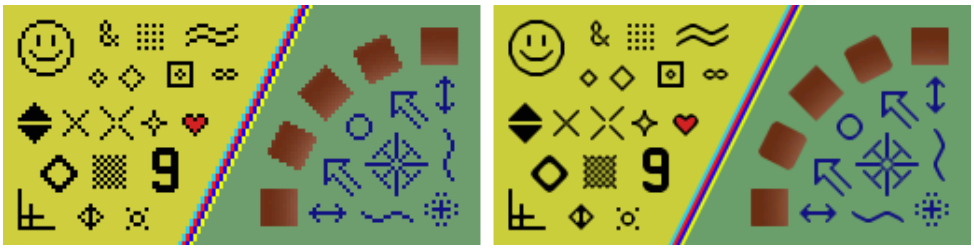
A família de filtros ampliadores HQX foi desenvolvida por Stepin (2001) e oferece opções para aumentar a imagem em 2, 3 ou 4 vezes. Cada filtro recebeu um nome próprio, como, por exemplo, o hq2x para a ampliação de 2x. O processo proposto foi desenhado

especificamente para uso com *pixel art* mas não tem preocupação em manter a paleta de cores original. É por natureza simples, pois se baseia diretamente nos pixels da imagem fazendo uma varredura sobre os mesmos, sem abstrações mais complexas.

Para cada pixel que for lido, são verificados os demais adjacentes a ele, semelhante a uma convolução com *kernel* 3x3. Utilizado um algoritmo de comparação de cores e um *threshold* (limiar de referência) é definido se cada um destes 8 pixels tem a mesma cor que o central. Como cada um pode ser igual ou diferente, existem 256 combinações possíveis. Todas estas situações estão previstas em uma tabela interna que descreve como expandir o pixel central em uma nova matriz de 2x2, 3x3 ou 4x4 que melhor represente o contexto do pixel original. Após isso é aplicada uma técnica de *anti-aliasing*. (STEPIN, 2001)

O *anti-aliasing* é um procedimento que visa suavizar a transição de cores em uma imagem. No caso do *pixel art* ou outras imagens digitais, serve para eliminar o serrilhamento. Utilizando um exemplo prático, ao observar um círculo puramente preto em um fundo branco em um monitor, seria possível perceber os pixels que vão formando a curvatura, tornando-a ligeiramente quadriculada. Porém, com um *anti-aliasing* aplicado, estes pixels mais próximos da borda das cores são borrados de forma a dar a impressão de curvatura suave. Nos filtros HQX, esse processo é feito após aumentar a escala da imagem pois se fosse aplicado diretamente no *pixel art* causaria um desfoque grande, comprometendo o entendimento da figura. O resultado de uma aplicação completa do filtro pode ser visto na Figura 21, onde a imagem da esquerda está ampliada com o padrão *nearest neighbor* que mantém as cores originais e torna os pixels evidentes, enquanto que na imagem da direita o filtro HQX foi utilizado na ampliação.

Figura 21: Aplicação do filtro hq3x



Fonte: (STEPIN, 2001)

A vantagem deste filtro é que ele não exige componentes externos para funcionar, como por exemplo bibliotecas de processamento gráfico. Mesmo sendo original de 2001, essa técnica foi reescrita em diversas linguagens diferentes e para diferentes sistemas operacionais, o que evidencia sua eficácia no problema que busca resolver. É possível, inclusive, automatizar chamadas por linha de comando para aplicação deste filtro. Além disso, é um filtro de uso livre já aplicado em diversos emuladores de jogos. Por estes motivos, foi escolhido como ferramenta para ser utilizada no desenvolvimento prático deste projeto.

6.3 TÉCNICA APLICADA NA TRADUÇÃO IMAGEM-PARA-IMAGEM

De acordo com o que foi visto na seção 5.3, o processo de tradução imagem-para-imagem utiliza redes neurais geradoras (GAN) e convolucionais (CNN) para extrair informações e representá-las em um novo formato. Nesta seção será visto uma implementação prática desta técnica conhecida como Pix2Pix, abordando a história da sua criação e ao final os motivos que embasam a aplicação neste projeto.

No artigo *Image-to-Image Translation with Conditional Adversarial Networks* proposto por Isola *et al.* (2017) foi observado que os problemas de tradução imagem-para-imagem eram resolvidos com algoritmos e implementações específicos para cada situação, porém a questão no final era sempre mapear um conjunto de pixels de entrada para um conjunto de pixels de saída. A partir disso se propuseram a desenvolver um *framework* que fosse viável de aplicar em todas as situações.

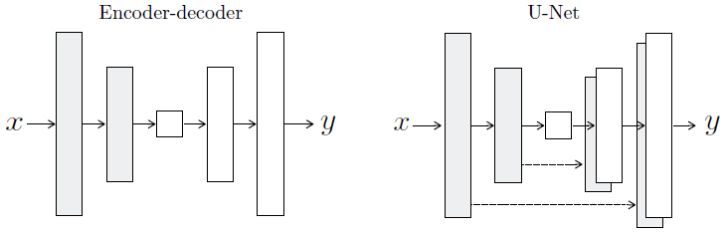
Como dito, a abordagem escolhida foi a utilização de CNN e GAN e explicou-se o motivo dessa escolha. Redes CNN já eram aplicadas para tarefas de geração de imagens, mas apesar de o processo de aprendizado ser automático, ainda era necessário algum esforço manual para definir a função de custo (seção 4.2) que pontua a qualidade dos resultados da rede. Uma boa função de custo seria aquela que faz a CNN para reduzir a diferença entre os pixels da imagem de referência esperada e a imagem que foi gerada pela rede, porém isso acaba gerando imagens borradas já que a rede tenta equilibrar todas as saídas plausíveis em uma única imagem. (ISOLA *et al.*, 2017)

A aplicação de GAN (subseção 4.3.2) neste contexto permite que a função de custo seja definida dinamicamente conforme a rede vai aprendendo a diferenciar as imagens geradas das imagens verdadeiras. Isso porque a rede vai identificando quais são as características importantes para o problema de tradução imagem-para-imagem específico. Por exemplo, em um problema de coloração de imagens, a distribuição das bordas deve ser mantidas e as cores precisam ser alteradas, mas isso pode ser irrelevante para outros tipos de problemas. Os autores deste projeto aplicaram um tipo específico de GAN chamado *Conditional Generative Adversarial Networks* (cGAN), em que a saída a ser gerada é condicionada a um dado de entrada ao invés de ser construída a partir de um ruído aleatório. Neste caso, o dado de entrada é um mapa das características extraídas da imagem de entrada que devem servir de base para a geração da saída. (ISOLA *et al.*, 2017)

Neste projeto também foi feita uma melhoria no gerador de imagens, pra que não dependa somente destes dados abstratos para construir a saída. Devido à combinação de uma CNN que faz convoluções cada vez menores, e a GAN que vai deconvolucionando dados para gerar informações, o diagrama de uma rede tradutora de imagens geralmente se assemelha a uma ampulheta deitada. Essa construção é chamada rede codificadora-decodificadora (do inglês *encoder-decoder*) e pode ser vista na Figura 22.

A melhoria apresentada neste projeto foi aplicar uma especialização deste tipo de rede, chamada U-Net. Nela, cada camada interna de entrada se conecta com a respectiva camada de saída, permitindo que algumas informações sejam repassadas diretamente. Isso resolve o problema do gargalo de processamento central, onde seria necessário trafegar todas as informações. Com essa conexões diretas entras as camadas, o aprendizado separa as informações que podem trafegar diretamente daquelas que realmente precisam de processamento. Por exemplo, no caso do problema de coloração de imagens em preto-e-branco, os dados das bordas e luminosidade de cor passam diretamente para a imagem de saída enquanto que a rede interna precisa se preocupar apenas com segmentar as regiões e gerar novas cores realistas para cada uma delas. (ISOLA *et al.*, 2017)

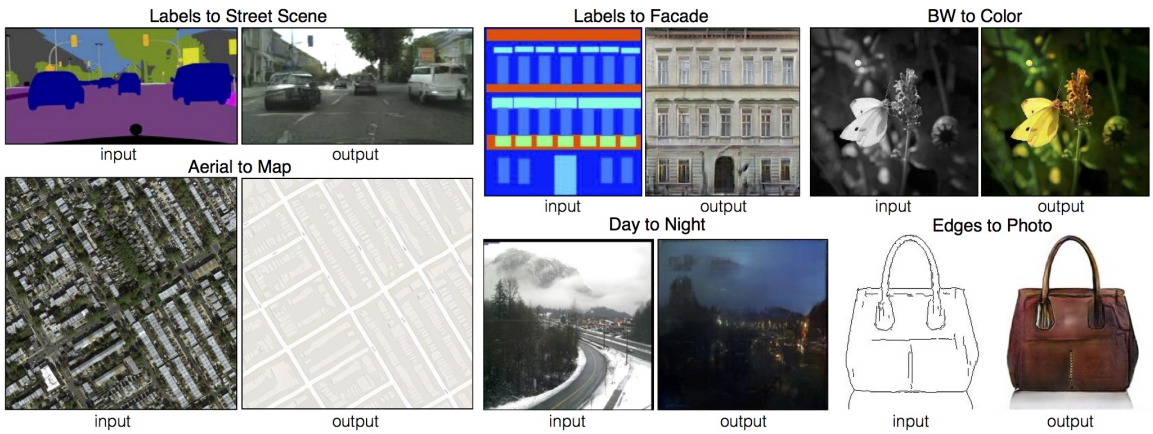
Figura 22: Rede codificadora-decodificadora e a U-Net aplicada no Pix2Pix



Fonte: (ISOLA *et al.*, 2017)

Com toda essa estrutura, o funcionamento foi posto à prova com alguns problemas de transformação de imagens que podem ser vistos na Figura 23, que destaca as diversas possibilidades atendidas pelo mesmo *framework*. O modelo desenvolvido por este projeto foi denominado Pix2Pix e disponibilizado publicamente no GitHub. Desta forma, várias pessoas puderam ter acesso a este estudo e fazer experimentos próprios. Por esta característica de aprendizado que abstrai detalhes técnicos de projeto da rede neural e pelo fato de ser acessível ao público geral, selecionou-se o Pix2Pix para ser aplicado no desenvolvimento deste trabalho.

Figura 23: Exemplos de tradução imagem-para-imagem utilizando Pix2Pix



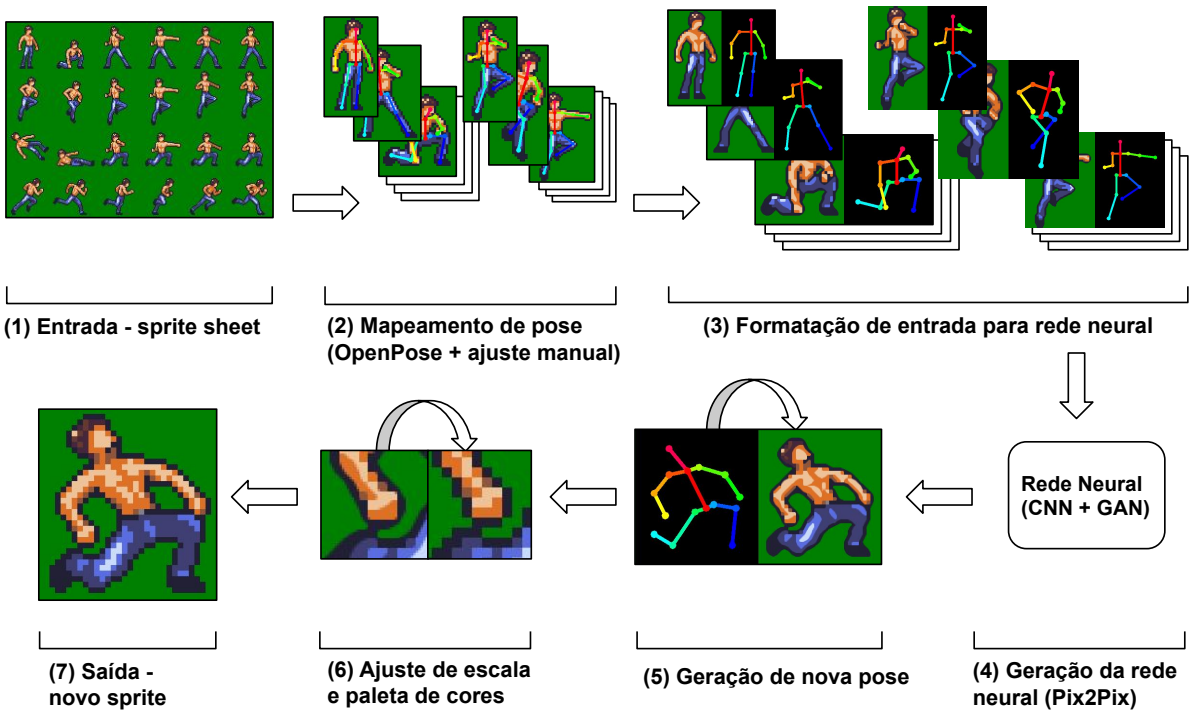
Fonte: (ISOLA *et al.*, 2017)

7 DEFINIÇÃO DO PROCESSO DE GERAÇÃO AUTOMATIZADA DE IMAGENS

Com tudo que foi abordado até agora, esta pesquisa definiu um processo que possibilita o desenvolvimento dos objetivos aplicando os conceitos e as ferramentas apresentados anteriormente. A lógica proposta é ilustrada na Figura 24, onde se destaca também os ajustes necessários para integração entre as diferentes ferramentas utilizadas. Inicialmente será explicado o processo através de uma visão geral, para então conceituar detalhadamente cada etapa nas próximas seções. Visto isso, os capítulos seguintes abordarão a forma prática como foi implementado cada um destes estágios.

O processo inicia com a entrada de um *sprite sheet* (definido na seção 2.2) que possui todas as figuras do personagem que deseja-se utilizar no treinamento da rede neural, o qual pode ser visto na Figura 24 no bloco (1). Na etapa (2), será extraída a pose para cada um dos *sprites* (utilizando a técnica apresentada na seção 6.1), sendo efetuados ajustes manuais para refinar essa saída. A etapa (3) finaliza a preparação para o treinamento da rede neural, gerando novas imagens individuais que vinculam cada pose com o respectivo *sprite* de origem. Ainda nesta etapa, será aplicado o filtro para mudança de escala visto na seção 6.2 para prover à rede neural uma entrada de dados mais suavizada.

Figura 24: Processo de mudança da pose de um personagem



Fonte: elaborado pelo autor, sprites adaptados de “Hero spritesheets (Ars Notoria)” (OpenGameArt.org)

Na etapa (4), essas imagens serão utilizadas como entrada do *Pix2Pix* (conforme definido na seção 6.3) para construir a rede neural tradutora imagem-para-imagem, onde se concentram também os fundamentos apresentados no Capítulo 4. Após concluir o treinamento, na etapa (5) poderá ser enviada uma nova pose que nunca foi processada pela rede durante o treinamento e a rede neural deverá gerar como saída uma nova imagem coerente com o conjunto de treinamento, mas que não será ainda um *pixel art* finalizado. A próxima etapa, representada no bloco (6), fará a conversão da escala e a aplicação da paleta original de forma que a imagem de saída da rede neural fique com as mesmas proporções e as mesmas cores do conjunto de treinamento. Após tudo isso, na etapa (7) será obtida uma imagem em *pixel art* em que o artista poderá fazer retoques e finalizar o desenho. Com este resumo, as próximas seções deste capítulo detalham conceitualmente as diferentes etapas do processo proposto, seus requisitos e os cuidados necessários. O desenvolvimento prático destes conceitos será apresentado posteriormente no Capítulo 8.

7.1 ENTRADA DE DADOS

Tendo sido abordada a visão geral do processo, podem ser definidos os requisitos específicos da etapa de entrada de dados. É ela que dá início ao processo de criação de novas imagens e servirá de base para todas as etapas seguintes, portanto é importante para definir o escopo das imagens que este projeto pretende trabalhar.

Primeiramente, o processo aqui proposto é limitado a imagens construídas em *pixel art* que representem um mesmo personagem com características constantes no decorrer das animações. Ou seja, cuja constituição física, roupas, cores e design geral sejam coerentes entre todas as imagens. Além disso, como viu-se na seção 6.1, será utilizada uma ferramenta de detecção de pose que é baseada em seres humanos, portanto o processo é mais adequado para personagens que tenham características humanoides. A próxima seção apresentará como pode ser flexibilizada essa restrição ao aplicar a técnica neste contexto.

Por fim, é necessário reunir imagens suficientes para ilustrar as características principais do personagem, a fim de que possam ser assimiladas pela rede neural. Um *sprite sheet* completo fornece uma determinada quantidade de imagens, das quais pode ser selecionado um conjunto ou todas para servirem de base para geração de novos *sprites*. Não é possível definir uma quantidade exata de imagens necessárias, porém é possível aumentar a quantidade de repetições de treino até que o resultado da RNA seja satisfatório. Assim como foi visto na seção 4.2, a quantidade exata de épocas de treino também é variável conforme a aplicação. Dito isso, a próxima seção destaca como segmentar essas informações, preparando-as para a construção da rede neural.

7.2 MAPEAMENTO DE POSE

Esta etapa visa extrair informações do *sprite sheet* completo para transformar cada imagem complexa do personagem em uma abstração, ou seja, uma simplificação que poderá ser trabalhada posteriormente para criar novas posições. É necessário definir um modelo simples que represente o personagem e que possa ser gerado depois como entrada para rede neural. Assim como explicado na seção anterior, também é importante que o modelo de mapeamento escolhido seja coerente entre todas as imagens. Ou seja, quanto mais criterioso for o padrão lógico de relacionamento entre imagem e modelo, tanto melhor será a qualidade das informações de entrada da rede neural. Em termos práticos, se foi definido que o olho esquerdo do personagem é representado por um ponto roxo, esse ponto não pode ser utilizado para outra finalidade e deve estar sempre posicionado exatamente sobre o olho esquerdo do personagem.

Nesse sentido, como este projeto envolve a aplicação de algoritmos de detecção de pose que já possuem um mapa de pontos saída definido (conforme visto na seção 6.1), convém utilizar este mesmo modelo para mapear as imagens, ou seja, relacionando cada parte do corpo com a representação da pose em forma de esqueleto. Entretanto, nada impede o uso de modelos corporais diferentes, sendo que neste caso podem ser buscadas outras técnicas de detecção de pose ou mesmo feito o mapeamento de forma totalmente manual. Como o objetivo deste trabalho está relacionado com algoritmos de detecção de pose para humanos, não serão abordadas formas de mapeamento para outros tipos de imagens com estrutura completamente distinta, como animais ou objetos.

Dito isso, percebe-se que o uso de uma rede neural tradutora imagem-para-imagem torna bastante flexível o método de mapeamento das imagens. Qualquer que seja o modelo proposto, basta que ele seja claro e seguido entre todas as imagens para que a rede neural possa interpretar o padrão estabelecido. É possível simplificar não somente informações sobre a pose do personagem, como também dimensões (variando por exemplo a grossura da linha) ou iluminação (alterando a intensidade das cores), ou ainda outros padrões mais complexos que tragam mais dados ao modelo.

É importante entender que quanto mais complexo for o modelo, mais treinamento será necessário até que a rede neural consiga generalizar uma solução. Conforme foi visto na seção 4.2, uma complexidade maior de aspectos a assimilar se relaciona com mais camadas ocultas na rede neural e com isso atrasa-se o estado de *overfitting* e do desaparecimento de gradiente. Neste projeto não será abordado de forma mais profunda essas possibilidades de modelos mais abstratos, porém será visto na seção 8.2 um exemplo de aplicação que faz com que a rede neural assimile mais dados sobre as cores do que sobre a pose do personagem. (NIELSEN, 2015)

7.3 ENTRADA DA REDE NEURAL

Feita a definição do modelo e mapeamento das imagens a serem utilizadas como treino, o passo a ser abordado nessa seção é a preparação para o treinamento da rede neural. O objetivo desta etapa é fazer a conexão entre o mapeamento feito anteriormente e o formato específico de entrada esperado pela ferramenta de construção de rede neural que for aplicada.

Para fazer isso, é necessário primeiramente ter definido qual será essa ferramenta e estudar suas especificações. Uma vez compreendido seu funcionamento, devem ser elencados os requisitos a serem seguidos para entrada de dados e se há caraterísticas que possam ser aproveitadas para otimizar o processo de treinamento e que não tenham sido inseridas no modelo de mapeamento.

No caso deste projeto, foi escolhido trabalhar com a rede neural tradutora imagem-para-imagem Pix2Pix, portanto é sobre ela que se faz essa análise. A lógica da abordagem utilizando essa ferramenta é treinar a rede neural a partir de exemplos de pares de imagens em um arquivo único contendo uma imagem de entrada e uma imagem de destino em cada metade. Mediante exemplos e treinamento suficientes, é gerada uma rede neural capaz de gerar imagens para novas entradas seguindo o mesmo padrão aprendido. Ou seja, é necessário criar uma série de arquivos de imagem neste formato.

Outra característica desse processo é referente à resolução da imagem de entrada. Devido à arquitetura interna que constrói a rede neural, são utilizadas imagens com dimensão de exatamente 256x256 pixels. Tamanhos diferentes disso são ampliados ou reduzidos através de mecanismos internos que visam adequar ao processamento convolucional que foi conceituado na subseção 4.3.1. Dessa forma, se for passada uma imagem maior, haverá truncamento das informações; enquanto que uma resolução menor ocasiona um sub-aproveitamento do treinamento pois várias regiões da imagem expandida terão informações idênticas. Essa segunda situação é o caso do uso de *pixel art*, em que a ampliação tradicional apresenta transições mais abruptas das cores e formas, conforme visto na seção 5.2. Tento elencado essas duas particularidades sobre o formato e a resolução de entrada, destaca-se que uma forma de trabalhar estes aspectos será vista no desenvolvimento prático do Capítulo 8. A partir da análise e preparação proposta nessa seção, pode ser iniciado o processo de treinamento que é detalhado a seguir.

7.4 GERAÇÃO DA REDE NEURAL

Na etapa de geração da rede neural se concentram os conceitos estudados anteriormente no Capítulo 4 referentes a redes neurais convolucionais (CNN) e redes neurais adversárias geradoras (GAN). Aqui a ferramenta escolhida para geração da rede neural é executada utilizando como entrada as imagens montadas pela etapa anterior. Com isso,

nesta etapa também são executados dois subprocessos para possibilitar o treinamento de forma mais efetiva. O primeiro deles é a configuração do ambiente e o segundo o estudo de parâmetros e configurações disponíveis que possam ser aproveitados.

A configuração do ambiente é necessária para permitir a geração da rede neural. Devem ser verificados os requisitos de sistema, dependências com recursos de terceiros e demais condições para executar a ferramenta escolhida. No caso do Pix2Pix que será aplicado neste projeto, o ambiente padrão conforme documentação é o Linux ou OSX, porém o site *Machine Learning for Artists* já estudou uma forma de disponibilizar a plataforma em sistema operacional Windows e instrui quanto às dependências e o processo de instalação (KOGAN; TSENG; REFSGAARD, 2016). Desta forma, percebe-se que seguindo estas instruções será possível rodar o Pix2Pix neste ambiente para efetuar o treinamento.

Por fim, o estudo de parâmetros visa aproveitar as características específicas da ferramenta escolhida. No contexto deste projeto, a ferramenta Pix2Pix disponibiliza dois recursos interessantes de se aplicar durante o treinamento: a escolha da quantidade de épocas de treino e a possibilidade de salvar pontos de controle (do inglês *checkpoints*). A escolha da quantidade de épocas de treino já foi comentada anteriormente como importante para calibrar a qualidade dos resultados, sendo pauta para experimentação para identificar a quantidade mais adequada. Enquanto isso, o recurso do ponto de controle facilita esses testes porque permite salvar o resultado do treinamento e retomá-lo posteriormente para executar mais épocas se necessário. Tendo entendido o funcionamento dessa ferramenta, a rede neural é construída a partir da entrada montada anteriormente, para que na próxima seção seja visto como aplicá-la.

7.5 APLICAÇÃO DA REDE NEURAL

Após construir a rede neural a partir do treinamento, essa etapa visa aplicá-la para geração de novas imagens. Para tal, é necessário retornar ao modelo de mapeamento de pose construído na seção 7.2 e criar novas posições a serem geradas. Em um contexto prático não haverá referência exata para seguir, sendo possível criar poses livremente conforme a necessidade de novas imagens. Entretanto, como neste projeto pretende-se validar os resultados, as novas poses são escolhidas com base em imagens conhecidas do personagem, as quais não fizeram parte do treinamento e portanto não influenciaram a construção da rede neural.

Nas etapas anteriores, identificou-se como necessário estudar as características de entrada e de geração da rede neural utilizada. Da mesma forma, neste momento também é importante verificar os parâmetros exatos de saída da rede neural, identificando as restrições que podem haver e a necessidade de ajustes na saída, que serão implementados na próxima etapa. No cenário deste projeto que aplica a rede neural tradutora

imagem-para-imagem Pix2Pix, não são observadas configurações adicionais de saída que possam ser aplicadas, porém, nota-se a mesma situação que ocorre na entrada de dados: a saída é gerada com exatos 256x256 pixels de resolução independente do tamanho da imagem de entrada, desta forma sendo preciso adequar a resolução para reestabelecer as dimensões originais. Além disso, as cores da imagem também precisam ser tratadas para permanecerem coerentes com o *sprite-sheet* do personagem. Tendo identificadas essas duas necessidades, pode ser efetuada a próxima etapa de normalização da imagem.

7.6 NORMALIZAÇÃO PARA *PIXEL ART*

A última etapa do processo visa implementar ajustes na imagem para normalizar situações que não tenham sido tratadas pela geração da rede neural. O requisito inicial desse momento é ter identificado quais são essas características a serem alteradas na imagem e o resultado esperado é a saída final do processo, com imagem pronta para ser trabalhada pelo artista.

Como foi visto anteriormente, com o uso do Pix2Pix é necessário ajustar a resolução da imagem e as suas cores. Com relação a resolução, basta fazer o processo inverso de ampliação que foi indicado para entrada da rede neural na seção 7.3, enquanto que as cores podem ser normalizadas obtendo a paleta de cores do personagem e aplicando-a sobre a imagem gerada pela rede neural. Além disso, outros tratamentos que visem melhorar a coerência da imagem podem ser efetuados, como, por exemplo, trabalhar os contornos do personagem para garantir que tenham exatamente 1 pixel de largura, aplicar filtros para reduzir ruído da imagem gerada ou até mesmo reconstruir completamente a imagem a partir da saída da rede neural utilizando técnicas mais complexas. Porém, definições mais específicas desses tratamentos dependem fortemente do contexto em que está sendo aplicado este processo. Como este projeto se propõe a abordar uma situação mais genérica, serão desenvolvidos os processos que podem aplicados em qualquer caso, que são o ajuste de escala e a correção da paleta de cores. Um exemplo de implementação prática disso está detalhado na subseção 8.1.6.

Com tudo que foi apresentado neste capítulo pode compreendido a teoria do processo em sua visão geral e os detalhes das etapas individualmente. O próximo capítulo aborda uma implementação prática desses conceitos.

8 DESENVOLVIMENTO PRÁTICO DA PROPOSTA

Definido conceitualmente o processo proposto, este capítulo aborda uma forma prática de como pode ser desenvolvido o projeto. É destacado o modo como efetivamente se implementou cada uma das etapas e a abordagem escolhida para lidar com os requisitos apresentados anteriormente. A forma específica como foi construída cada etapa do processo servirá para entender o funcionamento efetivo e os cuidados necessários, mas não é obrigatório que sejam utilizadas as mesmas ferramentas para atingir o objetivo. Podem ser aplicadas outras linguagens de programação, diferentes redes neurais ou outras lógicas de vinculação entre as imagens de entrada e saída, sem que isso interfira no funcionamento geral.

Propõe-se efetuar dois experimentos que passem por todos os estágios do processo e que possam ser avaliados ao final. Cada um deles utilizará o processo de geração de imagens para uma finalidade distinta, a fim de avaliar o desempenho da solução proposta em diferentes contextos e evidenciar algum grau de flexibilidade da aplicação. No que diz respeito à forma de construção prática de cada etapa, cada implementação será apresentada a partir da lógica principal e do funcionamento dos componentes de programação empregados. Como a solução proposta não se limita a uma linguagem de programação ou implementação únicas, os detalhes como lógicas específicas ou o código-fonte na íntegra não serão relevantes para entendimento do processo na prática. Na seção que segue está detalhado o primeiro destes experimentos.

8.1 EXPERIMENTO 1 - GERAÇÃO COMPLETA DE UM *SPRITE*

A primeira proposta prática consiste em gerar um *sprite* completo a partir somente da imagem abstrata de entrada da rede neural, verificando, portanto, a capacidade de gerar imagens sem grande auxílio do artista, além da definição dessa entrada conforme o modelo escolhido. Cada subseção a seguir detalha o desenvolvimento de uma das etapas abordadas durante a definição do processo (Capítulo 7).

8.1.1 Entrada de dados

Para iniciar o processo é necessário um conjunto de imagens que servirão como treinamento para a rede neural (seção 7.1). No contexto do desenvolvimento de jogos, foi visto que esse tipo de imagem é chamado *sprite sheet* e é o resultado do trabalho criativo das empresas desenvolvedoras. É portanto tratado como parte do seu patrimônio, sendo geralmente protegido por direitos autorais e raramente publicado com livre acesso para uso diferente dos fins originais.

Foi desenvolvido portanto um *sprite sheet* específico para uso neste projeto, o qual pode ser divulgado e estudado sem este tipo de restrições. Para esse desenvolvimento foi contatado um aluno do segundo semestre do curso de Tecnólogo em Jogos Digitais da Universidade Feevale, que está portanto inserido no ambiente de estudo e desenvolvimento de jogos. Ele frequenta o Laboratório de produção de jogos do instituto, que conta com softwares voltados a esta área de desenvolvimento e é gerenciado por monitores e voluntários que prestam auxílio a alunos em suas atividades de aulas ou projetos.

Nesse contexto, foi criado pelo aluno citado anteriormente um *sprite sheet* completo para ser usado como recurso deste projeto. De posse dos requisitos que foram especificados na seção 7.1 e das técnicas que ele dispõe para o desenho, além do acesso ao ambiente de desenvolvimento do laboratório de produção de jogos, foram investidas cerca de 40 horas de trabalho. Foi utilizada principalmente a ferramenta Adobe Photoshop, que é um programa de edição avançada de imagens e fotografias também aplicável no contexto do *pixel art*. Com isso, criou-se um *sprite sheet* completo sem interferência dos autores deste projeto, que pode ser visto na íntegra no Apêndice A, que é explicado a seguir.

Com relação a estas especificações para este trabalho, foi considerado um cenário de uso que se aproximasse do real, ou seja, com um personagem que tivesse várias animações compondo uma sequência lógica e suficiente para aplicação em um jogo. Assim sendo, o personagem foi criado com animações de movimentação como corrida, pulo e recuo, além de diversas animações de ataque e uma sequência de vitória ao final. Não houve preocupação em atribuir um nome específico para cada sequência de animação, no entanto, cada *sprite* está numerado para que seja possível identificar separadamente todas as imagens que compõe o *sprite sheet*. Além disso há uma identificação de cor e uma borda que pode estar grifada em destaque para alguns *sprites*. Essa distinção será explicada e será relevante na subseção 8.1.3 deste experimento.

Além desse requisito de que houvesse animações distintas, foi solicitado também uma quantidade específica de *sprites* desenvolvidos. Considerando que pretendia-se utilizar mais que uma centena de imagens como treino e reservar ainda uma quantidade para efetuar os testes, foi solicitada a criação de 200 imagens, das quais 150 serão utilizadas para treinamento e as 50 restantes para validação dos resultados. Com essa etapa concluída, o projeto passou a ter uma entrada de dados para ser processada e avaliada, como será visto a seguir.

8.1.2 Mapeamento de pose

O mapeamento de pose é uma etapa crítica para estabelecer o padrão de qualidade do processo e, como foi visto anteriormente na seção 7.2, é necessário definir um modelo que represente de forma simplificada a imagem a ser gerada. No caso deste projeto que trabalha com detecção de pose já foi estabelecido o uso do modelo BODY_25

da ferramenta OpenPose (vista na seção 6.1). Além disso, foi visto que é possível e recomendado incrementar o modelo com informações adicionais que forem relevantes para o mapeamento.

Sabendo disso, este projeto desenvolveu um software para mapeamento da pose utilizando a linguagem de programação Java. A seguir é detalhado como ocorre esse mapeamento separando em três processos que ocorrem dentro desta etapa: extração da pose, desenho da pose e configuração do modelo.

Nesta etapa em que é necessário trabalhar com imagens, escolheu-se utilizar a classe Java *BufferedImage* que oferece suporte a diversas operações de manipulação gráfica. No primeiro processo a ser abordado, que é a extração da pose, é segmentado o *sprite sheet* completo em imagens individuais através da seleção do usuário. Para cada uma delas, é executado o algoritmo do OpenPose, que recebe uma imagem como entrada e devolve uma lista de mapas de calor, nos quais o ponto mais intenso de cada mapa é a posição mais provável de uma determinada parte do corpo do modelo BODY_25. A lista completa de pontos devolvidos com o nome original e a respectiva tradução é a seguinte:

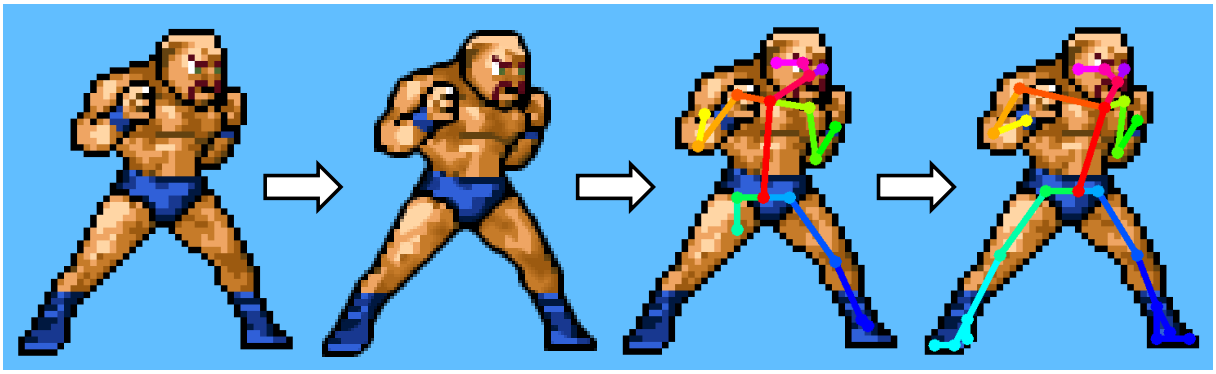
- 0-Nose (nariz)
- 1-Neck (base pescoço)
- 2-RShoulder (ombro direito)
- 3-RElbow (cotovelo direito)
- 4-RWrist (pulso direito)
- 5-LShoulder (ombro esquerdo)
- 6-LElbow (cotovelo esquerdo)
- 7-LWrist (pulso esquerdo)
- 8-MidHip (meio dos quadris)
- 9-RHip (quadril direito)
- 10-RKnee (joelho direito)
- 11-RAnkle (tornozelo direito)
- 12-LHip (quadril esquerdo)
- 13-LKnee (joelho esquerdo)
- 14-LAnkle (tornozelo esquerdo)
- 15-REye (olho direito)
- 16-LEye (olho esquerdo)
- 17-REar (orelha direita)
- 18-LEar (orelha esquerda)
- 19-LBigToe (dedão do pé direito)
- 20-LSmallToe (dedinho do pé direito)
- 21-LHeel (calcanhar esquerdo)
- 22-RBigToe (dedão do pé direito)
- 23-RSmallToe (dedinho do pé direito)
- 24-RHeel (calcanhar direito)

Como a imagem em *pixel art* tende a ser pequena e com isso ficar sensível a erros no mapeamento desses pontos, foi criado um recurso adicional nesta etapa de extração da pose que permite mudar a escala da imagem que é recebida como entrada. Ou seja,

ao invés de analisar diretamente o *pixel art*, é opcionalmente aplicado o filtro visto na seção 6.2 para ampliar a resolução da imagem 2, 3 ou 4 vezes e com isso fornecer uma imagem maior para ser analisada. Aumentar a resolução da imagem pode auxiliar na extração da pose pois, conforme foi visto na seção 5.1, os sistemas de estimativa de pose são baseados em figuras humanas e desta forma o *pixel art* se torna mais semelhante se for ampliado e suavizado por esta técnica.

A partir das informações de coordenadas que são retornadas pelo OpenPose, utilizou-se a classe *Graphics2D* para desenhar pontos e traços sobre a imagem original, sendo possível efetuar ajustes manuais para cada um dos pontos se necessário. Além disso, foi criada uma forma de não exibir os pontos e traços que não seja interessante visualizar, como por exemplo o braço esquerdo do personagem quando visto em visão lateral direita. Na Figura 25, pode ser observado esse processo: a partir da imagem original, é gerada a versão ampliada e tem-se com isso uma estimativa de pose que pode não estar completa nem totalmente adequada. Após isso essa pose inicial é refinada com ajustes manuais e fica adequada à imagem.

Figura 25: Processo de extração da pose com mudança de escala e ajustes manuais



Fonte: elaborado pelo autor

Como nota-se, o mapeamento apesentado até agora está limitado às informações que o modelo BODY_25 consegue retornar. Para deixar o processo mais flexível e permitir mapear informações adicionais, foi criada uma forma de tornar o modelo de mapeamento da pose totalmente configurável. Além de modelos corporais diferentes, essa configuração do modelo também possibilita mapear informações adicionais como a roupa ou cabelo do personagem, que geralmente não serão considerados pelas ferramentas de detecção de pose mas podem ser representadas no modelo para aumentar a qualidade das imagens geradas para treinamento. Para resolver isso, foi criada uma tabela que descreve esse modelo através de uma lista de pontos de pose que pode ser vista na Figura 26. Cada ponto é definido pelas seguintes informações:

1. **ID**: identificação única do ponto de pose. É um número sequencial crescente.

2. **Nome**: identificação descritiva para esclarecer a finalidade do ponto de pose.

3. **Offset X e Offset Y**: posição inicial deste ponto de pose na visualização padrão do modelo, variando entre 0 e 1. Indica, com isso, em qual posição este ponto deve permanecer caso não seja mapeado automaticamente pela detecção de pose. É usado para exibir e configurar uma pose inicial que permita visualizar o modelo de forma clara. Ou seja, conforme o usuário vai posicionando os pontos no modelo, estas duas informações são atualizadas para indicar em qual posição eles se encontram em relação ao espaço ocupado pelo modelo.

4. **Cor**: coloração para diferenciar visualmente este ponto de pose dos demais. Neste projeto em que foi usado como referência o modelo BODY_25, escolheu-se as mesmas cores para os pontos de pose que foram aproveitados diretamente deste modelo. Para os pontos novos criados, foram buscadas novas cores nitidamente distintas das já aplicadas.

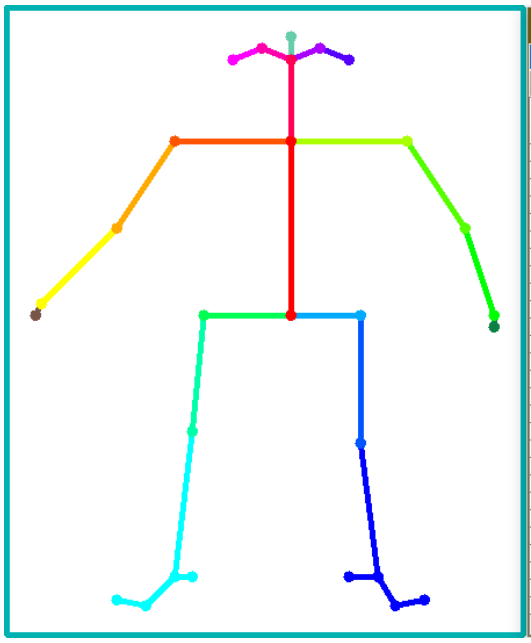
5. ***Link***: código (ID) de outro ponto de pose ao qual este ponto deve se conectar. Essa informação é o que gera o esqueleto do modelo a partir dos pontos. Por exemplo, o ponto do pulso direito se conecta com o cotovelo direito para através desta linha representar o antebraço direito no modelo.

6. **BODY_25 ID**: código do ponto de pose do modelo BODY_25 ao qual este ponto se relaciona. Essa informação é usada para atribuir automaticamente a pose onde for possível.

Além disso, a ordem em que os pontos aparecem nessa tabela proposta indica a ordem em que eles serão desenhados, servindo para mapear a informação de profundidade no caso de o personagem estar com partes do corpo sobrepostas. Ou seja, se um braço do personagem está atrás de outra parte do corpo, pode ser indicado para desenhar os respectivos pontos do esqueleto antes dos demais, de forma que eles sejam sobrepostos por outras partes do corpo que estejam em frente.

A partir dessas informações, o modelo pode ser desenhado sobre o *sprite sheet* e pode receber informações do mapeamento automático de pose BODY_25, sendo possível, portanto, vincular cada *sprite* individual ao sua respectiva abstração conforme o modelo. Na Figura 26 pode ser visto o modelo que será efetivamente utilizado neste experimento, no qual além dos pontos padrão do BODY_25 foi utilizada a configuração de modelo para acrescentar mais três informações: mão direita (ID 3), mão esquerda (ID 7) e topo da cabeça (ID 16). Como estes pontos não possuem relacionamento com o sistema de detecção de pose, eles deverão ser sempre ajustados manualmente. Mesmo assim, foram criados pois agregam informações importantes para que o personagem possa ser mapeado de forma integral.

Figura 26: Tabela de dados para modelo personalizado de mapeamento da pose



Draw Earlier (Up) | Draw Later (Down) | Add | Remove

ID	Name	Off-X	Off-Y	Color	LinkTo	BODY_25 ID
0	RShoulder	0,326	0,213	255,85,0	15	2
1	RElbow	0,213	0,352	255,170,0	0	3
2	RWrist	0,067	0,472	255,255,0	1	4
3	RightHand	0,056	0,491	121,85,71	2	-1
4	LShoulder	0,775	0,213	170,255,0	15	5
5	LElbow	0,888	0,352	85,255,0	4	6
6	LWrist	0,944	0,491	0,255,0	5	7
7	LeftHand	0,944	0,509	10,128,67	6	-1
8	MidHip	0,551	0,491	255,0,0	15	8
9	RHip	0,382	0,491	0,255,85	8	9
10	RKnee	0,36	0,676	0,255,170	9	10
11	RAnkle	0,326	0,907	0,255,255	10	11
12	LHip	0,685	0,491	0,170,255	8	12
13	LKnee	0,685	0,694	0,85,255	12	13
14	LAnkle	0,719	0,907	0,0,255	13	14
15	Neck	0,551	0,213	255,0,0	-1	1
16	HeadTop	0,551	0,046	102,205,170	17	-1
17	Nose	0,551	0,083	255,0,85	15	0
18	REye	0,494	0,065	255,0,170	17	15
19	LEye	0,607	0,065	170,0,255	17	16
20	REar	0,438	0,083	255,0,255	18	17
21	LEar	0,663	0,083	85,0,255	19	18
22	LBigToe	0,753	0,954	0,0,255	14	19
23	LSmallToe	0,809	0,944	0,0,255	22	20
24	LHeel	0,663	0,907	0,0,255	14	21
25	RBigToe	0,27	0,954	0,255,255	11	22
26	RSmallToe	0,213	0,944	0,255,255	25	23
27	RHeel	0,36	0,907	0,255,255	11	24

Fonte: elaborado pelo autor

Finalmente, outro exemplo de aplicação deste modelo configurável já foi visto neste trabalho anteriormente na Figura 24 presente no Capítulo 7, em que o modelo de pose do personagem de exemplo foi construído através de uma tabela de pose personalizada para ficar mais simples, sem a informação dos olhos ou dos pés do personagem. Com tudo isso, cada uma das 200 imagens do *sprite sheet* completo foram mapeadas conforme o modelo apresentado anteriormente e ajustadas manualmente, sendo possível prosseguir para a próxima etapa que é a preparação do processamento da rede neural.

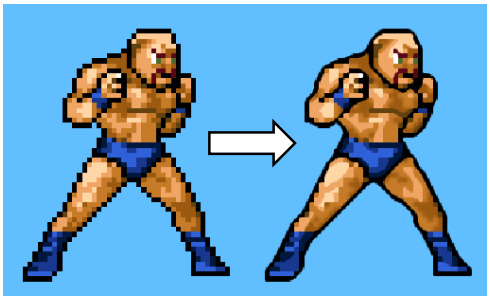
8.1.3 Entrada da rede neural

Conforme visto anteriormente, o treinamento utilizado a ferramenta Pix2Pix é feito com pares de imagens. Portanto, o mapeamento entre o modelo de pose e cada imagem que foi feito na etapa anterior é usado neste momento para criar novos arquivos para o treinamento da RNA. Isso é feito varrendo cada pose mapeada e extraindo a região da imagem que se encontra sobre ela para uma novo arquivo separado que vincula essas duas informações.

Quanto à resolução da imagem, foi aplicado o cálculo $256/L$ para descobrir a proporção entre a largura (L) da imagem e os 256 pixels utilizados pelo algoritmo interno do Pix2Pix. Se esta proporção for maior que 2 vezes, é aplicada a técnica de mudança de escala vista na seção 6.2 para suavizar a imagem, para só então ampliar este resultado até atingir os 256 pixels de largura desejados como pode ser visto na Figura 27. Desta forma, são geradas imagens com exatamente 256 pixels de largura e o processamento interno do Pix2Pix precisará ajustar somente a altura da imagem. Acredita-se que com isso será

gerada menos distorção das informações durante o treinamento.

Figura 27: Comparativo da ampliação direta sem suavização e com aplicação do filtro hq4x



Fonte: elaborado pelo autor

Tendo definido o método para montar as imagens de treino, é necessário ainda separar o conjunto de 200 *sprites* entre 150 imagens de treino e 50 para teste. Para fazer isso, é interessante que os dois conjuntos tenham uma distribuição de características semelhante e, desta forma, é necessário um critério para diferenciar as imagens e permitir essa classificação. Foi decidido separar os *sprites* de acordo com a dificuldade para desenho dos mesmos, indicando quais são fáceis, médios ou difíceis. Assim será possível criar os dois conjuntos com proporções iguais de imagens de cada tipo, provendo um contexto semelhante para o treino e para o teste. Os critérios indicados para essa análise são aqueles definidos nas técnicas de avaliação qualitativa de imagens da seção 3.3.

Para evitar conflitos de interesse neste momento da classificação dos *sprites*, selecionando, por exemplo, as imagens de difícil geração para treino e as fáceis para teste, foi solicitado auxílio externo, buscando encontrar um docente da Universidade Feevale que fosse familiarizado com técnicas de desenho e análise de imagens. Com isso, foi convidado para participar desta etapa o professor Júlio Cesar Da Rosa Herbstrith, que é mestre em História, Teoria e Crítica da Arte pelo Programa de Pós-Graduação em Artes Visuais do Instituto de Artes da UFRGS (2012). Além disso, formou-se bacharel em Artes Visuais com habilitação em Desenho, pelo Instituto de Artes da UFRGS (2007). Ele integra o quadro docente da Universidade Feevale desde 2013, nos cursos de graduação e pós-graduação, onde atualmente coordena o curso de Design de Interiores. Ele possui seis anos e meio de experiência docente nas áreas de desenho, expressão gráfica e história, teoria e crítica da arte. Finalmente, tem experiência profissional em animação, colaborando com o longa metragem “As Aventuras do Avião Vermelho” de 2014.

Com tudo isso, acredita-se que ele pode contribuir neste momento de classificação das imagens, empregando critérios técnicos alinhados com as necessidades deste projeto e provendo uma avaliação imparcial. A partir do referencial que foi estudado na seção 3.3 e da experiência que possui, ele propôs o método de aplicação destes conceitos do desenho de observação. No contexto da geração de imagens por meios computacionais, o professor

sintetizou três elementos que pautam a análise a ser feita sobre as imagens, conforme segue:

1. **Proporção:** Relação entre as partes e o todo da forma representada. Para ser qualificado como bom o desenho deve manter uma harmonia visual entre o referencial e a representação.

2. **Escorço:** Segundo Melo (2011), "O escorço é, portanto, a representação de uma figura volumétrica vista numa perspectiva forçada onde as partes do corpo se encontram em sobreposição e as suas superfícies se apresentam aparentemente distorcidas". Sendo o escorço uma das técnicas de desenho mais complexas para a representação gráfica, mesmo para o aprendizado do ser humano, no que tange à RNA este critério é fundamental para que a imagem gerada possa ser avaliada quanto à qualidade formal, harmonia entre as partes, relações de espaçamento e profundidade entre as partes do desenho. No que se refere ao modelo, como figura humana, o bom desenvolvimento do escorço auxilia na compreensão da anatomia da figura.

3. **Volumetria:** Esse aspecto pode ser desenvolvido com a linha, no que se refere ao uso de diagonais que denotam profundidade, mas também com a aplicação de luz e sombra que pode ser determinada por uma variação tonal. Quando desenvolvida de forma adequada, a representação passará ao observador a sensação de tridimensionalidade. Do contrário, o desenho apresentará problemas formais que podem confundir a percepção do observador, levando-o a não relacionar a representação com a referência.

O professor então aplicou suas técnicas próprias e estes critérios para classificar as imagens do *sprite sheet* por dificuldade de desenho. Segundo ele, o referencial de execução fácil é aquele que permite maior clareza entre as partes e o todo e se encontra em situação de repouso aparente. Além disso, onde os eixos de orientação do corpo e suas partes estão definidos de forma clara e apresentam a totalidade da extensão do corpo. O referencial de dificuldade média é aquele que possui clareza entre as partes e o todo, porém está em situação de movimento e pode gerar eixos de direcionamento com representação mais complexa. Estar em posição frontal ou de três-quartos também pode deixar o referencial mais complexo com dificuldade média. Por fim, a referência de difícil execução não possui clareza total entre as partes e o todo é complexificada por um movimento aparente e por uma situação de escorço.

O resultado dessa classificação pode ser visto no Apêndice A, onde se encontram identificados 66 *sprites* fáceis pela cor verde, 82 médios na cor amarela e 52 difíceis na cor vermelha. Tendo essa separação é possível identificar as imagens que serão usadas para o teste. As 50 imagens que deseja-se selecionar para essa tarefa correspondem a

1/4 do total de imagens, portanto pode ser aplicada esta proporção em cada um destes conjuntos. Conclui-se disso que deve haver no mínimo 16 imagens fáceis, 20 médias e 13 difíceis no conjunto de treino, e mais 1 qualquer para totalizar a quantidade de 50 que foi proposta. Para selecionar quais são as imagens de treino dentro de cada conjunto, foi escrito um algoritmo, utilizando a linguagem de programação Java, que gera uma lista de 200 números sequenciais e então reordena essa lista aleatoriamente, o qual pode ser visto na Figura 28.

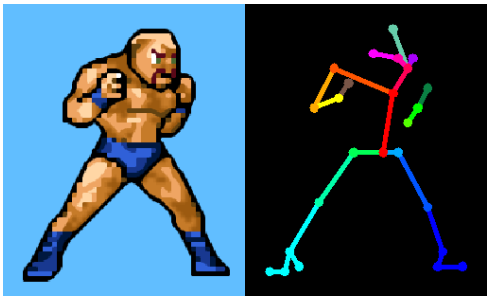
Figura 28: Algoritmo para escolha dos *sprites* de teste

```
1       ArrayList<Integer> lista = new ArrayList<>();
2       for(int i = 1;i<=200;i++){
3           lista.add(i);
4       }
5       Collections.shuffle(lista, new Random(100));
6       System.out.println("\nLista reorganizada: \n" + lista);
```

Fonte: elaborado pelo autor

Essa lista reordenada foi lida a partir do começo, buscando a imagem indicada pelo número que ficou em cada posição. Selecionou-se imagens até preencher a quantidade mínima de cada categoria - permitindo até 1 a mais para totalizar as 50 imagens. Deste modo, foi obtido o conjunto de *sprites* para teste e o restante ficou reservado para o treinamento. Essa lista de imagens selecionadas para teste também se encontra discriminada no Apêndice A, estando destacada pelas imagens hachuradas com a borda maior em negrito. As imagens cujo número identificador não possui essa borda destacada são o conjunto de treino. Com tudo isso, foram construídas as imagens de entrada conforme o exemplo da Figura 29, onde a imagem-alvo se encontra na metade esquerda, enquanto que a representação utilizando o modelo simplificado está à direita. Assim sendo, é possível utilizar as 150 imagens de treino como entrada para geração da rede neural, que é vista a seguir.

Figura 29: Exemplo de uma imagem de entrada utilizada para geração da RNA



Fonte: elaborado pelo autor

8.1.4 Geração da rede neural

Tendo gerado as imagens de entrada, pode ser efetivamente construída a rede neural, no caso deste projeto utilizando como ferramenta o Pix2Pix. Conforme abordado anteriormente na seção 7.4, é necessário primeiramente configurar o ambiente, instalando as dependências para execução do treinamento no sistema operacional Windows. O primeiro requisito a ser instalado é a plataforma de computação paralela CUDA, que é desenvolvida pela empresa NVIDIA e permite a execução de algoritmos utilizando o poder de processamento da placa gráfica do computador, a chamada GPU. É recomendável instalar também a extensão cuDNN (do inglês *CUDA Deep Neural Network*), que habilita recursos e otimizações específicos para operações com RNA, os quais serão úteis para agilizar o processo de treinamento. Com isso, o sistema operacional fica preparado para as operações gráficas e de RNA utilizadas no processo.

Pode então ser instalado o ambiente de programação que permite a execução do algoritmo do Pix2Pix, que é escrito em Python e, portanto, exige o interpretador dessa linguagem de programação. Após instalá-lo, fica disponível no *prompt* de comandos do sistema operacional o comando `pip` (*package installer for Python*). O `pip` é um gerenciador de pacotes que busca e instala bibliotecas de código-fonte desta linguagem, e será utilizado para instalar a última dependência. Esse último recurso externo é o Tensorflow, uma biblioteca de operações de aprendizado de máquina que faz uso dos recursos computacionais da plataforma CUDA e é executado em Python. Por fim, o que vai efetivamente invocar e gerenciar todo processo de treinamento é o algoritmo do Pix2Pix, que pode ser obtido através do repositório *online* GitHub conforme explicado na publicação original desta ferramenta (ISOLA *et al.*, 2017).

Concluída essa configuração do ambiente, resta a aplicação dos parâmetros disponíveis para treinamento que também foram estudados anteriormente na seção 7.4. Como o Pix2Pix é executado através de linha de comando, os parâmetros são informados através de palavras específicas após a chamada principal. O formato geral deste comando está descrito abaixo:

```
py -3.6 pix2pix.py --mode [train|test] --input_dir AAAA --output_dir
BBBB --which_direction [BtoA|AtoB] --max_epochs 200 --checkpoint CCCC
```

Onde se destaca:

1. **`py -3.6 pix2pix.py`**: É a chamada principal do algoritmo, indicando em qual versão da linguagem Python ele será executado.

2. **`--mode`**: Este parâmetro indica o modo de execução. A palavra `train` indica uso para construir a RNA, gerando como saída um modelo de rede neural artificial.

Enquanto isso, a palavra `test` faz a aplicação de uma RNA que tenha sido gerada anteriormente, gerando como saída imagens correspondentes às entradas fornecidas.

3. **`--input_dir`**: Indica o diretório `AAAA` do qual devem ser lidos os pares de imagens para treino, ou para aplicação da RNA no modo de teste.

4. **`--output_dir`**: Indica o diretório `BBBB` no qual deve ser salvo o resultado do treino ou o resultado da geração de imagens a partir da RNA.

5. **`--which_direction`**: Específico da função de treino, indica se o objetivo do treino é aprender a converter a imagem da direita para esquerda (`AtoB`), ou da esquerda para direita (`BtoA`).

6. **`--max_epochs`**: Outro parâmetro específico do modo de treino que indica a quantidade de épocas a serem executadas.

7. **`--checkpoint`**: Este parâmetro opcional do modo treino indica um diretório `CCCC` que contenha uma RNA gerada anteriormente para servir como ponto de partida para. Isso permite fazer o treino gradativamente, além de possibilitar conferir resultados parciais. No modo de teste este parâmetro é obrigatório para indicar onde está a RNA a ser aplicada na geração de imagens.

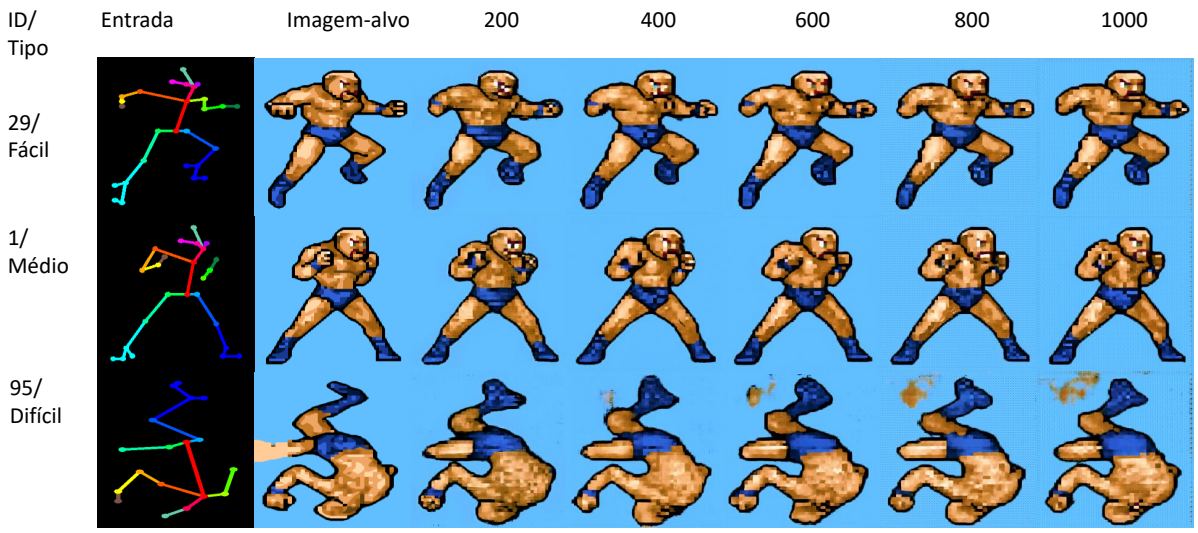
Com tudo isso, foi feito o processo de treinamento utilizando esta linha de comando e gerando resultados parciais a cada 200 épocas de treino, até atingir a quantidade de 1000 épocas. Com o treino efetuado, pode ser aplicada a RNA resultante para criação de novas imagens.

8.1.5 Aplicação da rede neural

A penúltima etapa retoma o modelo de mapeamento da pose, desta vez para construir novas imagens. Como já haviam sido mapeadas as 200 imagens anteriormente, foram selecionadas as 50 imagens que não participaram do treinamento para serem geradas a partir da RNA neste momento. Não foram identificados parâmetros novos na ferramenta Pix2Pix que sejam específicos para aplicação nessa etapa: basta utilizar o parâmetro `--mode` indicando modo `test` para aplicar a rede neural gerada anteriormente.

Desta forma, utilizou-se o resultado de cada treinamento efetuado anteriormente, gerando um conjunto de 50 imagens de teste para as redes neurais com 200, 400, 800, 600 e 1000 épocas de treino. Na Figura 30 encontram-se alguns exemplos das imagens fornecidas como entrada para a RNA criada e as respectivas saídas geradas, sem nenhum ajuste efetuado ainda. Ou seja, tanto a imagem-alvo quanto a saída gerada ainda não são classificadas como *pixel art*. A seguir será visto como é o processo de normalização dessas saídas.

Figura 30: Exemplo de entrada e saídas geradas pela RNA treinada a partir do *sprite sheet*



Fonte: elaborado pelo autor

8.1.6 Normalização para *pixel art*

A etapa final do processo é a normalização da saída da RNA de forma que assuma novamente as características do *pixel art* (seção 7.6). Foram propostas duas transformações na imagem de saída da RNA para cumprir este objetivo: ajuste da paleta de cores e mudança da escala da imagem.

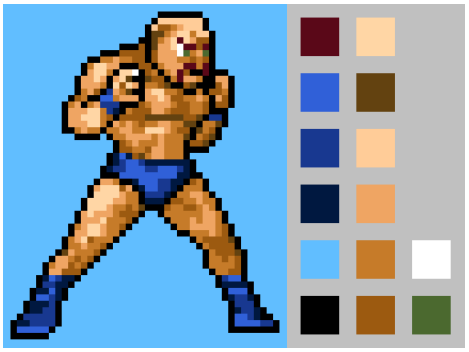
O primeiro tratamento altera as cores da imagem para ficar de acordo com a paleta presente no *sprite sheet* original do personagem. Nota-se que o processo de geração da RNA não limita a quantidade de cores, pois vai compondo a saída através de convoluções que manipulam os *pixels* até formar a imagem de saída. A Figura 31 demonstra as 14 cores que existem no *sprite sheet* utilizado neste projeto, as quais devem, portanto, ser as únicas existentes no *sprite* final. Para trabalhar este aspecto, utilizou-se novamente a programação Java, que possui na classe *BufferedImage* vários recursos para manipular imagens. Essa classe permite escolher o tipo de arquitetura da imagem, ou seja, de qual forma as informações das cores e formas são armazenadas em memória. Uma das tipagens disponíveis é o *TYPE_INT_RGB*, que grava cada pixel da imagem no formato RGB. Essa simplicidade o torna bastante prático para manipulações gerais, sendo este o modelo aplicado até agora nas implementações deste projeto. Porém, para essa tarefa de normalização convém utilizar o tipo *TYPE_BYTE_INDEXED*, que registra cada cor distinta da imagem apenas uma vez, gerando um índice de cores disponíveis, ou seja, exatamente o mesmo conceito de uma paleta de cores usada em *pixel art* que foi visto na seção 2.2. Ao invés de descrever a cor de cada pixel, este modelo indica qual o índice da paleta a ser aplicado. Desta forma, para normalizar as cores bastam três comandos principais:

1. Criar um objeto da classe *IndexColorModel* atribuindo a ele cada cor existente na paleta do personagem.

2. Utilizar este objeto para construir um *BufferedImage* do tipo *TYPE_BYTE_INDEXED*.

3. Desenhar sobre este objeto de imagem o *sprite* a ser normalizado, que neste caso é a saída da RNA. Os mecanismos internos dessa construção vão buscar na paleta a cor mais adequada para cada pixel que for inserido na imagem.

Figura 31: Paleta do personagem proposto neste projeto. A cor cinza serve apenas como plano de fundo para destacar as demais cores.

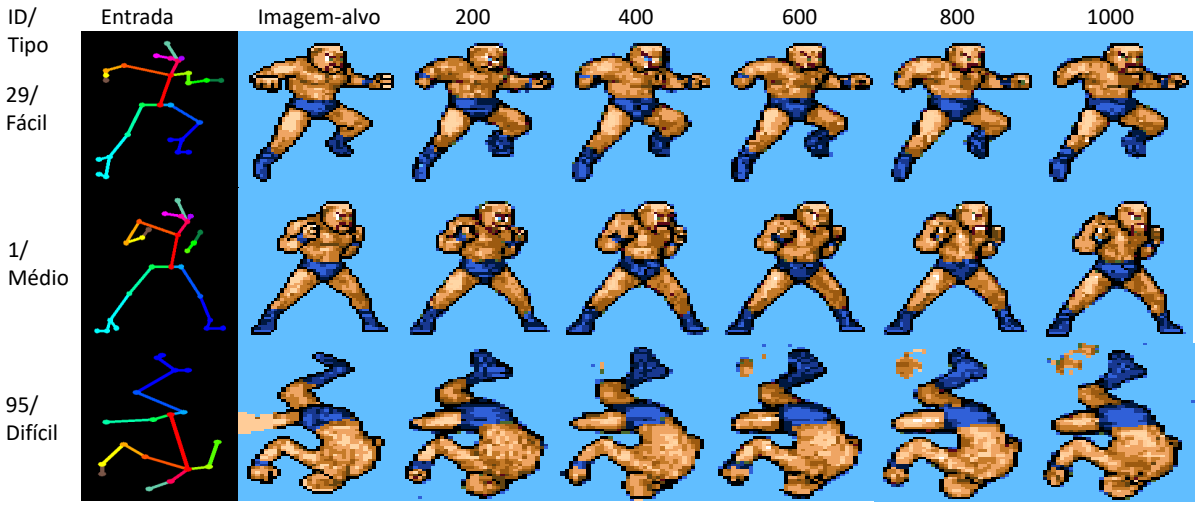


Fonte: elaborado pelo autor

Com a normalização das cores resolvida, resta adequar as dimensões da imagem, que neste momento ainda se encontra com os 256 pixels de largura e altura que são gerados pela saída da RNA. O tamanho desejado do sprite a ser gerado é o tamanho em que foi desenhado o respectivo modelo de pose que serviu de entrada. Dessa forma, o ajuste a ser aplicado pode ser obtido efetuando o cálculo inverso ao que foi visto na subseção 8.1.3. Calcula-se portanto $L/256$ e $A/256$ para descobrir a relação entre a largura L e a altura A desejados para o *sprite* e os 256 pixels de saída da RNA.

A programação efetuada para mudar a resolução da imagem também aplica a classe Java *BufferedImage*. Obtém-se neste objeto a instância da classe *Graphics2D*, que permite configurar o método de desenho de tudo que for inserido na imagem. É possível então manipular a escala do desenho através do método *scale(double x, double y)*, bastando informar neste método a relação de altura e largura calculados anteriormente. Assim, ao desenhar o resultado da RNA sobre este novo objeto, será aplicada essa escala e a imagem gerada terá exatamente o mesmo tamanho do modelo original que serviu de base para a entrada para rede neural. Os mecanismos internos da classe *Graphics2D* definem qual o pixel mais adequado para ser desenhado em cada ponto da imagem através do método *nearest neighbor* que foi citado na seção 6.2. Com estes dois ajustes efetuados, a Figura 32 demonstra alguns *sprites* gerados ao final do processo. Uma planilha de todos os resultados obtidos neste experimento pode ser observada no Apêndice B.

Figura 32: Exemplos de novos *sprites* normalizados como *pixel art*



Fonte: elaborado pelo autor

8.2 EXPERIMENTO 2 - COLORAÇÃO DE UM *SPRITE*

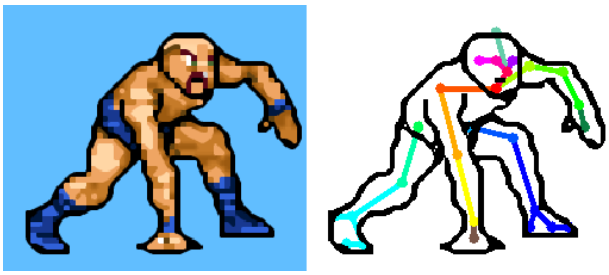
O segundo experimento proposto visa executar o processo em um contexto diferente, evidenciando que é possível flexibilizar para outras aplicações e demonstrando uma nova lógica de mapeamento das informações. Também espera-se tornar mais claro os pontos que podem ser alterados e o impacto da mudança do modelo de mapeamento sobre os resultados gerados. É proposto mapear neste experimento não somente a pose mas também os contornos do personagem. Desta forma a rede neural já receberá esta informação e o aprendizado se concentrará na distribuição das cores com base nestes contornos e na pose. O processo executado é praticamente o mesmo, sendo necessário alterar somente a etapa de geração da entrada para treinamento da rede neural, que é detalhada a seguir.

8.2.1 Adequação da entrada da rede neural

Para atender a proposta deste modelo de mapeamento da pose, identificou-se que os contornos deste personagem estão representados sempre pelo tom mais escuro do *sprite*. A partir disso, foi escrito um algoritmo para registrar essa informação adicional nas imagens de treino. Buscou-se, através dos canais RGB da imagem, os pixels que possuíssem o valor menor que 15 em cada um dos três canais, desta forma, extraindo somente as cores com tonalidade bastante escura, muito próximas ao preto absoluto. Tendo identificado essas regiões da imagem, é possível combiná-las com a pose do personagem. Como a cor de fundo utilizada até então no desenho da pose já era a cor preta, foi necessário substitui-la neste modelo pela cor branca, para que fosse possível diferenciar dos contornos do personagem. Sob esta imagem da pose com o fundo branco, foi feito o desenho dos contornos aplicando o mesmo algoritmo de aumento da escala, de forma que os fossem

desenhadas linhas suaves sem pixels evidentes. A Figura 33 exemplifica como fica este modelo de mapeamento aplicado a um *sprite*.

Figura 33: Exemplo de um *sprite* mapeado conforme o modelo do segundo experimento

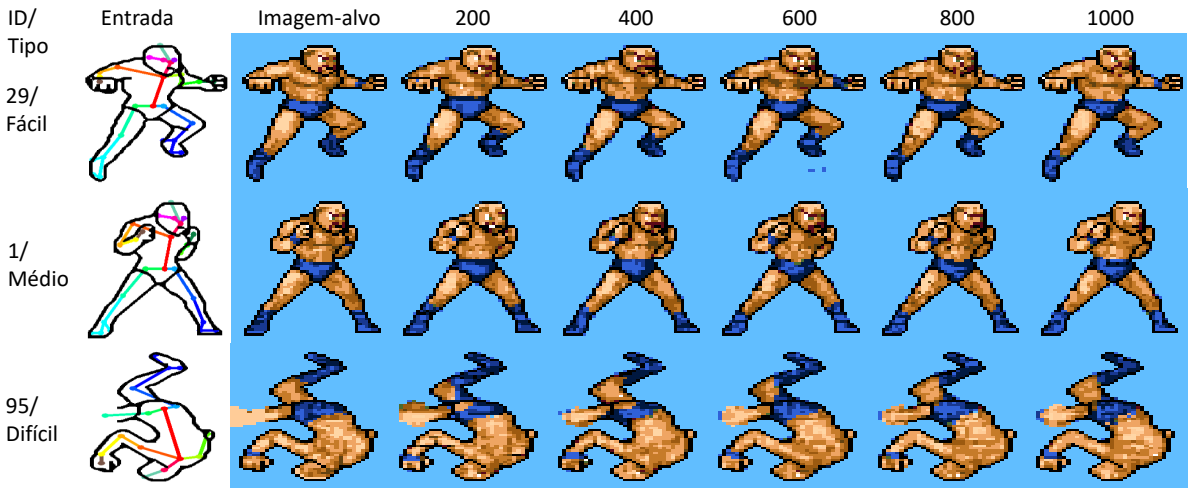


Fonte: elaborado pelo autor

8.2.2 Aplicação da rede neural e normalização

A partir dessa modalidade de mapeamento das imagens, foi gerado um novo conjunto de imagens de treino para a RNA. Seguiu-se exatamente a mesma proposta vista anteriormente no que diz respeito à separação dos grupos de imagens entre treino e teste, quantidade de épocas de treino e metodologia de normalização após aplicação da RNA gerada. Ou seja, a única diferença é o modelo de pose utilizado. Com isso, a Figura 34 demonstra exemplos desta entrada, as respectivas imagens de referência que seriam esperadas como resultado ideal, além dos *sprites* efetivamente obtidos ao final do processo. Uma planilha contendo todas as imagens resultantes deste segundo experimento pode ser observada no Apêndice C.

Figura 34: Exemplos de *sprites* do segundo experimento normalizados como *pixel art*



Fonte: elaborado pelo autor

9 RESULTADOS E AVALIAÇÃO

Feito o processo completo de geração de novos *sprites*, este capítulo propõe verificar a efetividade dessa proposta, aplicando as técnicas de avaliação de imagens do Capítulo 3 e explorando outros conceitos úteis para interpretação de dados. Inicialmente será vista a metodologia de avaliação abordando conceitos de análise estatística, para então apresentar o processo de obtenção dos dados. Por fim, os resultados específicos são demonstrados e comentados nas demais seções.

9.1 METODOLOGIA DE AVALIAÇÃO

Enquanto que na subseção 8.1.3 foi utilizado o método qualitativo para separar as imagens entre fáceis, médias e difíceis, nesta seção é retomada esta classificação e são aplicadas também as técnicas quantitativas para avaliação. O primeiro passo para efetuar essa investigação dos resultados é definir o método a ser seguido. É proposto então obter informações (ou escores) que qualifiquem as imagens geradas e com isso utilizar conceitos de análise estatística para interpretar esses dados, evidenciando as características do processo.

A primeira das técnicas a ser aplicada no estudo dos dados é o teste de Kruskal-Wallis, que permite identificar se determinado conjunto de amostras exerce dominância estocástica sobre outro conjunto. Ou seja, ele examina se os resultados apresentados no conjunto testado são probabilisticamente superiores a outros resultados (HOLLANDER; WOLFE, 1999). Este método pode ser aplicado para verificar, por exemplo, a influência da categoria da imagem, identificando variações nos escores entre as imagens fáceis, médias e difíceis.

Além disso, é possível a análise intragrupo e intergrupo de outras formas. Nesse sentido, pode ser considerando como agrupamento dos resultados a quantidade de épocas de treino (que variou entre 200 e 1000 repetições) e o tipo de avaliação efetuada (imagem binarizada e paleta de cores). Assim, o método *Generalized Equations Estimating* (GEE) permite contabilizar a correlação existente entre medidas repetidas dos mesmos indivíduos, ou seja, quando já há uma noção de progressão dos dados (LIANG; ZEGER, 1986). O método GEE também é conhecido como Modelos Marginais e pode ser considerado uma extensão de Modelos Lineares Generalizados (NELDER; WEDDERBURN, 1989). O fato de examinar a correlação entre medidas da mesma unidade amostral permite aplicar esta técnica sobre as imagens geradas, pois foram todas criadas a partir de um mesmo conjunto de origem em todos os experimentos. Nesse contexto, a análise intergrupo consiste em dimensionar a homogeneidade dos experimentos fixando as iterações (épocas de

treino) e variando o método de avaliação, ou seja, evidenciando se os resultados foram significativamente melhores ou piores para a avaliação binária ou para o teste da paleta. Enquanto isso, a análise intragrupo consiste em confrontar o número de iterações em cada um dos experimentos e fixar a forma de avaliação, desta forma verificando dentro da avaliação binária e dentro da avaliação da paleta se houve variações relevantes conforme mudou-se a quantidade de treino efetuado.

O resultado destes procedimentos de análise é demonstrado através do valor-p, que é uma estatística que sintetiza a probabilidade de um teste de hipótese ser verdadeiro. Neste estudo está sendo verificada a hipótese nula, que é a situação na qual a variação nos escores comparados não é significativa o suficiente para concluir que existe relação de dependência entre eles. Normalmente considera-se o nível de significância em 5%, desta forma um valor-p menor do que 0,05 gera evidências para rejeição da hipótese nula e permite assumir que os resultados foram de fato influídos pela variável alterada entre os agrupamentos.

Tendo estabelecidos os conceitos sobre a metodologia de análise dos dados, pode ser apresentado o processo de obtenção das informações a serem estudadas. Aplicou-se a técnica de comparação binária (seção 3.1) e de avaliação da paleta (seção 3.2) para todos os cenários gerados no experimento 1 (seção 8.1) e no experimento 2 (seção 8.2). Como houve duas formas de avaliação e variou-se entre cinco quantidades de épocas (200, 400, 600, 800 e 100), há então vinte cenários de teste distintos, que podem ser considerados como os agrupamentos de informações a serem estudados. Além disso, cada um destes cenários executou o processo de geração para cinquenta imagens, gerando um total de mil avaliações que verificam a diferença entre a imagem ideal esperada e o que foi gerado pelo processo. Conforme apresentado ao discutir essas técnicas de avaliação no Capítulo 3, essa diferença foi sempre representada por um número que varia entre 0 e 1. Estes serão os dados a serem analisados para verificar se houve algum padrão significativo, sendo possível detalhar a seguir três investigações que foram feitas sob esses dados dos experimentos.

9.2 RESULTADOS PELA MODALIDADE DE AVALIAÇÃO

A primeira comparação dos resultados foi feita variando a forma de avaliação e fixando a quantidade de épocas de treino, desta forma verificando se houve variação significativa dos resultados dependendo do critério de avaliação empregado. Desta forma, a Tabela 1 demonstra esse agrupamento de resultados destacando a média, desvio padrão e o respectivo valor-p associado a cada cenário comparado. Observa-se que para ambos os experimentos, em todas as quantidades de iterações de treino, houve variação significativa do resultado, denotada pelo valor-p abaixo de 0,05. As médias dos valores foram maiores na avaliação pela paleta de cores, indicando que o desempenho na avaliação binária da imagem foi o melhor em todos os casos por identificar menos diferenças entre as imagens.

Tabela 1: Comparação dos escores por modalidade de avaliação

Experimento	Iterações	Avaliação	Média	D.P.	Valor-p
Experimento1	200	Binária	0,0340	0,0205	**<0,001**
		Paleta	0,3660	0,0543	
	400	Binária	0,0205	0,0192	**<0,001**
		Paleta	0,3430	0,0664	
	600	Binária	0,0284	0,0210	**<0,001**
		Paleta	0,3481	0,0613	
	800	Binária	0,0229	0,0198	**<0,001**
		Paleta	0,3448	0,0647	
	1000	Binária	0,0336	0,0188	**<0,001**
		Paleta	0,3392	0,0547	
Experimento 2	200	Binária	0,0133	0,0146	**<0,001**
		Paleta	0,3026	0,0611	
	400	Binária	0,0164	0,0142	**<0,001**
		Paleta	0,2984	0,0546	
	600	Binária	0,0182	0,0146	**<0,001**
		Paleta	0,3044	0,0538	
	800	Binária	0,0153	0,0139	**<0,001**
		Paleta	0,2754	0,0549	
	1000	Binária	0,0131	0,0156	**<0,001**
		Paleta	0,2728	0,0708	

Fonte: elaborado pelo autor

Essa diferença nos escores obtidos entre as duas modalidades de avaliação fica evidente também pela distância numérica dos resultados: enquanto que a avaliação da imagem binária identificou no pior caso 3,4% de diferença, a modalidade de comparação da paleta atingiu 36,6% nas mesmas condições. Porém, essa situação comprovada pela análise estatística já era esperada pelo fato de que a avaliação da paleta é mais criteriosa, uma vez que trata cada cor como uma comparação separada. Enquanto isso, a verificação pela imagem binária tem uma visão mais global da imagem e identifica somente variações de forma ou de luminosidade das informações representadas.

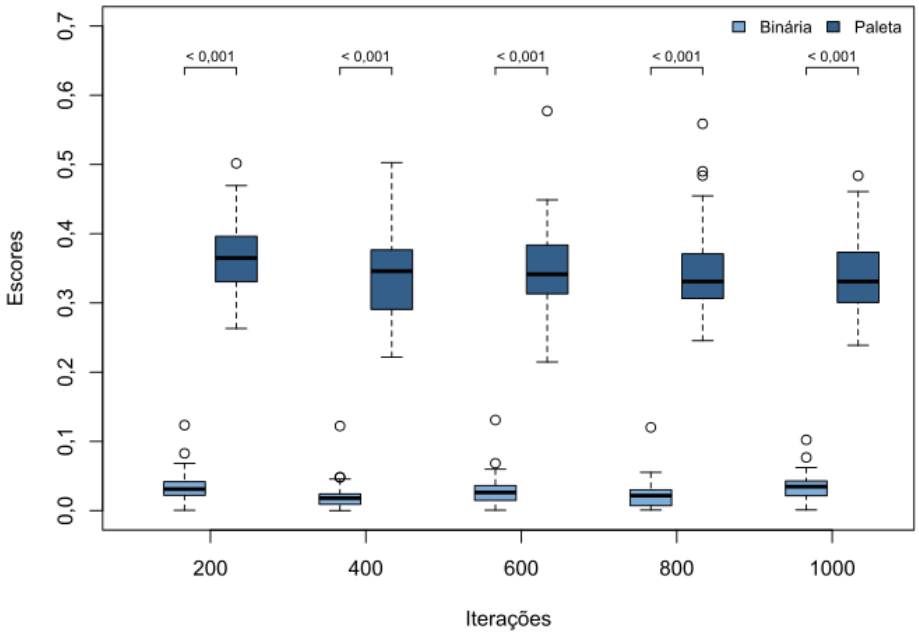
Buscando entender os motivos que contribuíram para esta diferença na grandeza dos resultados, nota-se que a avaliação de paleta que foi definida na seção 3.2 é composta pela média simples do resultado obtido por cada cor distinta da paleta do personagem. Ou seja, cada cor contribui igualmente para compor esse escore, embora cada cor não contribua da mesma forma para formar a totalidade da imagem. Isso pode ser melhor compreendido retomando a Figura 31 que foi utilizada anteriormente para apresentar a paleta de cores do personagem utilizado nos experimentos. Neste exemplo específico, existe apenas 2 pixels da imagem inteira que são da cor verde, os quais representam os olhos do personagem. Além disso, apenas 3 pixels na cor branca, que completa os olhos e pode aparecer também onde há um brilho muito intenso a ser representado. Essas duas cores aparecem pouco em nos sprites e, embora sejam menos perceptíveis na imagem, participam com o mesmo peso na avaliação da paleta. Além disso, a quantidade reduzida destas cores também facilita a ocorrência de escores mais altos, pois, conforme foi visto no Capítulo 3, o erro é normalizado independente da grandeza comparada. Por exemplo,

se deveria haver 2 pixels na cor verde e foi gerado somente 1, contabiliza-se um erro de 50% nesta cor. Uma forma de melhorar esta técnica de avaliação seria substituir a média simples por uma média ponderada, desta forma cada cor poderia contabilizar o escore de diferença proporcionalmente a sua participação na imagem, o que ajudaria a diminuir o peso de erros menores, enquanto aumentaria o rigor para os tons mais notáveis.

Por fim, outro fator que pode ter contribuído para elevar as médias na avaliação pela paleta é a quantidade de tons de pele. Havendo 6 tons da cor marrom aumenta-se a chance de a RNA gerar uma cor que será classificada de forma incorreta ao aplicar a paleta na etapa de normalização (vista na subseção 8.1.6). Esse fator, no entanto, é uma dificuldade do personagem escolhido para o experimento e realmente pode ocorrer conforme são utilizadas mais tons semelhantes na paleta. Com tudo isso, entende-se que a avaliação binária também está sujeita a estes comportamentos, porém, numa escala bem menor, obtendo resultados melhores.

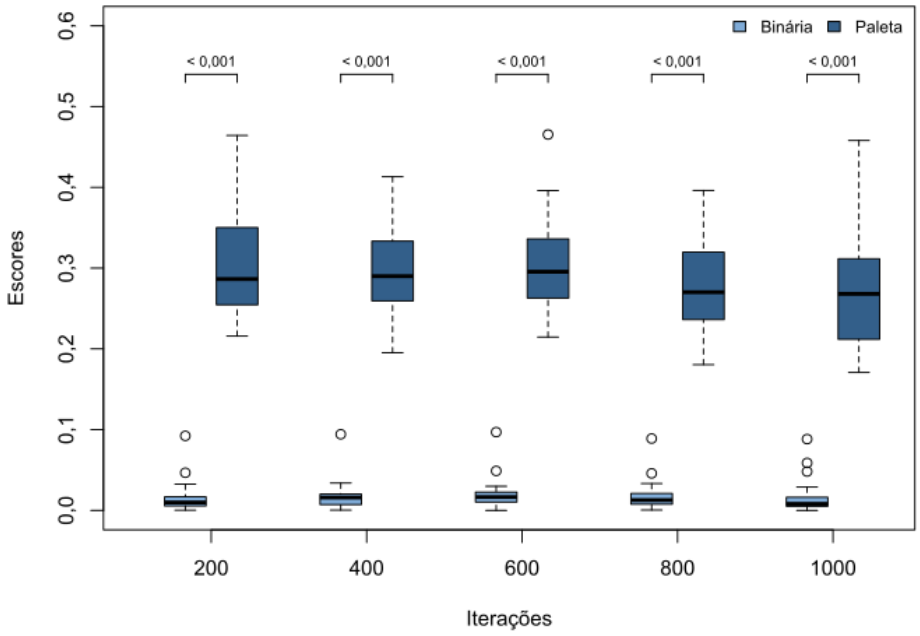
Dito isso, a Figura 35 apresenta a visão de resultados do experimento 1 através de um diagrama de caixa. Nesta imagem pode ser observada a distribuição dos quartis, valores aberrantes (do inglês *outliers*) e as medianas, bem como as relações de dependência que legitimam o desempenho superior da análise pela imagem binária. Da mesma forma, a Figura 36 traz essas informações para o experimento 2.

Figura 35: Resultados do experimento 1 pela modalidade de avaliação



Fonte: elaborado pelo autor

Figura 36: Resultados do experimento 2 pela modalidade de avaliação



Fonte: elaborado pelo autor

9.3 RESULTADOS PELA QUANTIDADE DE ITERAÇÕES DE TREINO

A Tabela 2 apresenta outra análise, que utiliza como agrupamento o experimento e o método de avaliação, efetuando o comparativo de resultados entre as diferentes épocas de treino utilizadas, além de demonstrar a comparação entre os dois experimentos. Analisando inicialmente a comparação geral, na coluna "E1 x E2", tanto na avaliação binária quanto na avaliação de paleta houve relação significativa (valor-p menor que 0,05) para todas as épocas de treino. As menores médias são do experimento 2, indicando que este obteve melhor desempenho de acordo com a métrica de comparação das imagens.

Tabela 2: Comparação dos escores em relação a 200 épocas de treino

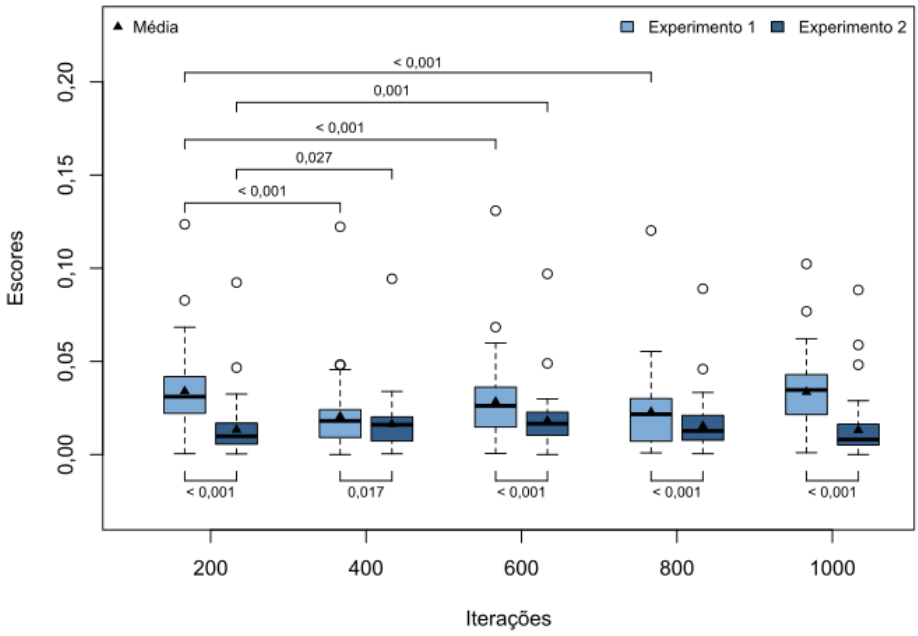
Avaliação	**Iterações**	**Experimento 1 (E1)**			**Experimento 2 (E2)**			**E1 x E2**
		Média	**D.P.**	**Valor-p**	**Média**	**D.P.**	**Valor-p**	
Binária	200	0,0340	0,0205	-	0,0133	0,0146	-	**<0,001**
	400	0,0205	0,0192	**<0,001**	0,0164	0,0142	**0,027**	**0,017**
	600	0,0284	0,0210	**<0,001**	0,0182	0,0146	**0,001**	**<0,001**
	800	0,0229	0,0198	**<0,001**	0,0153	0,0139	0,090	**<0,001**
	1000	0,0336	0,0188	0,812	0,0131	0,0156	0,869	**<0,001**
Paleta	200	0,3660	0,0543	-	0,3030	0,0611	-	**<0,001**
	400	0,3430	0,0664	**<0,001**	0,2980	0,0546	0,450	**<0,001**
	600	0,3480	0,0613	**0,002**	0,3040	0,0538	0,769	**<0,001**
	800	0,3450	0,0647	**0,001**	0,2750	0,0549	**<0,001**	**<0,001**
	1000	0,3390	0,0547	**<0,001**	0,2730	0,0708	**<0,001**	**<0,001**

Fonte: elaborado pelo autor

A diferença numérica entre os dois experimentos sugere uma melhoria mais expressiva na avaliação pela paleta. Utilizando a quantidade de 200 épocas como exemplo, a avaliação binária reduziu as diferenças em 2,07% no segundo experimento, e o outro método apresentou redução de 6,3% neste caso. Este resultado pode ser devido ao fato de que o experimento 2 já trazia mais informações sobre a imagem desejada na entrada da RNA. Como os contornos do personagem já estavam definidos, o processo de geração concentrou-se em preencher essas áreas, desta forma ocorrendo menos deformações em relação à imagem esperada. Quanto à melhoria maior na avaliação pela paleta, nota-se que os contornos também auxiliaram a não preencher cores onde não deveria haver, desta forma, estando menos propenso a gerar saídas com diferenças na coloração.

Quanto a variação nas iterações de treino, pode ser observado o valor-p apresentado entre os dois experimentos, que nesta tabela é calculado utilizando como referência a quantidade de 200 épocas de treino. Verifica-se no experimento 1 diferenças significativas na avaliação binária entre as iterações 400, 600 e 800, sendo que a quantidade de 200 gerou a pior média, ou seja, houve melhoria ao aumentar o treino. Enquanto isso, no experimento 2 a avaliação binária gerou variação relevante apenas entre as iterações 400 e 600, onde o melhor resultado apareceu neste caso na quantidade de 200. O mesmo patamar na métrica de avaliação adotada foi atingido somente ao efetuar a quantidade de épocas de 800 e 1000, ou seja, houve uma queda seguida de um aumento no desempenho ao incrementar a quantidade de épocas. A Figura 37 demonstra estes resultados da avaliação binária através do diagrama de caixa.

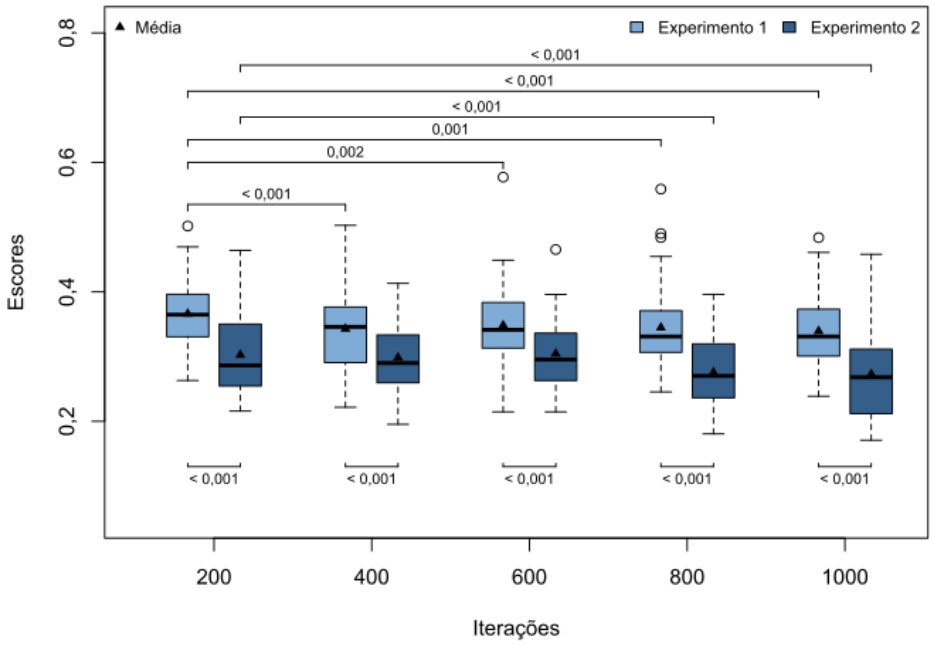
Figura 37: Comparativo dos resultados da avaliação de imagem binária



Fonte: elaborado pelo autor

Enquanto isso, a investigação da avaliação pela paleta de cores revelou diferenças significativas no experimento 1 para todas as quantidades de iterações em relação ao treino com 200 épocas. O pior desempenho ocorreu com a quantidade de 200 iterações. De forma semelhante, o experimento 2 também teve o pior desempenho bruto com 200 iterações, porém houve variação significativa apenas para a quantidade 800 e 1000 épocas. A Figura 38 demonstra estes resultados através do diagrama de caixa.

Figura 38: Comparativo dos resultados da avaliação de imagem pela paleta



Fonte: elaborado pelo autor

A análise tomando somente a quantidade de 200 épocas como referência já permite observar a variação no desempenho de forma mais geral, identificando se de fato houve melhora ao fazer mais treino. Porém, para uma análise ainda mais crítica convém comparar as demais quantidades de épocas entre si, permitindo filtrar mais detalhadamente as faixas onde já se viu que a variação foi significativa. Para este fim foram construídas versões completas desta comparação que são vistas a seguir. Estas matrizes podem ser observadas tanto por linha quanto por coluna, pois os dados de comparações iguais estão replicados. Além disso, os valores-p significativos estão destacados em negrito para melhorar a visualização.

A partir disso, pode ser verificado com mais detalhes cada situação que ocorreu, iniciando pelo experimento 1 que está retratado na Tabela 3. Observando o cenário da avaliação binária, nota-se variação significativa em todas as comparações exceto na relação 200-1000. Aliado a este fato, nota-se que a iteração 400 obteve a menor média e que foi significativa para todas as comparações, indicando que seu desempenho foi de fato o melhor. Enquanto isso, na avaliação pela paleta deste experimento 1, observou-se ante-

riormente que a quantidade de 200 iterações teve o pior resultado pois obteve a maior média e foi suficientemente diferente das demais. A análise completa das demais iterações revelou que nenhuma delas foi significativamente melhor que as demais. Ou seja, a partir de 200 iterações o desempenho medido por esta técnica de avaliação melhorou mas não variou de forma relevante.

Tabela 3: Comparação completa do experimento 1 por quantidade de épocas de treino

Aval.	**Iter.**	**Experimento 1**					
		Média	**200**	**400**	**600**	**800**	**1000**
Binária	200	0,0340	-	**<0,001**	**<0,001**	**<0,001**	0,812
	400	0,0205	**<0,001**	**-**	**<0,001**	**0,016**	**<0,001**
	600	0,0284	**<0,001**	**<0,001**	**-**	**<0,001**	**<0,001**
	800	0,0229	**<0,001**	**0,016**	**<0,001**	**-**	**<0,001**
	1000	0,0336	0,812	**<0,001**	**<0,001**	**<0,001**	**-**
Paleta	200	0,3660	-	**<0,001**	**0,002**	**0,001**	**<0,001**
	400	0,3430	**<0,001**	-	0,503	0,796	0,576
	600	0,3480	**0,002**	0,503	-	0,551	0,144
	800	0,3450	**0,001**	0,796	0,551	-	0,346
	1000	0,3390	**<0,001**	0,576	0,144	0,346	-

Fonte: elaborado pelo autor

Analisando agora o panorama geral do experimento 2, na Tabela 4, observa-se melhor a situação ocorrida na avaliação binária, em que o aumento de iterações causou num primeiro momento uma queda na performance nas 400 e 600 iterações, para então recuperar o mesmo patamar original ao rodar com 800 e 1000 iterações. Um padrão semelhante surgiu ao comparar as demais iterações, destacando que a faixa entre 400 e 600 iterações teve desempenho abaixo das demais. A quantidade de 800 repetições só foi significativamente melhor que o resultado com 600 épocas, ao qual está associada a maior média. Disso conclui-se que somente as quantidade de 200 e 1000 repetições tiveram resultado realmente melhor que todas as demais épocas. A explicação para este comportamento que ora melhora e ora piora ao aumentar as iterações pode ter relação com a descida de gradiente que foi estudada na seção 4.1 do capítulo que abordou os conceitos de redes neurais. Como visto naquele momento, a RNA possui uma taxa de aprendizado que se relaciona com a quantidade de épocas executadas. Se essa taxa é muito baixa, a RNA se aproxima muito devagar da função de custo mínima e demora até atingir o melhor desempenho. Enquanto isso, se essa taxa de aprendizado for muito alta, pode fazer com que a função de custo ultrapasse o ponto mínimo e comece a retroceder e avançar até atingir o melhor ponto, o que gera flutuações no desempenho. (NIELSEN, 2015)

Por outro lado, a investigação dos resultados pela paleta de cores demonstra claramente que a faixa de 800 até 1000 repetições apresentou melhora em relação às demais

quantidades e que elas foram equivalentes entre si. Existem duas explicações possíveis para estes cenário. A primeira é que o gradiente para atender esta modalidade de avaliação estava baixo, ou seja, a RNA demorou para assimilar o treinamento e melhorar o resultado. A segunda possibilidade é a ocorrência do *overfitting* (seção 4.2), que normalmente é ruim pois faz com que a RNA perca flexibilidade, porém, é possível que neste caso do *pixel art* tenha ajudado. Como muitas vezes o padrão de cores se repete, por exemplo ao representar o rosto do personagem com um conjunto específico de pixels, pode ser que um estado de *overfitting* auxilie a gerar na saída uma imagem mais precisa nessas regiões com mais detalhes.

Tabela 4: Comparação completa do experimento 2 por quantidade de épocas de treino

Aval.	**Iter.**	**Experimento 2**					
		Média	**200**	**400**	**600**	**800**	**1000**
Binária	200	0,0133	-	**0,027**	**0,001**	0,090	0,869
	400	0,0164	**0,027**	**-**	0,143	0,344	**0,015**
	600	0,0182	**0,001**	0,143	**-**	**<0,001**	**0,001**
	800	0,0153	0,090	0,344	**<0,001**	**-**	0,052
	1000	0,0131	0,869	**0,015**	**0,001**	0,052	**-**
Paleta	200	0,3030	-	0,450	0,769	**<0,001**	**<0,001**
	400	0,2980	0,450	-	0,357	**0,002**	**<0,001**
	600	0,3040	0,769	0,357	-	**<0,001**	**<0,001**
	800	0,2750	**<0,001**	**0,002**	**<0,001**	-	0,721
	1000	0,2730	**<0,001**	**<0,001**	**<0,001**	0,721	-

Fonte: elaborado pelo autor

9.4 RESULTADOS PELA DIFICULDADE DE REPRESENTAÇÃO

Finalmente, a última análise dos resultados pode ser feita agrupando pela categoria de dificuldade, que classificou as imagens entre fáceis, médias e difíceis de acordo com o grau de aplicação das habilidades de desenho. Essa avaliação foi feita agrupando o experimento e o método de avaliação (binária ou paleta) e variando o identificador de dificuldade, sendo representada na Tabela 5.

Houve situações como a avaliação binária do experimento 1 e a avaliação paleta do experimento 2 em que as imagens fáceis obtiveram desempenho médio pior do que as difíceis, enquanto que nos outros casos ocorreu o contrário. Além disso, em todas as situações o melhor desempenho ocorreu nas imagens de dificuldade média. Porém, como pode ser observado em todos os cenários o valor-p ficou superior a 0,05. Isso significa que essa diferença numérica dos resultados não é significativa o suficiente para inferir que houve relação entre a categoria de imagem e o desempenho do processo gerador.

Ou seja, apesar de a análise qualitativa apontar diferenças na dificuldade de representação da imagem pelo artista, o processo transcorreu de forma semelhante entre

Tabela 5: Comparação dos escores por dificuldade

Experimento	Avaliação	Dificuldade	N(%)	Média	E.P.	1°Q.	2°Q.	3°Q.	Valor-p
Experimento 1 (Geração completa de um sprite)	Binária	Fácil	16 (32,00%)	0,0295	0,0031	0,0180	0,0296	0,0370	0,1888
		Médio	21 (42,00%)	0,0269	0,0023	0,0216	0,0260	0,0334	
		Difícil	13 (26,00%)	0,0274	0,0091	0,0075	0,0144	0,0378	
	Paleta	Fácil	16 (32,00%)	0,3617	0,0166	0,3183	0,3460	0,4059	0,3815
		Médio	21 (42,00%)	0,3342	0,0087	0,3022	0,3219	0,3712	
		Difícil	13 (26,00%)	0,3542	0,0147	0,3077	0,3412	0,3784	
Experimento 2 (Coloração de um sprite)	Binária	Fácil	16 (32,00%)	0,0152	0,0021	0,0096	0,0154	0,0177	0,3743
		Médio	21 (42,00%)	0,0144	0,0018	0,0091	0,0138	0,0164	
		Difícil	13 (26,00%)	0,0167	0,0064	0,0080	0,0095	0,0129	
	Paleta	Fácil	16 (32,00%)	0,3057	0,0143	0,2627	0,2840	0,3471	0,2071
		Médio	21 (42,00%)	0,2770	0,0099	0,2446	0,2660	0,2995	
		Difícil	13 (26,00%)	0,2945	0,0143	0,2521	0,2814	0,3212	

Fonte: elaborado pelo autor

todas as categorias, obtendo escores equivalentes ao final desta técnica de avaliação. Para entender um dos motivos de não ocorrer essa relação, pode ser comparado o processo de geração da RNA com o processo criativo de um artista. O artista deve considerar diversos aspectos da imagem na hora de construir a sua representação, como por exemplo os conceitos de escorço e volumetria abordados na seção 3.3 e na subseção 8.1.3. Enquanto isso, a RNA geradora de imagens abstrai esta complexidade e busca identificar apenas os padrões existentes. Ou seja, a RNA não aborda diretamente nenhum desses conceitos de desenho, se limitando a reproduzir as situações vistas durante o treinamento.

Nesse sentido, o modelo de mapeamento de pose proposto na subseção 8.1.2 se mostrou suficiente para gerar todas as imagens de forma homogênea entre as categorias de dificuldade. Por exemplo, houve o cuidado de representar a ocorrência de escorço através do controle da ordem de desenho dos elementos no modelo de pose. Com isso, os membros do corpo que ficavam atrás de outros apareciam desta forma no modelo, o que auxiliou a RNA a conseguir assimilar essa informação e poder representá-la nas imagens consideradas difíceis. Outros aspectos como a fonte de iluminação e volumetria que não estavam representados diretamente no modelo não foram relevantes o suficiente para prejudicar o desempenho nas imagens difíceis.

Com tudo isso, conclui-se que o processo proposto obteve bom desempenho, atingindo métricas de diferença na faixa de 15% até 30% para os cenários verificados. Efetuar o treinamento com mais épocas ajudou a reduzir as diferenças detectadas na maior parte das situações, porém, a quantidade exata que alcança melhores resultados depende do aspecto que está sendo avaliado. Quanto ao processo de análise, a observação dos resultados utilizando uma metodologia bem definida permitiu entender o comportamento da RNA geradora de imagens neste contexto do *pixel art* e relacionar aos conceitos estudados. Além disso, ajudou a identificar possíveis melhorias no próprio método de verificação. Finalmente, os resultados são promissores para que seja gerado um esboço inicial de imagens com boa estrutura, sob a qual o artista possa fazer o refinamento e finalização.

10 CONSIDERAÇÕES FINAIS

Conclui-se que o método científico é indispensável para manter o rigor e a organização dos trabalhos no decorrer de um projeto. Estruturar e estudar uma fundamentação teórica nos primeiros capítulos se mostrou importante para construir o processo consciente de quais partes podem ser flexibilizadas e quais são cuidados a serem tomados. Este embasamento teórico que contempla o panorama geral do processo também pode servir de ponto de partida para estudos mais aprofundados em cada área específica. Vários dos conceitos vistos na teoria também foram imprescindíveis na etapa final para entender os resultados obtidos e identificar as melhorias e deficiências desta proposta. Além disso, o trabalho apresentou um método de desenvolvimento claro e objetivo, trazendo como contribuição científica a possibilidade de ser reproduzido em estudos futuros segundo as diretivas que foram apresentadas.

Percebe-se também que a problemática que define o tema deste trabalho é bastante atual e possui relevância. Como foi visto no início dos estudos, a técnica do *pixel art* se tornou ícone de nostalgia e provém um visual único ao jogador, tendo respaldo teórico que fundamenta essas características. Enquanto isso, o estudo de redes neurais encontra aplicações nos mais variados contextos, e neste projeto foi possível aliar essas técnicas da computação aos conceitos artísticos da indústria de jogos, mais especificamente ao conseguir atingir o estilo de *pixel art* a partir do processo gerador de uma rede neural. Os conceitos estudados inicialmente parecem não possuir relação direta, porém, no decorrer do trabalho foram vistos exemplos de aplicações práticas que não só auxiliam a entender e fixar a teoria como ressaltam a importância da troca de conhecimento entre áreas de atuação distintas. Relacionar essas duas áreas se mostrou uma tarefa desafiadora, porém, o estudo prévio forneceu o embasamento necessário. Aliando ferramentas que auxiliam o uso de RNA com técnicas de programação vistas durante a graduação, foi possível atingir os objetivos, que são retomados a seguir.

Neste projeto pretendia-se investigar, propor e implementar um processo de geração de imagens bidimensionais por meio de uma rede neural, inserido no desenvolvimento de jogos com personagens em *pixel art*. A etapa de investigação foi atendida nos capítulos iniciais de estudo do referencial teórico e das implementações práticas que poderiam ser utilizadas. Após isso, foi delineada no Capítulo 7 a proposta principal, com a definição de um roteiro de aplicação do processo contendo os cuidados necessários em cada etapa. Por último, a partir do Capítulo 8, foi visto o desenvolvimento prático que possibilitou o entendimento do uso real da proposta e gerou os dados para a análise de resultados.

Quanto aos resultados, foi possível observar relações entre os diferentes cenários experimentados. De modo geral, foi verificado que a maior quantidade de treino auxilia

a aprimorar os resultados, porém o número exato depende de qual aspecto da saída é mais relevante para a qualidade desejada. Em especial, destaca-se como contribuição a descoberta de que a classificação das imagens de acordo com a dificuldade de desenho não influiu de forma significativa nas métricas de saída em nenhum dos cenários estudados. Ou seja, mesmo com a avaliação técnica tendo identificado diferentes graus de complexidade ao representar as imagens, isso foi irrelevante sob a ótica do processo proposto utilizando redes neurais, que gerou saídas de forma homogênea para todos os casos.

Com o estudo dos resultados e das limitações existentes, foi possível identificar diversas melhorias a implementar em trabalhos futuros. A aplicação, que neste projeto se limitou a apenas um personagem em *pixel art*, pode ser estendida para outros contextos, como, por exemplo, figuras que não sejam humanoides, como animais ou objetos, bastando que seja criado e testado um modelo de mapeamento de pose adequado a este uso. Além disso, o algoritmo de normalização que gera efetivamente o *pixel art* ao final do processo partiu de uma tática bastante simplória de correspondência por aproximação das cores que foram geradas pela RNA. Podem ser estudados algoritmos mais complexos que extraiam características da imagem de saída e melhorem com isso o desempenho. Por fim, a avaliação utilizou comparações pixel-a-pixel da imagem, podendo ser aplicadas outras técnicas de cálculo de similaridade entre imagens, que permitam outros tipos de análise dos resultados.

Finalmente, nota-se que o processo que ficou estabelecido em etapas isoladas e claramente definidas permite trabalhar melhorias individualmente. Ou seja, uma vez respeitados os objetivos, método de integração e boas práticas indicados em cada uma dessas etapas, podem ser usadas ou desenvolvidas outras táticas sem comprometer o fluxo de informações e resultados gerais esperados. Por fim, ressalta-se que esta pesquisa possui aplicação prática possível no contexto a que se propõe do desenvolvimento de jogos, contribui cientificamente com o processo gerador de imagens apresentado e abre opções para continuidade da investigação em frentes específicas da proposta.

REFERÊNCIAS

AMBERER. *Image-to-Image Translation*. 2019. Disponível em: <https://amberer-.gitlab.io/papers_in_ai/img2img-translation.html>. Acesso em: 24 maio 2019. Citado na página 47.

BABU, S. C. *A 2019 guide to Human Pose Estimation with Deep Learning*. 2019. Disponível em: <https://blog.nanonets.com/human-pose-estimation-2d-guide/>. Acesso em: 19 maio 2019. Citado na página 43.

BARROS, P. *Aprendizagem de Maquina: Supervisionada ou Não Supervisionada?* 2016. Disponível em: <https://medium.com/opensanca/aprendizagem-de-maquina-supervisionada-ou-n%C3%A3o-supervisionada-7d01f78cd80a>. Acesso em: 16 maio 2019. Citado 2 vezes nas páginas 34 e 40.

BECKER, A. *12 Principles of Animation (Official Full Series)*. 2017. Disponível em: <https://youtu.be/uDqjIdI4bF4?t=393>. Acesso em: 26 abr. 2019. Citado na página 20.

CAO, Z. *et al.* OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In: *arXiv preprint arXiv:1812.08008*. [S.l.: s.n.], 2018. Citado 2 vezes nas páginas 50 e 51.

CAO, Z. *et al.* Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016. Disponível em: <http://arxiv.org/abs/1611.08050>. Citado na página 50.

CHAN, C. *et al.* Everybody dance now. *CoRR*, abs/1808.07371, 2018. Citado na página 15.

CONLEY, J. *et al.* Use of a game over: emulation and the video game industry, a white paper. *Nw. J. Tech. & Intell. Prop.*, HeinOnline, v. 2, p. 261, 2003. Citado na página 46.

COOK, D. *Design Testing: The use of addiction metrics to force rapid evolution of innovative game designs*. 2019. Disponível em: <http://www.lostgarden.com/2005/04-/design-testing-use-of-addiction.html>. Acesso em: 13 abr. 2019. Citado 2 vezes nas páginas 16 e 17.

CRAWFORD, G. *Video Gamers*. [S.l.: s.n.], 2012. Citado na página 16.

EDWARDS, B. *Drawing on the Right Side of the Brain: The Definitive, 4th Edition*. Penguin Publishing Group, 2012. ISBN 9781101561805. Disponível em: <https:/-/books.google.com.br/books?id=ofXvl2eDsnwC>. Citado 3 vezes nas páginas 25, 26 e 27.

FA, D. L. K. *2xSaI : The advanced 2x Scale and Interpolation engine*. 1999. Disponível em: <https://web.archive.org/web/20070307160526/http://elektron.its.tudelft.nl-/~dalikifa/>. Acesso em: 24 maio 2019. Citado na página 46.

FREITAS, C. E. *et al.* Um processo Ágil multidisciplinar de desenvolvimento de jogos para estúdios independentes. *XVI SBGames*, Curitiba, PR, p. 1232–1235, 11 2017. Citado 3 vezes nas páginas 13, 17 e 18.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. `http://www.deeplearningbook.org`. Citado na página 41.

GOODFELLOW, I. J. *et al.* Generative adversarial nets. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Cambridge, MA, USA: MIT Press, 2014. (NIPS'14), p. 2672–2680. Disponível em: <http://dl.acm.org/citation.cfm?id=2969033.2969125>. Citado 2 vezes nas páginas 40 e 42.

GUPTA, V. *Deep Learning based Human Pose Estimation using OpenCV (C++ / Python)*. 2018. Disponível em: <https://www.learnopencv.com/deep-learning-based-human-pose-estimation-using-opencv-cpp-python/>. Acesso em: 21 maio 2019. Citado na página 45.

GUTTAG, K. *Transcript of Conference on Delphi TI Net*. 1993. Disponível em: <https://groups.google.com/forum/!msg/comp.sys.ti/QlLThkrm8Po/GtnkyhtF4nMJ>. Acesso em: 07 jun. 2019. Citado na página 21.

HAYKIN, S. *Redes Neurais: Princípios e Prática*. [S.l.]: Artmed, 2007. ISBN 9788577800865. Citado 3 vezes nas páginas 28, 29 e 30.

HOLLANDER, M.; WOLFE, D. *Nonparametric Statistical Methods*. 3. ed. New York, N.Y.: John Wiley & Sons, 1999. (Wiley Series in Probability and Statistics). Citado na página 76.

ISOLA, P. *et al.* Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 5967–5976, 2017. Citado 4 vezes nas páginas 14, 53, 54 e 70.

JURIO, A. *et al.* Image magnification using interval information. *IEEE Transactions on Image Processing*, IEEE, v. 20, n. 11, p. 3112–3123, 2011. Citado na página 46.

KARRAS, T.; LAINE, S.; AILA, T. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. Citado 2 vezes nas páginas 15 e 49.

KOGAN, G.; TSENG, F.; REFSGAARD, A. *Pix2Pix*. 2016. Disponível em: <https://ml4a.github.io/guides/Pix2Pix/>. Acesso em: 03 set. 2019. Citado na página 59.

KOPF, J.; LISCHINSKI, D. Depixelizing pixel art. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, v. 30, n. 4, p. 99:1 – 99:8, 2011. Citado na página 46.

KUO, M.-H.; YANG, Y.-L.; CHU, H.-K. Feature-aware pixel art animation. *Comput. Graph. Forum*, The Eurographs Association & John Wiley & Sons, Ltd., Chichester, UK, v. 35, n. 7, p. 411–420, out. 2016. ISSN 0167-7055. Disponível em: <https://doi.org/10.1111/cgf.13038>. Citado 2 vezes nas páginas 13 e 46.

LEE, H. *et al.* Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009. (ICML '09), p. 609–616. ISBN 978-1-60558-516-1. Disponível em: <http://doi.acm.org/10.1145/1553374.1553453>. Citado na página 38.

LIANG, K.-Y.; ZEGER, S. L. Longitudinal data analysis using generalized linear models. *Biometrika*, v. 73, n. 1, p. 13–22, 04 1986. ISSN 0006-3444. Disponível em: <https://doi.org/10.1093/biomet/73.1.13>. Citado na página 76.

LIU, M.-Y. *et al.* Few-shot unsupervised image-to-image translation. *arXiv preprint arXiv:1905.01723*, 2019. Citado na página 48.

MAZZOLENI, A. *Scale2x*. 2001. Disponível em: <https://www.scale2x.it/>. Acesso em: 24 maio 2019. Citado na página 46.

MELO, C. P. de. *O Corpo Humano Em Escorço*. Tese (Doutorado) — Universidade de Lisboa, 2011. Citado na página 68.

NELDER, J. A.; WEDDERBURN, R. W. M. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, v. 135, n. 3, p. 370–384, 1989. Disponível em: <https://rss.onlinelibrary.wiley.com/doi/abs/10.2307/2344614>. Citado na página 76.

NEWELL, A.; YANG, K.; DENG, J. Stacked hourglass networks for human pose estimation. In: *Computer Vision – ECCV 2016*. [S.l.: s.n.], 2016. Citado na página 44.

NIELSEN, M. *Neural Networks and Deep Learning*. Determination Press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/index.html>. Citado 5 vezes nas páginas 29, 30, 34, 57 e 83.

OMERNICK, M. *Creating the Art of the Game*. Pearson Education, 2004. (New Riders Games). ISBN 9780132705073. Disponível em: <https://books.google.com.br-/books?id=27-4GmyMBGEC>. Citado na página 13.

PATTERSON, J.; GIBSON, A. *Deep Learning: A Practitioner's Approach*. [S.l.]: O'Reilly Media, 2017. ISBN 9781491914212. Citado 2 vezes nas páginas 36 e 39.

PRESSMAN, R.; MAXIM, B. *Engenharia De Software: UMA ABORDAGEM PROFISSIONAL*. MCGRAW HILL - ARTMED, 2016. ISBN 9788580555332. Disponível em: <https://books.google.com.br/books?id=smNFvgAACAAJ>. Citado 2 vezes nas páginas 13 e 16.

RADFORD, A.; METZ, L.; CHINTALA, S. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2015. Citado na página 41.

RAJ, B. *An Overview of Human Pose Estimation with Deep Learning*. 2019. Disponível em: <https://medium.com/beyondminds/an-overview-of-human-pose-estimation-with-deep-learning-d49eb656739b>. Acesso em: 19 maio 2019. Citado na página 43.

RAND, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, Taylor & Francis, v. 66, n. 336, p. 846–850, 1971. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1971-.10482356>. Citado na página 22.

RIDPATH, C.; CHISHOLM, W. *Techniques For Accessibility Evaluation And Repair Tools*. 2000. Disponível em: <https://www.w3.org/TR/AERT/color-contrast>. Acesso em: 29 set. 2019. Citado na página 23.

SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. Acesso em: 11 maio 2019. Citado 4 vezes nas páginas 35, 36, 37 e 38.

SANDERSON, G. *Gradient descent, how neural networks learn | Deep learning, chapter 2*. 2017. Disponível em: <https://www.youtube.com/watch?v=IHZwWFHWa-w>. Acesso em: 6 maio 2019. Citado na página 34.

SCHELL, J. *The Art of Game Design: A Book of Lenses*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008. ISBN 0-12-369496-5. Citado 2 vezes nas páginas 16 e 18.

SHARMA, A. *Understanding Activation Functions in Neural Networks)*. 2017. Disponível em: <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>. Acesso em: 5 maio 2019. Citado 3 vezes nas páginas 30, 31 e 32.

SILBER, D. *Pixel Art for Game Developers*. Taylor & Francis, 2015. ISBN 9781482252309. Disponível em: <https://books.google.com.br/books?id=n0zRrQEACAAJ>. Citado 3 vezes nas páginas 18, 19 e 20.

SILVA, T. *An intuitive introduction to Generative Adversarial Networks (GANs)*. 2018. Disponível em: <https://medium.freecodecamp.org/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394>. Acesso em: 16 maio 2019. Citado 2 vezes nas páginas 41 e 42.

STEPIN, M. *hq4x Magnification Filter*. 2001. Disponível em: <https://web.archive.org-/web/20131216092117/http://www.hiend3d.com/hq4x.html>. Acesso em: 24 maio 2019. Citado 3 vezes nas páginas 46, 51 e 52.

THOMAS, F.; JOHNSTON, O. *The Illusion of Life: Disney Animation*. Disney Editions, 1981. Disponível em: <https://books.google.com.br/books?id=k5TMoAEACAAJ>. Citado na página 20.

TOMPSON, J. *et al.* Efficient object localization using convolutional networks. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2015. Citado na página 44.

TOSHEV, A.; SZEGEDY, C. Deeppose: Human pose estimation via deep neural networks. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2014. Citado na página 43.

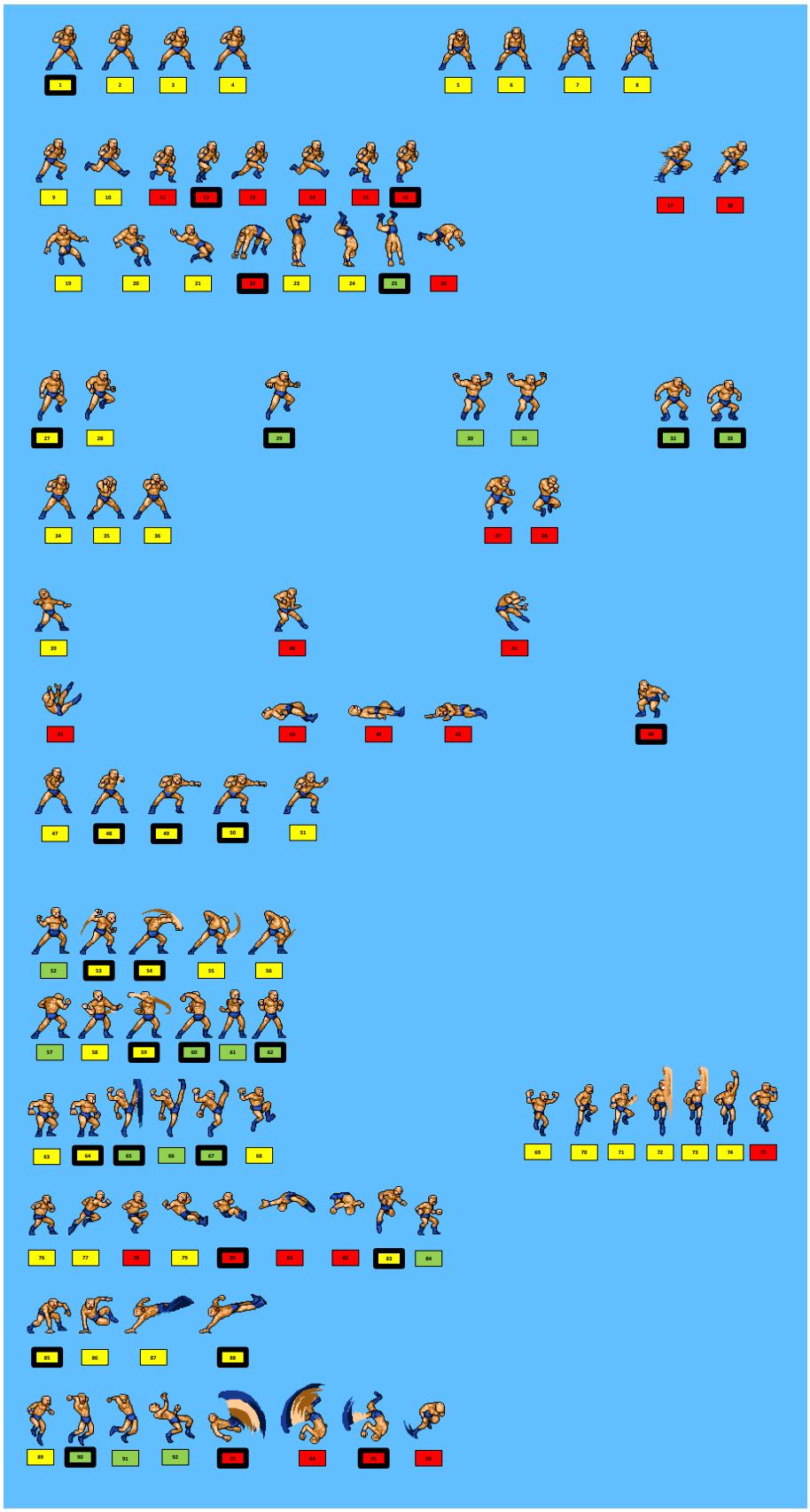
TRIPATHY, S.; KANNALA, J.; RAHTU, E. Learning image-to-image translation using paired and unpaired training samples. *arXiv preprint arXiv:1805.03189*, 2018. Citado na página 48.

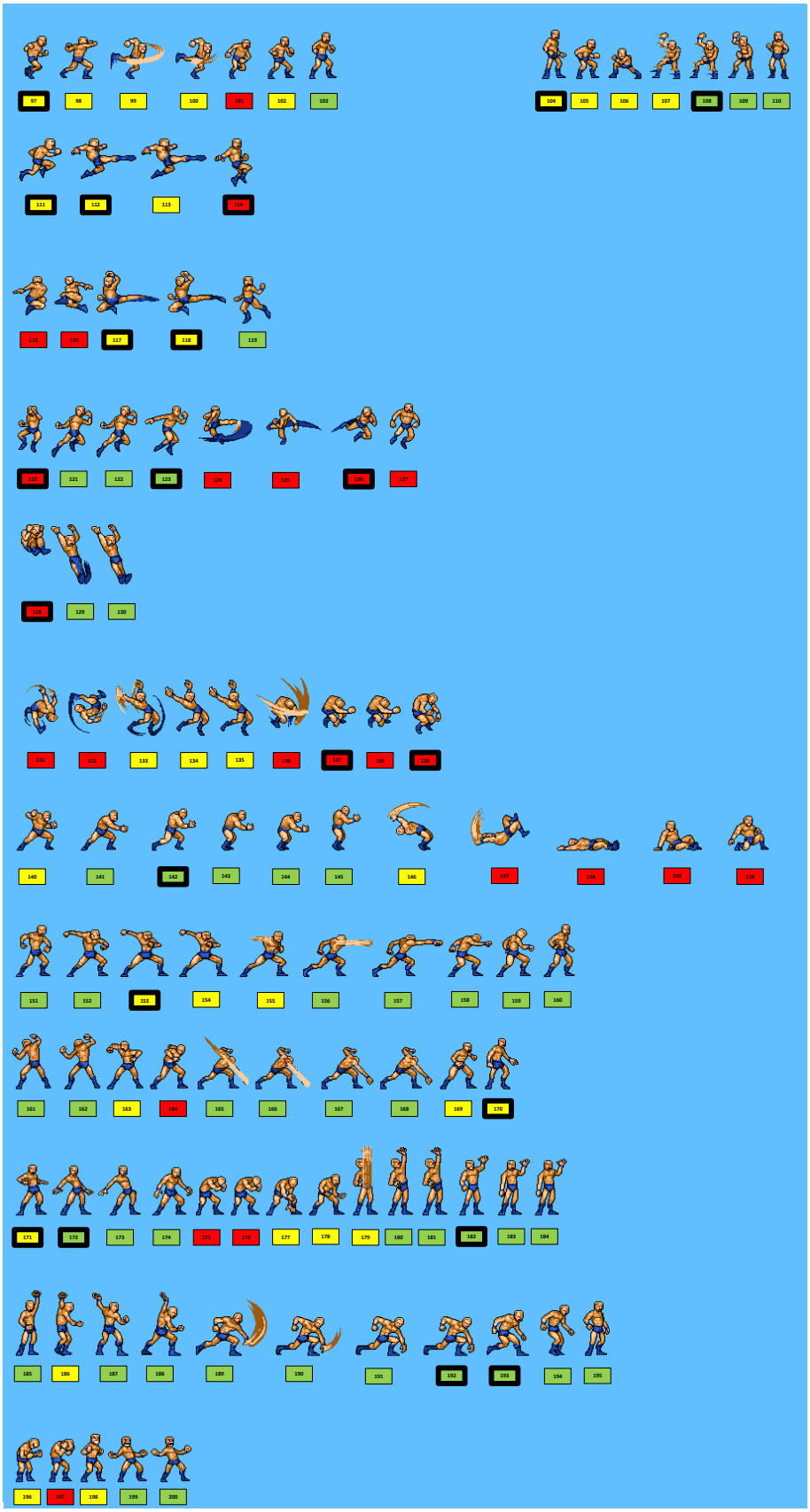
WEI, S.-E. *et al.* Convolutional pose machines. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. Citado na página 44.

ZHU, J.-Y. *et al.* Unpaired image-to-image translation using cycle-consistent adversarial networkss. In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. [S.l.: s.n.], 2017. Citado na página 48.

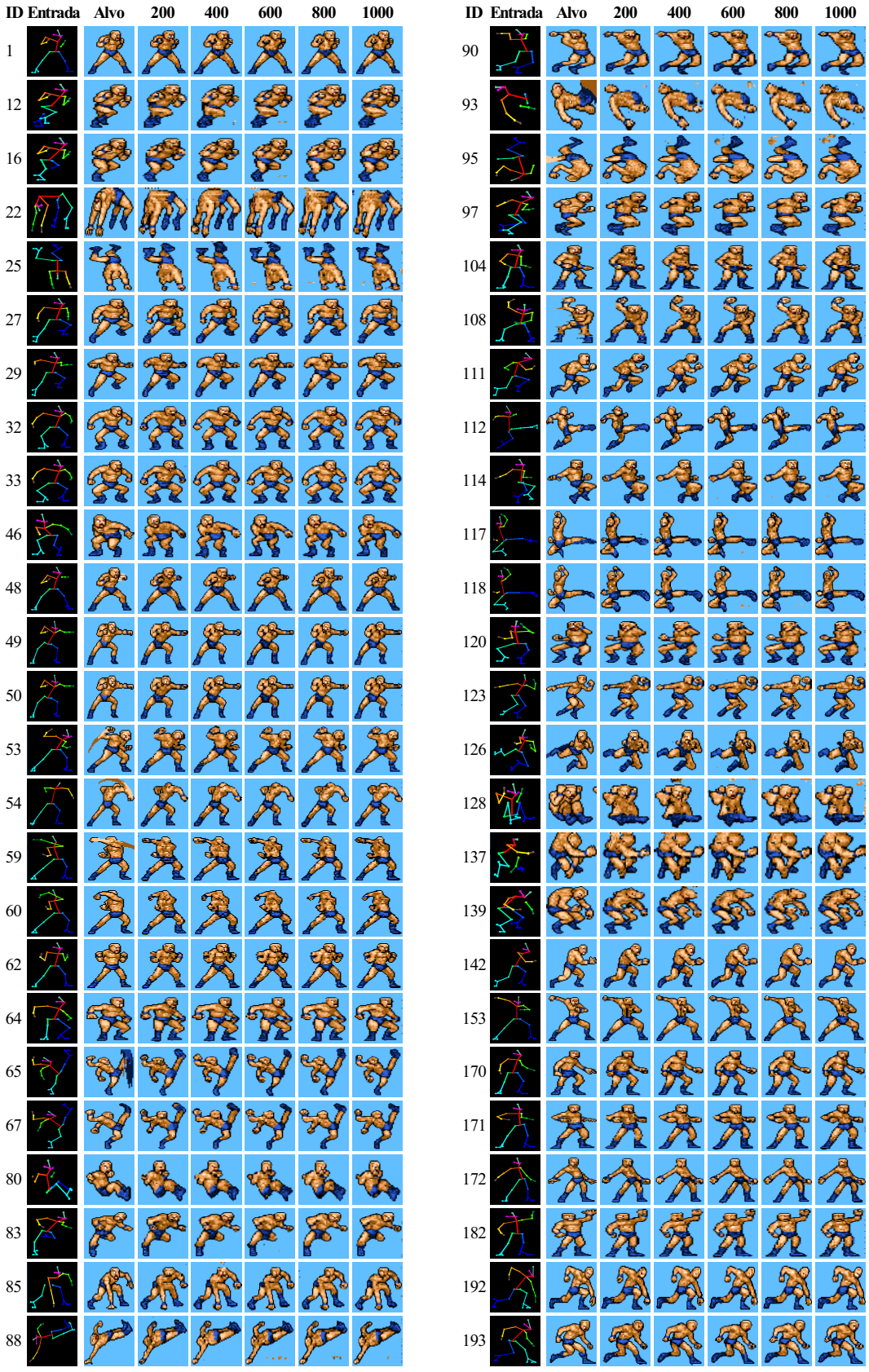
Apêndices

A *SPRITE SHEET* COMPLETO





B RESULTADOS - EXPERIMENTO 1



C RESULTADOS - EXPERIMENTO 2