Reconhecimento Facial com Super-Resolução: uma abordagem utilizando Redes Generativas e Joint-Learn

Rafael Augusto de Oliveira - a42879

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Prof. Dr. Pedro João Soares Rodrigues

Prof. Dr. Arnaldo Candido Junior

Prof. Dr. Pedro Luiz de Paula Filho

Bragança

2019-2020

Reconhecimento Facial com Super-Resolução: uma abordagem utilizando Redes Generativas e Joint-Learn

Mestrado em Sistemas de Informação

Escola Superior de Tecnologia e Gestão

Rafael Augusto de Oliveira - a42879

Trabalho orientado por:

Prof. Dr. Pedro João Soares Rodrigues

Prof. Dr. Arnaldo Candido Junior

Prof. Dr. Pedro Luiz de Paula Filho

Bragança

2019-2020

Dedicatória

Para minha mãe Janete Terezinha Bernardi de Oliveira, com todo o meu amor e carinho.

Agradecimentos

Agradeço ao meu pai João Batista de Oliveira, minha madrinha Zélia Bernardi, minha namorada Margarida Santos, e minhas primas Carolina e Gabriela Guzzo e Jussimara Macedo, por todo o amor e suporte que me dão para que eu possa fazer o meu melhor a cada dia. Vocês fazem tudo valer a pena. Amo todos vocês, com todo o meu coração.

Também agradeço aos meus irmãos por consideração, João Pedro e Marcos Nespolo, Eduardo Ribeiro, João Pedro Teixeira, Carlos Bertoncelli, Raian Westin, e Gerson "Sucrilhos" Coelho Junior, e a minha família em Bragança composta por Ana Cláudia Marques, Andressa Mazur, Andrezza Miranda, Breno Borba, Caio Rodrigues, Giuseppe Setem, João Jorge Coelho e Matheus Wilkerd, por toda ajuda, aprendizado e companheirismo.

Um agradecimento especial ao meus orientadores Arnaldo Candido Junior, Pedro João Soares Rodrigues, e Pedro Luiz de Paula Filho, por me acompanharem nessa caminhada pela graduação e pós-graduação. Seus ensinamentos foram valiosos.

Agradeço também as duas instituições, Universidade Tecnológica Federal do Paraná (UTFPR) e Instituto Politécnico de Bragança (IPB), por proporcionarem o acordo de duplo-diploma e pelas oportunidades que me foram concedidas na graduação e no mestrado.

Finalmente, agradeço a cada um que me ajudou nessa jornada. Muito obrigado!

Resumo

Câmeras de monitoramento são largamente utilizadas para supervisionar estabelecimentos, de forma a coibir atos de violência. Um dos modos de melhorar esse sistema é pelo reconhecimento das pessoas que circulam nesse espaço, preferencialmente pelo rosto dos indivíduos. Um desafio existente é reconhecer os rostos quando as condições de imagem são adversas, seja pela utilização de equipamentos de baixa qualidade ou pela distância entre o sujeito e a câmera, impactando assim nas taxas de acerto dos sistemas de reconhecimento. Para tal, técnicas de aumento de resolução e melhoria na qualidade das imagens podem ser aplicadas antes de realizar o reconhecimento da face, de forma a melhorar a acurácia da última. Dentre essas técnicas de Super-Resolução, o atual estado da arte dá-se pelo uso de Redes Adversárias Generativas (GAN). No uso conjunto de Super-Resolução com reconhecimento de faces, uma opção que se mostrou promissora é o treinamento das redes de Super-Resolução e Reconhecimento Facial de forma conjunta, de modo a direcionar a rede no aprendizado de características de aumento de qualidade de imagem que conduzam a uma melhor acurácia em reconhecer os indivíduos. Neste trabalho, realizamos o treinamento conjunto de Super-Resolução e Reconhecimento de Faces, sendo a primeira uma rede Generativa e a última uma ResNet50, treinados com o auxílio de uma rede Discriminativa, nos moldes do treinamento de redes GAN, de forma a testar a eficácia desse sistema no reconhecimento dos indivíduos em imagens de baixa qualidade. As imagens de resolução aumentada geradas pela rede foram satisfatórias, mas não conseguimos realizar a convergência do reconhecimento das faces em tempo hábil. Com o presente trabalho, desejamos que os achados em nossos experimentos sirvam de insumo para mais pesquisas nesse tópico. Os códigos estão disponíveis publicamente

em `https://github.com/OliRafa/SRFR-GAN`.

Palavras-chave: Super-Resolução, Reconhecimento de Faces, Redes Adversárias Generativas, Aprendizado de Máquina.

Abstract

Surveillance cameras are broadly used in supervising private places to restrain violent acts. One of the ways of improving this system is recognizing people in this space, preferably by using an individual's face biometrics. An existing challenge is to recognize faces when imaging conditions are adverse, either by low-quality cameras or the distance between the subject and the camera, thus impacting the accuracy of these recognizing systems. Super-Resolution (SR) techniques can be used to improve both image resolution and quality before recognizing the face, to improve the accuracy of the recognition task. Among these techniques, the actual State of the Art uses Generative Adversarial Networks (GAN). When used together, one promising option is to train Super-Resolution and Face Recognition as one single network, conducting the network to learn SR features that will improve its capability when recognizing faces. In the present work, we trained a Super-Resolution Face Recognition model using this joint-learn approach, combining a Generative network for SR, and a ResNet50 for Face Recognition. The model was trained with a Discriminator network, following the GAN training framework. The images generated by the network were convincing, but we couldn't converge the FR model in a timely manner. We hope that our contributions could help future works on this topic. Code is publicly available at `https://github.com/OliRafa/SRFR-GAN`.

Keywords: Super-Resolution, Face Recognition, Generative Adversarial Networks, Machine Learning.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A utilização de câmeras de monitoramento é um dos métodos mais conhecidos para segurança física em estabelecimentos, de forma a coibir que atividades ilícitas tais como violência ou furto ocorram [1], apesar de sua eficácia ainda ser controversa [2]. Para aumentar a eficácia desse sistema e, portanto, auxiliar no monitoramento desses locais, reconhecer as pessoas que ali estão é de grande importância, sendo esse um dos tópicos de interesse dentro da área da Visão Computacional e do Reconhecimento de Padrões.

O *Face Recognition* (FR) (em português, Reconhecimento Facial) está presente em diversas tarefas como aplicação de filtros em faces para fotos e vídeos nas redes sociais, desbloqueio de aparelhos, permitir a entrada em estabelecimentos, ou reconhecer pessoas desaparecidas. O RF utiliza características (ou biometrias) extraídas da face de indivíduos e, posteriormente, utiliza-as para identificar quem são os sujeitos naquela imagem ou vídeo. Para que as características sejam extraídas, o RF depende primeiramente do *Face Detection* (FD) (em português, Detecção de Face), na qual busca-se identificar os rostos nas imagens para, posteriormente, extrair as biometrias.

Existe uma série de desafios para o RF, tais como a resolução das câmeras atualmente instaladas na maioria dos locais, a grande distância entre a câmera e o sujeito que gera baixa resolução na imagem e, por fim, possíveis artefatos nas imagens. Devido a esses problemas é muitas vezes impraticável o uso de uma aplicação em larga escala baseada em RF [3].

Como uma parcela significativa das câmeras de segurança instaladas atualmente são de baixa resolução, atualizar esse sistema para câmeras com alta resolução seria um investimento que muitos estabelecimentos não poderiam arcar. Seguindo essa premissa, além de ter de atualizar as câmeras, uma maior resolução de imagem implica também em maior tráfego na rede e na necessidade de maiores espaços para armazenamento dessas imagens, necessitando assim que toda a infraestrutura seja atualizada de uma vez, elevando ainda mais o investimento necessário.

Mesmo com câmeras de alta resolução, os problemas quanto a baixa resolução devido a distância entre sujeito e câmera e os possíveis artefatos na imagem ainda seriam existentes, reduzindo assim a possibilidade de acerto na identificação do sujeito na imagem.

De forma a aprimorar esse sistema, técnicas de *Super-Resolution* (SR) (em português, Super-Resolução) podem ser aplicadas para melhorar tanto a resolução quanto a qualidade das imagens.

A SR é um método para reconstrução de imagens de alta qualidade a partir de uma ou mais imagens de baixa qualidade da mesma cena, de modo a reduzir as degradações existentes. Suas aplicações são diversas, como em imagens aéreas e de satélites, processamento de imagens médicas, compressão de imagem e vídeo, reconhecimento de ação, e extração de biometrias [4].

Algoritmos de SR aplicados em FR podem ajudar na melhoria das taxas de reconhecimento dos sistemas de vigilância, de forma a reduzir custos e aprimorar a segurança em estabelecimentos. Dentre as técnicas atuais de SR, a utilização de *Deep Learning* é uma das mais promissoras, sendo que dessas, redes GAN são atualmente o *State of the Art* (SOTA) (em português, Estado da Arte).

Para o reconhecimento da face em imagens de baixa qualidade, uma das técnicas que possui os melhores resultados é o treinamento em conjunto das redes de SR e FR [5].

O presente trabalho propõe a utilização de redes generativas para aumento de resolução e melhoria na qualidade de imagens de faces, de forma a ser treinada conjuntamente com a rede de reconhecimento das mesmas, i. e., uma rede conjunta que realiza tanto o aumento da resolução quanto a identificação do rosto presente na imagem.

1.1 Objetivos geral e específicos

O objetivo principal deste estudo é avaliar os resultados obtidos no Reconhecimento Facial, por meio da melhoria na qualidade de imagem utilizando Super-Resolução em redes GAN com *Joint-Learn*. Afim de alcançar o objetivo geral proposto, foram definidos os seguintes objetivos específicos:

- Investigar, implementar e desenvolver a rede neural conjunta de GAN *Super-Resolution* (SR) com *Face Recognition* (FR);
- Testar diferentes hiperparâmetros no treinamento das redes;
- Analisar o resultado do desempenho do reconhecimento facial, comparando-o com outras redes SOTA em *datasets* largamente utilizados na literatura.

1.2 Estruturação do documento

O presente trabalho está organizado em 6 capítulos. Neste capítulo está apresentado a introdução sobre os assuntos a serem trabalhados, bem como a motivação para tal e os objetivos a serem alcançados.

No capítulo 2 são apresentados os conceitos que servirão como base para a pesquisa, sendo eles *Digital Image Processing*, *Neural Network* e *Deep Learning*, e *Generative Adversarial Networks*. Logo após, no capítulo 3, são expostos de forma mais aprofundada os assuntos *Super-Resolution*, *Face Recognition* e sua inter-relação, bem como trabalhos correlatos utilizando FR em baixa resolução e SR.

Os materiais e métodos utilizados são expostos no capítulo 4 e, na sequência, os experimentos são apresentados junto da discução dos resultados no capítulo 5. Por fim, no capítulo 6, são apresentados a conclusão do trabalho e os trabalhos futuros.

Capítulo 2

Machine Learning e Processamento Digital de Imagens

Neste capítulo serão abordados os assuntos introdutórios que servirão como base para o capítulo seguinte, sendo eles: *Machine Learning* (Seção 2.2), Redes Neurais Artificiais (Seção 2.3), *Deep Learning* (Seção 2.4), Processamento Digital de Imagem (Seção 2.1), Redes Neurais Convolucionais (Seção 2.4.2), e *Generative Adversarial Networks* (Seção 2.4.3).

2.1 Processamento Digital de Imagens

Uma imagem digital pode ser compreendida como um sinal, muitas vezes multidimensional, que contém informações sobre a estrutura física a ser retratada [6]. Esse sinal pode ser modelado matematicamente por uma função de duas dimensões $f(x, y)$, descrita na Equação 2.1, cuja amplitude de f é um escalar positivo na coordenada espacial (x, y), denominado de intensidade luminosa [7].

$$f(x, y) = i(x, y)r(x, y) \tag{2.1}$$

A componente $i(x, y)$ corresponde à iluminância, que pode ser traduzida como uma medida da quantidade de iluminação que incide na cena representada. Já a componente $r(x, y)$, chamada de reflectância, corresponde a quantidade de luz refletida pelas estruturas representadas, cujo valor varia entre 0 (absorção total) e 1 (reflexão total).

Para que a imagem seja digitalizada, necessita-se da discretização do sinal de entrada contínuo para o formato digital. Com isso, dois processos são realizados, chamados de amostragem e quantização. No primeiro, pontos de amostragem são selecionados e ordenados no plano, gerando assim a resolução espacial da imagem, representada por sua unidade de medida PPI (pixels por polegada, do inglês, *pixels per inch*). Pelo Teorema da Amostragem de Nyquist-Shannon, tem-se que se a função for amostrada em uma taxa de amostragem igual ou superior a duas vezes a frequência máxima do sinal, é possível recuperar a função original a partir das amostras sem perda de informação. Caso a função seja amostrada a uma taxa inferior, perde-se informação e, com isso, haverá distorções na imagem em forma de frequências adicionais, gerando assim artefatos conhecidos como *aliasing* [7].

A quantização realiza a tradução da intensidade luminosa da cena em um equivalente digital. Para tal, utiliza-se uma quantidade de bits b em potências de dois 2^b, geralmente em valor de 8 bits por pixel por canal (para vermelho, azul e verde, a exemplo de uma imagem colorida), mas outros valores podem ser encontrados em diferentes aplicações. Se uma quantidade baixa de bits for utilizada para a quantização, perde-se clareza na imagem, gerando falsos contornos, conforme exemplificado na Figura (2.1) [8].

O resultado da amostragem e da quantização é uma matriz (possivelmente multidimensional) de valores reais (Equação (2.2)), com m linhas e n colunas, onde cada elemento $a_{m,n}$ é chamado de pixel (do ingles, *picture element*) e representa uma intensidade luminosa na posição (x, y) [6].

(a) Quantização com 8 bits, valores podem variar entre 0 e 32

(b) Quantização com 4 bits, valores podem variar entre 0 e 15

(c) Quantização com 2 bits, valores podem variar entre 0 e 3

(d) Quantização binária, valores podem variar entre 0 e 1

Figura 2.1: Exemplos de diferentes quantidades de bits utilizadas na quantização. [9]

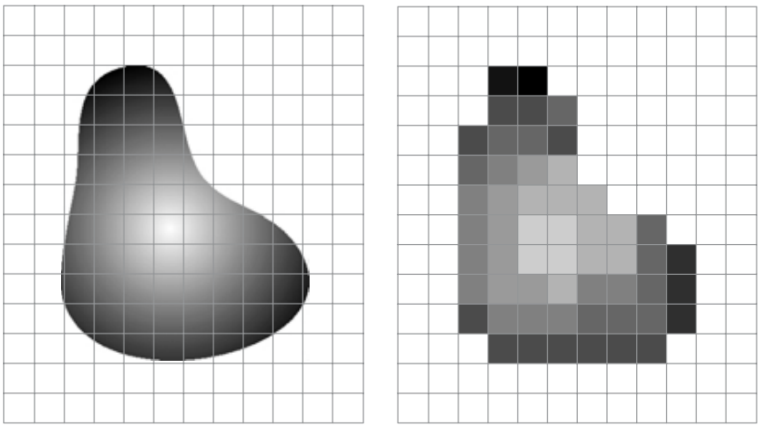
$$A_{m,n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix} \tag{2.2}$$

Em imagens monocromáticas cada entrada $a_{m,n}$ representa um valor na escala de cinza $l = f(x, y)$, que comumente está contido no intervalo $[0, l-1]$, onde 0 representa o preto, $l-1$ representa o branco e todos os valores intermediários são considerados tons de cinza variando entre o preto e o branco [7]. Já para imagens coloridas, cada entrada $a_{m,n}$ é

associada a um conjunto ordenado de valores $\{a_{m,n1}, a_{m,n2}, \ldots, a_{m,nk}\}$, gerando assim um tensor.

2.1.1 Baixa qualidade em imagens digitais

Uma das limitações na qualidade de uma imagem está relacionada a resolução espacial gerada pelo sensor de captura da imagem, como exemplificado na Figura (2.2). Tipicamente, os sensores podem ser de dois tipos, os CCD (do inglês, *charge-coupled device*) e os CMOS (do inglês, *complementary metal-oxide-semiconductor*), sendo ambos tipicamente arranjados em formato de matriz e que capturam imagens em duas dimensões. A densidade de pixels é determinada pela quantidade de elementos de sensor em uma determinada área, de forma que uma maior quantidade de elementos resulta em uma maior resolução espacial. O aumento do número de elementos de sensor, ou seja, maior quantidade por área mediante diminuição no tamanho dos mesmos, por vezes não é recomendado devido a menor incidência de luz por elemento de sensor por causa de seu tamanho reduzido, gerando assim ruídos na imagem final [10].



(a) Objeto real projetado no sensor de captura de imagem

(b) Imagem digital gerada após os processos de amostragem e quantização

Figura 2.2: Imagem capturada com sensor de baixa resolução. [7]

Mesmo em imagens de alta resolução, se existir uma grande distância entre a câmera e os objetos, a região de interesse - para o caso deste trabalho, faces de indivíduos - terá

baixa resolução, pela maior dificuldade de captar esses pequenos detalhes, vide Figura (2.3).

(a) Imagens originais em alta resolução

(b) Regiões de interesse em baixa resolução

Figura 2.3: Exemplos de regiões de interesse nativamente em baixa resolução extraídas de imagens de alta resolução. [5]

Outras limitações, ilustradas na Figura (2.4), estão associadas a aberrações de imagem, como efeitos de borramento que podem aparecer devido as lentes utilizadas na captura, *blur* ótico por movimentação, ou mesmo por limitações de custo, como as câmeras de vigilância que são limitadas pela velocidade de hardware, de rede, e pelo armazenamento [10].

O efeito de borramento, também conhecido como *blur*, é causado pela perda de frequências altas numa imagem, tirando a imagem de foco. Esse efeito pode ser causado também por movimentação do objeto a ser retratado ou da câmera, quando a velocidade de captura é mais lenta que a movimentação. No domínio do espaço, o *blur* faz com que os pixels vizinhos convirjam para um mesmo valor, deixando os contornos da imagem menos definidos [11].

Todos esses artefatos de imagem e a possível baixa resolução fazem com que as mesmas possuam baixa qualidade, interferindo negativamente na percepção por humanos e máquinas, principalmente em tarefas como reconhecimento de placas de veículos ou rostos em câmeras de segurança.

(a) Imagem original

(b) Imagem de baixa resolução

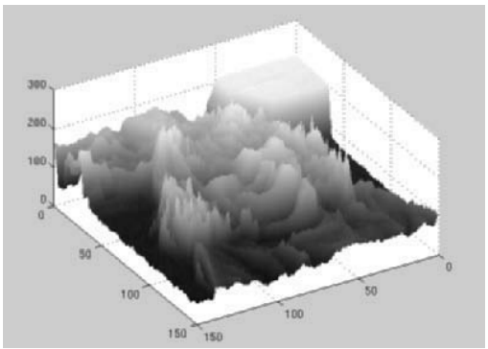
(c) Imagem com efeito Blur

(d) Imagem com efeito Blur gerado por movimentação

(e) Imagem com ruídos

Figura 2.4: Alguns exemplos de imagens de baixa qualidade. [8]

(a) Imagem digital no domínio do espaço



(b) Imagem digital no domínio da frequência

Figura 2.5: Uma mesma imagem em duas representações, a representação no domínio da frequência e sua contrapartida no domínio espacial. [8]

Para contornar essa problemática de imagens de baixa qualidade, tem-se um conjunto de técnicas processamento de imagem, que incluem a Super-Resolução, e que podem ser

divididas entre técnicas do domínio espacial e técnicas do domínio da frequência (Figura (2.5)). A primeira corresponde a imagem digital propriamente dita, manipulando os pixels que compõem a imagem. Já a última corresponde ao conjunto de técnicas para manipular as frequências que geram a determinada imagem, através da modificação da correspondente Transformada de Fourier [7].

2.1.2 Região de Interesse

A segmentação é uma forma de se trabalhar apenas com as regiões da imagem que tem importância para a resolução de determinado problema. Segundo [12], a segmentação de imagens possui aplicações diversas, como na área médica (localização de tumores e patologias, cirurgias assistidas por computador, estudo de estruturas anatômicas), localização de objetos em imagens de satélite, reconhecimento de digitais, reconhecimento facial, dentre outras.

Existem dois princípios para determinar a região de interesse, o princípio da descontinuidade e o princípio da similaridade. A similaridade realiza o agrupamento de pixels que possuam propriedades parecidas, como intensidade, cor, textura, dentre outras. Já a descontinuidade busca extrair regiões baseando-se em diferenças abruptas em intensidade, como bordas [7].

Uma borda é definida como um conjunto de pixels conectados em uma fronteira entre duas regiões que diferem valores na escala de cinza. De forma a encontrar essas bordas, utiliza-se a primeira e a segunda derivadas da função imagem. Pode-se classificar uma borda nos seguintes tipos, ilustrados na Figura (2.6) [12]:

- ***Step Edge***: nesse tipo existe uma mudança abrupta na intensidade dos pixels. Pode-se também encontrar esse tipo de borda como *Spike Edge*, em que existe uma mudança abrupta na intensidade dos pixels, seguida por um retorno imediato aos valores originais;

- ***Ramp Edge***: apresenta uma mudança gradual na intensidade dos pixels;

- ***Roof Edge***: nesse tipo a borda não é instantânea em uma curta distância, ela passa dos valores iniciais para valores intermediários, depois uma fina linha de valor final da borda, depois retorna aos valores intermediários, terminando nos valores da nova região.

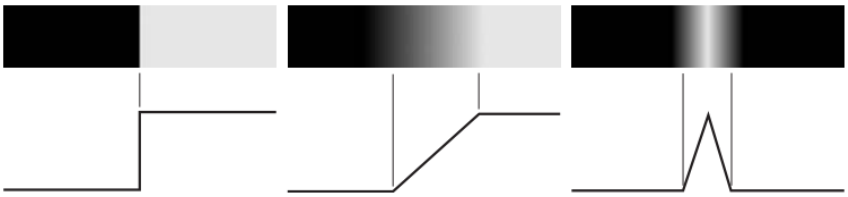


Figura 2.6: Exemplo de bordas em uma imagem. Em sequência: *Step Edge, Ramp Egde, e Roof Edge*. [7]

Segundo [7], a segmentação faz parte do processo intermediário do processamento digital de imagens, sendo essa uma etapa fundamental para o reconhecimento e classificação de objetos.

2.2 *Machine Learning*

A *Artificial Intelligence* (AI) (em português, Inteligência Artificial), como área do conhecimento, estuda os princípios do comportamento inteligente e a criação de agentes capazes de agir desse modo [13]. Buscando inspiração em outras áreas do conhecimento, dentre elas Filosofia, Matemática, Psicologia e Neurociência [14], a AI inspira-se muitas vezes na inteligência humana, na ideia de ação racional para desenvolver novos conceitos e técnicas.

Todo sistema que percebe o ambiente através de sensores e age por meio de atuadores transformando o meio onde se encontra, é denominado de agente. Partindo dessa premissa, um agente inteligente entende o ambiente onde está inserido e toma decisões apropriadas levando em conta as circunstâncias e objetivos a serem atingidos, idealmente realizando a melhor ação possível [14].

Parte dos agente ditos inteligentes possui a capacidade de aprender, ou seja, melhorar seu desempenho em determinada tarefa baseando-se na experiência [15]. A esse campo

de estudos denomina-se de *Machine Learning* (ML), onde o agente inteligente, utilizando processo lógico de indução, elabora conclusões genéricas a partir de um conjunto de dados (*dataset*) do problema específico [16].

Para que tal aprendizado ocorra, o agente pode receber alguma interação em seu treinamento por parte do projetista, direcionando esse aprendizado ao objetivo a ser alcançado. Segundo Russel e Norvig [14], essa interação ocorre em três tipos de aprendizado:

- **Aprendizado Supervisionado**: utilizado em problemas nos quais é necessário classificar instâncias, por exemplo, em um agente para diferenciar pacientes diabéticos de saudáveis. Consiste em apresentar ao algoritmo um *dataset* rotulado, contendo as informações necessárias do problema em questão e as suas corretas classificações, de forma que o agente crie uma função que mapeia as características do problema em respostas esperadas, e quando alimentado com novos exemplos, isto é, não vistos anteriormente, classifique-os de forma correta;

- **Aprendizado Não-Supervisionado**: onde nenhuma interação extra é feita com agente, sendo esse responsável, por exemplo, no caso do agrupamento, por identificar padrões nos dados de entrada não rotulados e agrupá-los. Um exemplo de tarefas de agrupamento consiste em, dado um *dataset* de itens vendidos em uma loja, identificar os diferentes perfis de consumidores;

- **Aprendizado por Reforço**: o agente aprende por meio do reforço de diferentes estímulos, traduzidos em uma pontuação, sendo esse estímulo positivo (recompensando pelo acerto), ou negativo (punindo pelo erro). Como o objetivo do algoritmo é maximizar a pontuação o problema a ser solucionado passa a ser do domínio da otimização.

Algumas aplicações de ML, segundo Faceli et al. [16], são:

- Reconhecimento de palavras faladas;

- Predição de taxas de cura de pacientes com diferentes doenças;

- Detecção de uso fraudulento de cartões de crédito;
- Condução de automóveis de forma autônoma em rodovias;
- Ferramentas que jogam gamão e xadrez de forma semelhante a campeões;
- Diagnóstico de câncer por meio da análise de dados de expressão gênica.

Para resolver alguns dos problemas citados, os algoritmos ditos clássicos de ML são suficientes. Já para problemas mais complexos, como condução autônoma de veículos, necessitam de algoritmos mais complexos, com maior poder computacional, sendo esses *Neural Networks*, ou até mesmo *Deep Learning* (Seção 2.4).

2.3 *Neural Networks*

O sistema nervoso humano é formado, em linhas gerais, por uma rede de neurônios que comunicam entre si. Sendo o neurônio a unidade principal da rede, essa célula altamente excitável é responsável por receber, processar e enviar informações, utilizando de impulsos elétricos para tal (Figura (2.7)).

Segundo Machado e Haertel [17], a maioria dos neurônios possui, basicamente, três estruturas com funções específicas:

- **Corpo celular**: centro metabólico do neurônio, responsável pela síntese de todas as proteínas neuronais. Junto com os dendritos, recebe os estímulos e processa essa informação, decidindo pela excitação ou inibição dos neurônios ligados a este;
- **Dendritos**: especializados em receber estímulos, traduzindo-os em alterações do potencial de repouso da membrana, e transmitir esses impulsos para o corpo celular ou axônio;
- **Axônio**: ligado ao corpo celular, comunica-se com outros neurônios pela sinapse e é especializado em gerar e conduzir impulsos nervosos do tipo “tudo ou nada”, ou seja, ou estimula completamente o neurônio seguinte, ou o inibe.

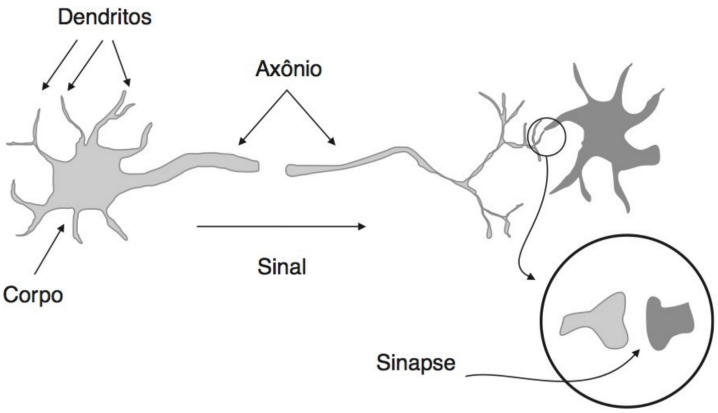


Figura 2.7: Representação de um neurônio biológico. [16]

Como o corpo humano tem um número altíssimo de neurônios interconectados, de 10 a 500 bilhões divididos em diversos módulos e redes, o cérebro humano tem massiva capacidade de paralelização, que promove grande velocidade no processamento de informações [16].

2.3.1 Perceptron

McCulloch e Pitts [18] propuseram um modelo matemático para o sistema nervoso humano. A partir desse trabalho inicial, desenvolveram-se as *Neural Network* (NN) (em português, Redes Neurais Artificiais) com o objetivo inicial desenvolver sistemas de aprendizado biologicamente plausíveis.

Inspirado no neurônio biológico, o Perceptron k (Figura (2.8)) recebe, de forma análoga a uma sinapse, uma combinação linear entre um vetor de entradas $\vec{x} = [x_1, x_2, x_3, \ldots, x_j]^T$ e um vetor de pesos $\vec{w_k} = [w_1, w_2, w_3, \ldots, w_j]$ (que determina a força e o sinal da conexão com o neurônio), chamado de potencial de ativação u_k, conforme Equação 2.3 [16].

$$u_k = \sum_{j}^{m} x_j w_{k,j} \tag{2.3}$$

A essa entrada é acrescido um *bias* b_k, que representa o quão propenso o neurônio é a disparar. Depois, é aplicada a função de ativação $\varphi(\cdot)$ (Seção 2.3.2), determinando assim a saída y_k, conforme as Equações 2.4 e 2.5 [19]. O Perceptron então é treinado variando

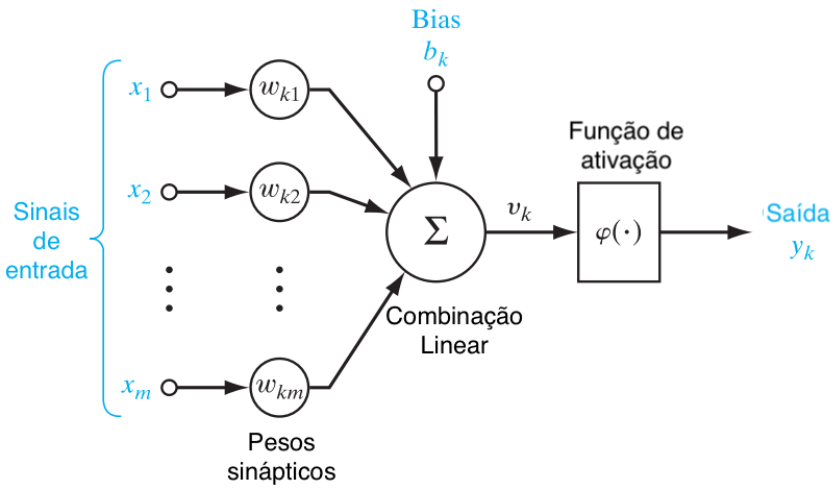


Figura 2.8: Representação de um neurônio artificial. [19]

o vetor de pesos e o *bias*, de forma a alterar sua saída.

$$v_k = u_k + b_k \tag{2.4}$$

$$y_k = \varphi(v_k) \tag{2.5}$$

2.3.2 Função de Ativação

A função de ativação $\varphi(\cdot)$ determina a força com a qual o neurônio dispara. Tem característica de ser uma função não-linear, aumentando assim a gama de problemas possíveis de serem resolvidos pelas NNs [20]. Além disso, realiza a normalização da saída do neurônio, restringindo os valores de saída entre 0 e 1, mas por vezes também restringindo a valores entre -1 a $+1$, a depender da característica necessária para resolver determinado problema.

A primeira função utilizada é conhecida como Função Passo, ou *Threshold Function*, expressa pela Equação 2.6, cujo gráfico é representado pela Figura (2.9). Sua característica é resultar em 1 se a entrada for não negativa, noutro caso 0.

$$\varphi_1(v_k) = \begin{cases} 1 & \text{se } v_k \geq 0 \\ 0 & \text{se } v_k < 0 \end{cases} \tag{2.6}$$

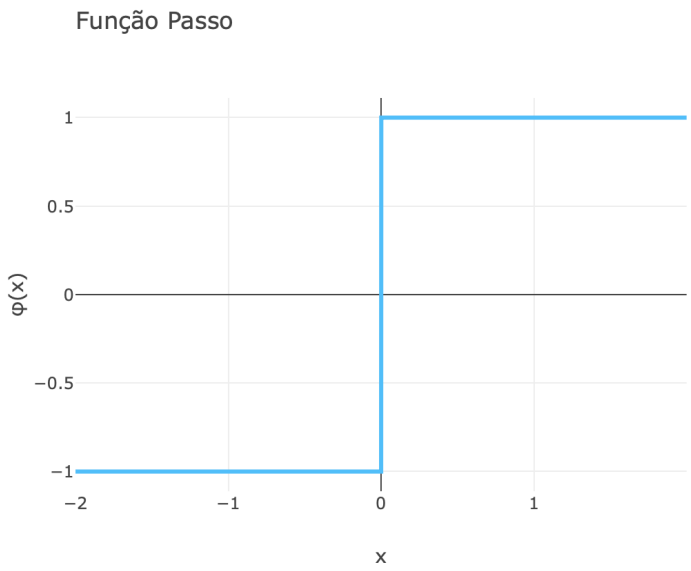


Figura 2.9: Gráfico da Função Passo.

Essa função impõe uma limitação ao neurônio artificial, mimetizando a necessidade de um pulso elétrico forte o suficiente para o neurônio biológico ser ativado, expressando assim a propriedade "tudo ou nada" do modelo neuronal proposto por McCulloch-Pitts.

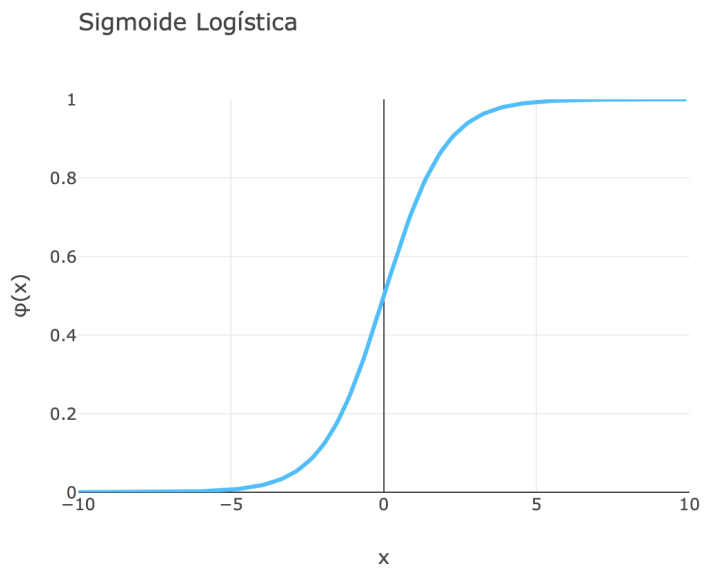
Historicamente, a função que foi mais utilizada é a Função Sigmoide Logística, descrita na Equação (2.7) e cujo gráfico é representado pela Figura (2.10a), pois a mesma é diferenciável e possui uma derivada fácil de ser calculada, sendo essa $\varphi_2(v_k)' = \varphi_2(v_k)[1-\varphi_2(v_k)]$ [19].

$$\varphi_2(v_k) = \frac{1}{1+e^{-av_k}} \tag{2.7}$$

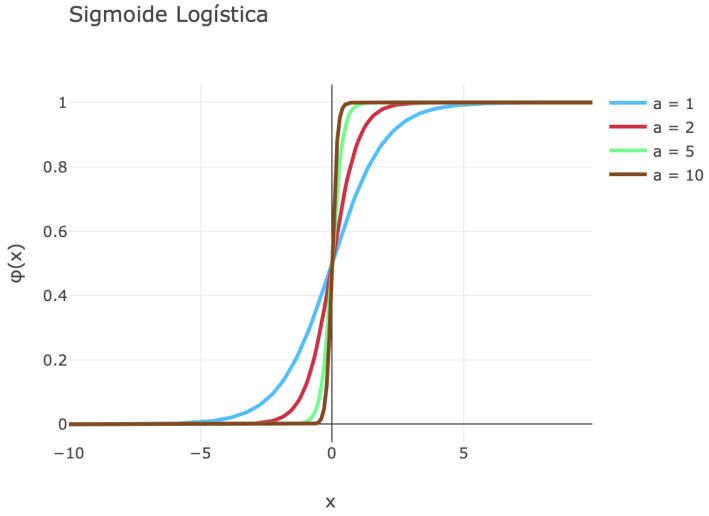
Diferentemente da Função Passo, a Sigmoide assume valores contínuos entre 0 e 1, sendo que, no limite, quando v_k é um número muito positivo, $e^{-av_k} \approx 0$ e, por consequência, $\varphi(v_k) \approx 1$. De forma análoga, quando v_k possui valores muito negativos, $e^{-av_k} \to \text{inf}$ e $\varphi(v_k) \approx 0$ [21].

O parâmetro a, por sua vez, indica a inclinação da curva, como ilustrado na Figura (2.10b).

No limite, conforme $a \to \infty$, a Sigmoide se aproxima do comportamento da Função Passo original [19].



(a) Sigmoide Logística com parâmetro $a = 1$



(b) Sigmoide Logística com variações no parâmetro a

Figura 2.10: Gráfico da Função Sigmoide Logística.

Uma variação também utilizada da Função Sigmoide é a Tangente Hiperbólica (Equação (2.8) e Figura (2.11)), utilizada quando é desejável que a saída do neurônio esteja no intervalo entre -1 e $+1$.

$$\varphi_3(v_k) = \tanh(v_k) \tag{2.8}$$

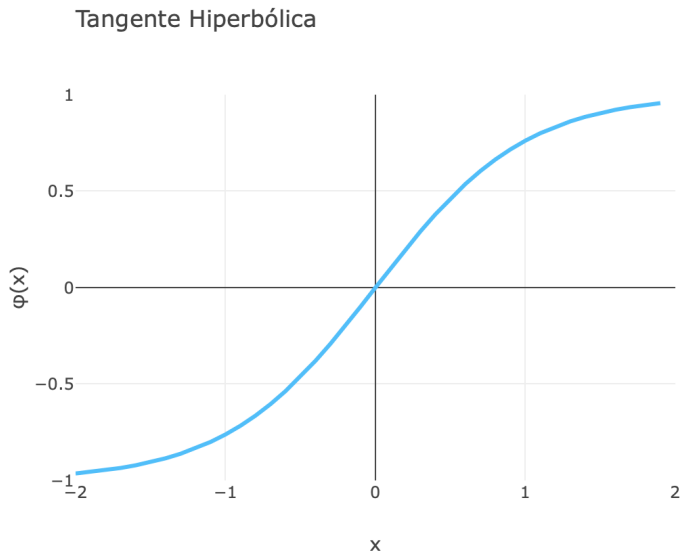


Figura 2.11: Gráfico da Função Tangente Hiperbólica.

2.3.3 *Multilayer Perceptron*

*Neural Network*s são aproximadores universais de funções [22]. A exemplo de um classificador, uma *Neural Network* (NN) mapeia as entradas $\vec{x}$ para a categoria y [20]. O Perceptron, por ser uma rede neural composta somente de um neurônio, apenas consegue classificar problemas linearmente separáveis, pois realiza a separação entre os valores das entradas passando um hiperplano pelas mesmas. Ademais, é limitado a apenas problemas de classificação binários, isto é, com apenas duas classes de saída [19].

De forma a aumentar a gama de problemas de possível resolução pelas NNs, as redes *Multilayer Perceptron* (MLP) são um conjunto de Perceptrons conectados e organizados em camadas (Figura (2.12)), sendo elas [19]:

- **Camada de entrada**: vetor $\vec{x}$ com as entradas da rede, que são numéricas, e que serão propagadas para zero ou mais camadas ocultas;
- **Camada oculta**: os neurônios dessa camada recebem as entradas e mapeiam suas saídas possivelmente para outra camada oculta, ou então para a camada de saída. Esta camada é chamada de "oculta", pois os neurônios presentes nela estão ocultos tanto para a entrada quanto para a saída da rede;

- **Camada de saída**: camada final da rede, com as saídas calculadas.

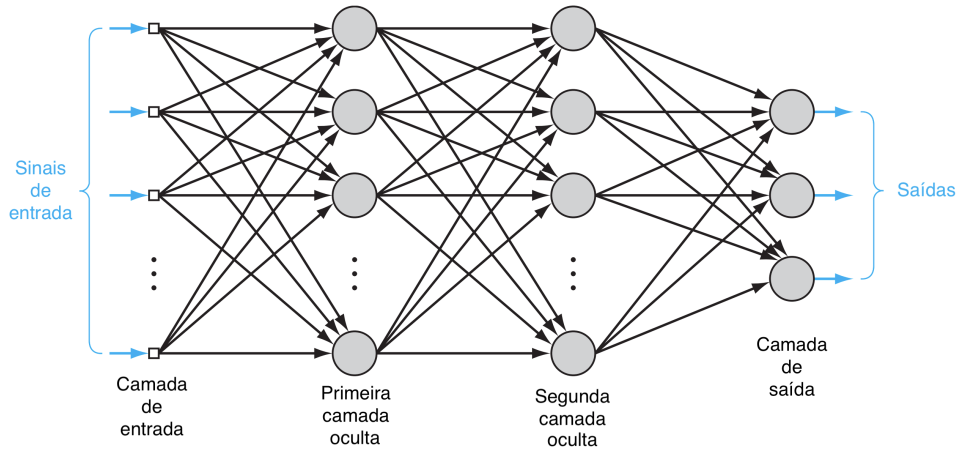
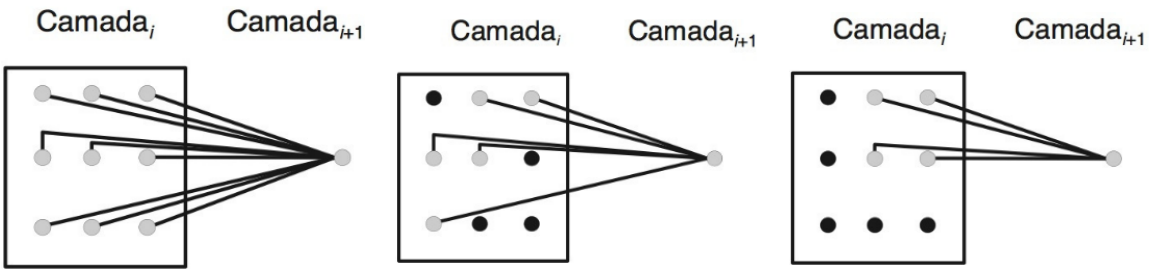


Figura 2.12: Grafo de uma rede *Feedforward Multilayer Perceptron* com duas camadas ocultas. [19]

Os neurônios da camada oculta realizam a extração de características, identificando quais são as características mais relevantes nos dados de treinamento. Essa extração é realizada por meio de transformações não-lineares, mapeando os dados de entrada para um novo espaço chamado de espaço de características (do inglês, *feature space*) [19]. Essa característica não-linear faz com que as NNs sejam muito mais poderosas que o Perceptron original, podendo ser aplicadas numa maior gama de problemas.

A arquitetura de rede mais clássica é chamada de *Feedforward*, onde o sinal é propagado de camada em camada, partindo da camada de entrada até a camada de saída [19]. De acordo com Faceli et al. [16], as redes podem apresentar diferentes padrões de conexão entre os neurônios (Figura (2.13)), podendo ser classificadas em:

- **Totalmente conectada**: quando os neurônios de uma camada estão conectados a todos os neurônios da camada seguinte;
- **Parcialmente conectada**: quando os neurônios de uma camada estão conectados a apenas alguns neurônios da camada seguinte;
- **Localmente conectada**: uma variação das redes parcialmente conectadas, onde as conexões entre neurônios estão em uma região bem definida.



(a) Totalmente conectada (b) Parcialmente conectada (c) Localmente conectada

Figura 2.13: Diferentes padrões de conexão presentes em NNs. [16]

Para que o aprendizado ocorra, a rede é inicializada com o vetor de pesos $\vec{w}$ e os *bias* b_k recebendo valores aleatórios e, após o algoritmo de aprendizado iterar sobre os exemplos de treinamento, os pesos da rede são modificados em caso de classificação incorreta do exemplo. Geralmente os *bias* são inicializados correspondendo a uma nova entrada no vetor de pesos $w_{0,k} = b_k$ e, como consequência, uma nova entrada ao vetor de entradas $y_0 = +1$.

A alteração dos pesos ocorre segundo a Equação (2.9), onde $\Delta\vec{w}$ (descrito na Equação (2.10)) corresponde a variação entre o valor obtido e o valor esperado $\vec{d}$, isto é, o erro da rede [15]. Considera-se que é utilizada a função de custo MSE (Equação (2.11)).

$$\vec{w_k} \leftarrow \vec{w_k} + \Delta\vec{w_k} \tag{2.9}$$

$$\Delta\vec{w_k} = \eta(\vec{d} - \vec{y})\vec{x} \tag{2.10}$$

Representada por η, a *learning rate* (em português, Taxa de Aprendizado) é uma constante positiva e de valor pequeno, que tem como objetivo regular quão grande será a mudança no valor dos pesos a cada iteração, e, por consequência, a velocidade com que a rede irá convergir.

A Equação (2.9) tem convergência garantida para casos linearmente separáveis, mas não para os casos não linearmente separáveis [15]. Além disso, uma nova forma de correção dos pesos se faz necessária, chamada de *Backpropagation*, a qual faz uso do aprendizado

baseado em gradiente.

2.3.4 Gradiente Descendente

O aprendizado baseado em gradiente advém do fato da minimização de uma função suave e contínua ser mais fácil de ser realizada comparado a funções combinatórias e discretas [23]. Para tal, é utilizada uma Função Custo $C(\cdot)$, que leva em conta os pesos atuais e a magnitude de seus erros comparado ao valor esperado. Com isso, tem-se um problema de otimização em que se minimiza a Função Custo que, por consequência, altera os pesos da rede de forma a minimizar o erro.

Do conjunto de possíveis funções, uma função que foi muito utilizada em problemas de regressão é o *Mean Squared Error* (MSE) (em português, Erro Quadrático Médio), que está descrito na Equação (2.11) [19].

$$C(\vec{w_k}) \equiv \frac{1}{2} \sum_{l=0}^{j} (d_l - y_l)^2 \tag{2.11}$$

A equação descrita está adaptada para o modo de treinamento *On-Line*, mas também pode ser adaptada para os outros modos de treinamento. Diferentes modos de treinamento serão discutidos mais adiante.

Por meio do uso do cálculo, pode-se derivar parcialmente $C(W)$ com relação a cada componente da matriz de pesos W, gerando assim o vetor gradiente $\nabla C(W)$. Esse vetor gradiente, cuja equação está expressa na Equação (2.12), aponta para a região de crescimento da função.

$$\nabla C(W) \equiv \left[\frac{\partial C(W)}{\partial w_0}, \frac{\partial C(W)}{\partial w_1}, \ldots, \frac{\partial C(W)}{\partial w_j} \right]^T \tag{2.12}$$

Como o gradiente da função indica a direção de crescimento da mesma, e a otimização desejada é a minimização de $C(W)$, então a negativa desse vetor indica a direção de descida da encosta, ou seja, da região de mínimo local da função [15]. Modificando a

Equação (2.10), tem-se a nova função ΔW, expressa por:

$$\Delta W = -\eta \nabla C(W) \tag{2.13}$$

E assim, pode-se então modificar a Equação (2.9), resultando na Equação 2.14:

$$W \leftarrow W - \eta \nabla C(W) \tag{2.14}$$

Computar o gradiente após iterar sobre todos os exemplos de treinamento pode ser extremamente custoso. Para tal, uma derivação desse aprendizado, denominado de *Stochastic Gradient Descent* (SGD) (em português, Gradiente Descendente Estocástico), computa o gradiente em pequenos lotes de exemplos aleatórios, sendo esse subconjunto do conjunto de dados de treinamento. Quando o SGD computa em lotes de tamanho 1, tem-se o Aprendizado On-line. A ideia parte do princípio que o gradiente é uma indicação da direção de crescimento da função e que pode ser aproximada utilizando um pequeno grupo de exemplos [20].

O algoritmo de SGD é utilizado no aprendizado por *Backpropagation*. Existem algumas variações e alternativas, como SGD com *Momentum*, *Stochastic Normalized Gradient Descent* (SNGD), Adam, Adagrad, NovoGrad, dentre outros, que serão explorados mais adiante neste capítulo.

2.3.5 Backpropagation

O erro cometido pela rede é propagado de camada em camada até a camada de saída. Para que os pesos sejam alterados, baseando-se na influência de cada neurônio no erro total, o algoritmo de *Backpropagation* atua em duas fases [16].

Na fase *Forward* são apresentados à rede os dados de treinamento, que cada neurônio da camada de entrada usará como valor de entrada para gerar sua saída que servirá de entrada para a próxima camada de neurônios e assim, sucessivamente, até a camada de saída que terá seu erro δ_k calculado. O ajuste de pesos é realizado na fase *Backward*

partindo da camada de saída até a primeira camada oculta, como mostra a Equação (2.15) [16]. Nota-se que o ajuste de peso na camada de saída é diferente do ajuste de pesos das camadas ocultas.

$$\delta_k = \begin{cases} \varphi' C(\cdot) & \text{se } n_k \in N_{\text{saída}} \\ \varphi' \sum w_{j,k} \delta_{k-1} & \text{se } n_k \in N_{\text{oculta}} \end{cases} \tag{2.15}$$

Para a camada de saída, o valor do erro é conhecido, então aplica-se diretamente a correção necessária nos pesos de cada neurônio. Nas camadas ocultas, o valor do erro precisa ser estimado, então leva-se em conta o erro cometido por cada neurônio da camada seguinte que está conectado ao neurônio k em questão.

Dessa forma, iterativamente, os pesos da rede são modificados para o problema em questão e, consequentemente, a rede consegue aprender.

Um dos problemas relacionados ao *Backpropagation* ocorre quando usado em conjunto com funções do tipo Sigmoide, como a Sigmoide Logística (Equação (2.7)) ou Tangente Hiperbólica (Equação (2.8)), pois o gráfico da função se torna quase plano para potenciais de ativação muito positivos ou negativos. Com isso, $\varphi'(v_k) \approx 0$, e então diz-se que a função satura, situação em que aprendizado do peso em questão fica estagnado ou muito lento. Dessa maneira, outras funções de ativação são preferíveis, como a ReLU (Equação 2.16) [21].

2.4 *Deep Learning*

A criação do algoritmo de *Backpropagation* possibilitou o aumento no número de camadas das redes neurais. Essa possibilidade ocorre devido ao fato de que os erros dos neurônios das camadas iniciais, que são propagados até camadas muito profundas da rede, conseguem ser retornados com boa aproximação de valor até as primeiras camadas, propiciando assim o aprendizado da rede.

O termo *Deep Learning* (DL) (em português, Aprendizado Profundo) advém do fato de que as redes utilizadas possuem muitas camadas, tornando-se assim profundas. As

NNs são aproximadores universais de funções e apenas uma camada oculta é suficiente para representar qualquer função, mas essa camada possivelmente terá uma vasta quantidade de neurônios, fazendo com que a rede possa não aprender e generalizar a função corretamente. Pela adição de novas camadas, a rede se torna mais apta a aproximar funções de forma mais eficiente (com menos neurônios) [20]. Para o caso das *Convolutional Neural Networks* (Seção 2.4.2), uma maior quantidade de camadas traz benefícios na extração de características de maior complexidade, resultando em melhores taxas de acerto da rede, conforme demonstrado por [24] e [25]. *Deep Networks* possibilitam também a criação de diferentes arquiteturas, como as já citadas CNN, as Redes Recorrentes e as Redes Generativas [19].

Para esse tipo de rede utiliza-se uma família de funções de ativação, sendo a mais conhecida a função *Rectified Linear Unit* (ReLU), por não sofrer do problema de saturação do neurônio [21] para valores positivos. Como nessa função a saturação ainda ocorre em valores negativos, existem algumas variantes como Elu, PReLu, Softplus, dentre outras. Adicionalmente, segundo [19], a função ReLU converge seis vezes mais rápido que a função Tangente Hiperbólica (Equação (2.8)) e possui menor custo computacional.

Conforme descrito na Equação (2.16) (Figura (2.14)), a função retorna o maior valor entre 0 e o valor de entrada, ou seja, transforma as entradas negativas em 0.

$$\varphi_4(v_k) = \max(0, v_k) \tag{2.16}$$

Outra opção de função de ativação é a denominada Mish [26], que possui características similares a ReLU, sendo essa também limitada inferiormente e ilimitada superiormente, onde, para o caso da função Mish, seus valores se encontram no intevalo $[\approx -0.31, \inf)$. Ser ilimitada superiormente é uma propriedade desejável pois evita saturação, o que causa o aumento no tempo de treinamento. Por outro lado, ser limitada inferiormente traz a vantagem de aumentar o poder de regularização da função [26].

De outra forma, Mish se difere da função ReLU por ser suave, e não-monotônica (conforme pode ser visto nas Figuras (2.15) e (2.16)). Com isso, os valores conseguem ser

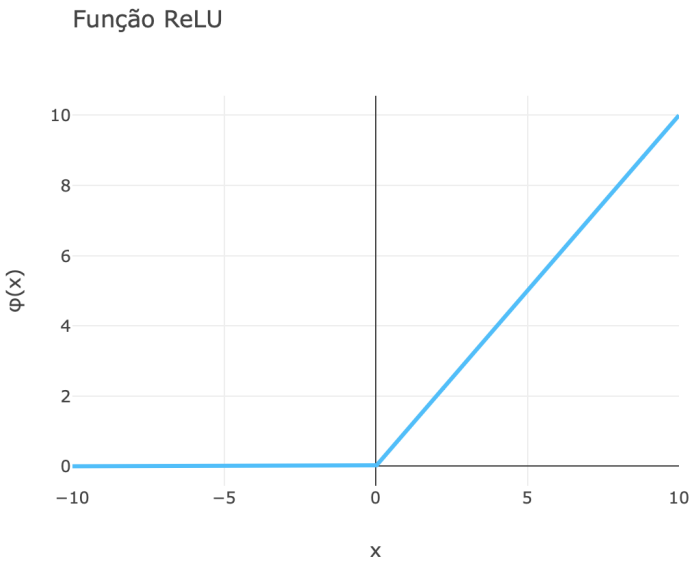


Figura 2.14: Gráfico da Função ReLU.

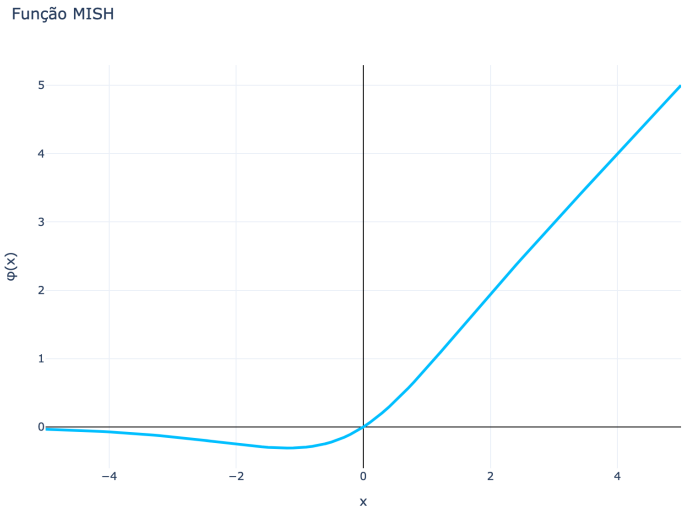
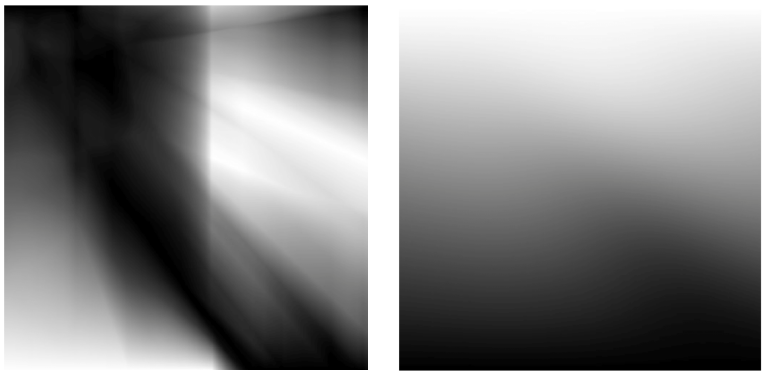


Figura 2.15: Gráfico da Função Mish.

melhor propagados pela rede tanto na fase *forward* quanto na fase *backward* do treinamento, principalmente em redes muito profundas, aumentando o poder de generalização e a acurácia das mesmas [26], [27]. Em comparação a ReLU, a Mish promoveu um aumento de acurácia de 1.671% no dataset CIFAR 100 [26].

A função Mish tem sua Equação descrita em (2.17).



(a) (b)

Figura 2.16: Comparação entre o *output landscape* das funções ReLU (a) e Mish (b). [26]

$$\varphi_5(v_k) = x\tanh(softplus(v_k)) = x\tanh(\ln(1+e^{v_k})) \tag{2.17}$$

Para realizar o treinamento utilizando *Backpropagation*, as redes profundas necessitam de uma maior quantidade de dados e de maior poder computacional devido ao número elevado de parâmetros [21].

ReLU é a mais largamente utilizada dentre as funções de ativação pela sua simplicidade na implementação, performance consistente e baixo custo computacional. Em comparação com a anterior, a Mish é mais custosa computacionalmente, aumentando em 1 segundo por época em comparação a ReLU em uma GPU Nvidia Tesla V100 [27]. Mesmo com esse pequeno aumento no custo computacional, a Mish se sobressai nos treinamentos em redes muito profundas em comparação com as outras possíveis funções de ativação pelas suas características anteriormente descritas, sendo considerada *State of the Art* (SOTA) (em português, Estado da Arte) atualmente (Outubro de 2020).

Outro obstáculo para esse conjunto de redes é denominado problema do *Vanishing Gradient* (em português, Dissipação do Gradiente), onde as camadas mais próximas a saída aprendem de maneira mais veloz que as camadas mais próximas a entrada, pois o gradiente do erro encolhe exponencialmente para valores muito pequenos, dificultando assim o aprendizado e afetando de forma mais contundente as Redes Recorrentes. Já as Redes *Feedforward* tem seu gradiente escalado pela raiz quadrada da profundidade da

rede, e esse problema pode ser mitigado aumentando a largura de cada camada, como demonstrado por [28].

Em contrapartida, o *Exploding Gradient* faz o oposto do problema anterior, em que seu gradiente do erro explode exponencialmente para valores muito grandes, tendendo a $+\infty$ ou $-\infty$, quebrando assim o treinamento. Para esses casos, a técnica de *Gradient Clipping* pode ser utilizada, em que se limita o valor máximo que o gradiente pode assumir, fazendo com que o mesmo não exploda [9].

2.4.1 Outros otimizadores e NovoGrad

Conforme descrito anteriormente, *Stochastic Gradient Descent* é o mais clássico otimizador para NNs, mas algumas variações e alternativas existem. Dentre essas opções, o NovoGrad [29] faz parte da família de otimizadores *Stochastic Normalized Gradient Descent* (SNGD) [30]. Esses otimizadores utilizam apenas a direção do vetor gradiente para fazer a atualização da matriz de pesos W, ignorando a magnitude desse vetor (Equação (2.18)) [30]. Com isso, segundo Ginsburg et al. [29], Hazan et al. [30] provaram que o SNGD é robusto aos problemas de *Vanishing* e *Exploding Gradient*, ainda assim tendo convergência garantida, mas sendo instável no início do treinamento.

$$W_{t+1} = W_t - \lambda_t * \frac{g_t}{\parallel g_t \parallel} \tag{2.18}$$

Onde $g := \nabla C(W)$.

De forma a aumentar a estabilidade do algoritmo SNGD, pode-se realizar a média dos gradientes. Para tal, o otimizador Adam [31] realiza as médias móveis m_t e v_t, conforme descrito na Equação (2.19), e posteriormente atualiza os pesos da rede conforme a Equação (2.20).

$$\begin{aligned} m_t &= \beta_1 * m_{t-1} + (1 - \beta_1) * g \\ v_t &= \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \end{aligned} \tag{2.19}$$

$$W_t = W_t - \lambda_t * \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{2.20}$$

Onde $\epsilon \ll 1$ é um parâmetro adicionado para garantir estabilidade númerica. β_1 e β_2 são coeficientes de valores $[0, 1)$.

O NovoGrad parte dessa premissa, combinando também as técnicas de normalização dos gradientes por camada [32] ($\hat{g}_t^l = \frac{g_t^l}{\|g_t^l\|}$, onde g_t^l diz respeito ao gradiente para a camada l na iteração t) para melhoria de performance e generalização no treinamento, utilizando-a para computar v_t, e de desacoplamento de *weight decay* d [33] $W_t = W_t - \lambda_t * (\frac{m_t}{\sqrt{v_t}+\epsilon} + d * W_t)$ também buscando melhora no poder de generalização da função.

Para tal, o NovoGrad calcula o segundo momento da média móvel v_t^l como descrito na Equação 2.21 [29].

$$v_t^l = \beta_2 * v_{t-1}^l + (1 - \beta_2) * \parallel g_t^l \parallel^2 \tag{2.21}$$

Após, calcula o primeiro momento m_t^l, utilizando v_t^l para normalizar o gradiente, e adicionando o *weight decay* desacoplado ao gradiente normalizado, como descrito na Equação (2.22) [33].

$$m_t^l = \beta_1 * m_{t-1}^l + (\frac{g_t^l}{\sqrt{v_t^l} + \epsilon} + d * W_t^l) \tag{2.22}$$

E, finalmente, atualiza os pesos da mesma maneira que o SGD, como descrito na Equação 2.23 [33].

$$W_{t+1}^l = W_t^l - \lambda_t * m_t^l \tag{2.23}$$

Com isso, geralmente o NovoGrad consegue performar melhor que os algoritmos SGD e Adam - requisitando metade da memória comparado ao primeiro -, é robusto às instabilidades do começo do treinamento, e consegue generalizar bem o suficiente para ser utilizado nos mais variados problemas, tais como classificação de imagens, reconhecimento de fala, tradução de texto, dentre outros [33].

2.4.2 *Convolutional Neural Network*

Inspirando-se em como os animais enxergam, essa arquitetura de rede é uma especialização do *Feedforward Multilayer Perceptron*, desenvolvida para reconhecer padrões 2D com alto grau de invariância a algumas transformações. Para tal, as *Convolutional Neural Network* (CNN) (em português, Redes Neurais Convolucionais) são redes parcialmente conectadas, e possuem três ideias arquiteturais: Campos Receptivos Locais, compartilhamento de pesos, e Pooling [23] (Figura (2.17)).

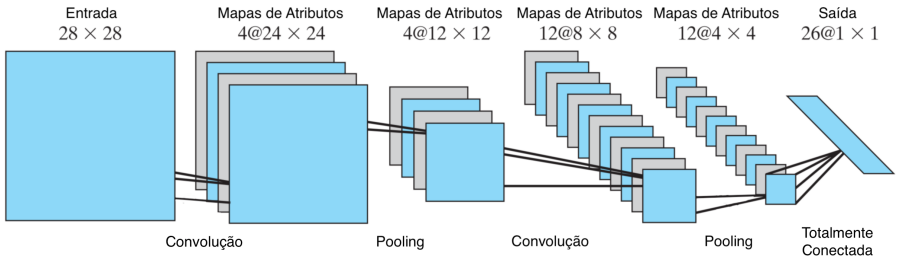


Figura 2.17: Exemplo de Rede Neural Convolucional com seis camadas. [19]

Campos Receptivos Locais

No processamento de imagens, é comum organizar-se os neurônios em matrizes ao invés de vetores. Essas matrizes, por sua vez, podem inclusive ser agrupadas em tensores de ordem superior.

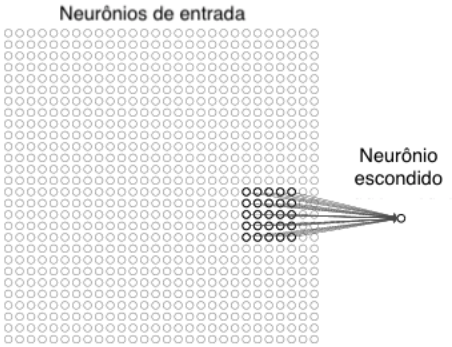


Figura 2.18: Criação de um neurônio escondido a partir de um Campo Receptivo Local de 5×5. [21]

Cada neurônio de uma camada recebe entradas de uma região bem definida da camada anterior, chamada de *Local Receptive Field* (em português, Campo Receptivo Local), vide Figura (2.18), fazendo com que características locais sejam extraídas e propagadas pela rede [23]. O campo receptivo local é geralmente uma região da matriz de entrada. Essa região tende a ser pequena e o conjunto de entradas pertencentes a ela está ligado a apenas um neurônio da camada seguinte, de modo que o conjunto de vários campos de uma camada n gera a camada $n + 1$ com uma quantidade de neurônios menor que da camada atual, conforme ilustra a Figura (2.19).

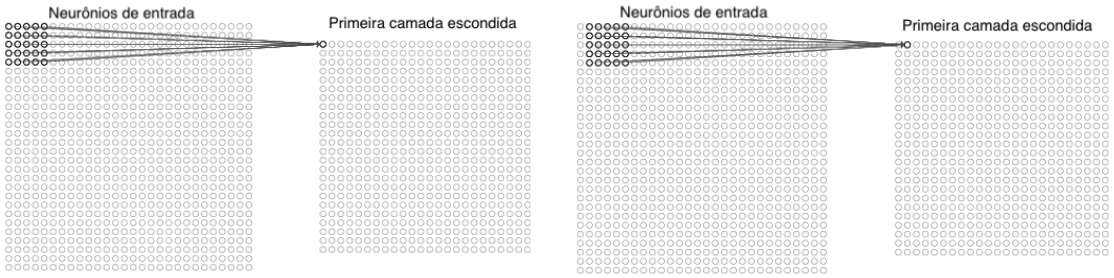


Figura 2.19: A exemplo de uma imagem de 28×28 pixels na camada de entrada, se o Campo Receptivo Local for de $5x5$, logo, a primeira camada oculta terá seu número de neurônios reduzido para 24×24 neurônios. [21]

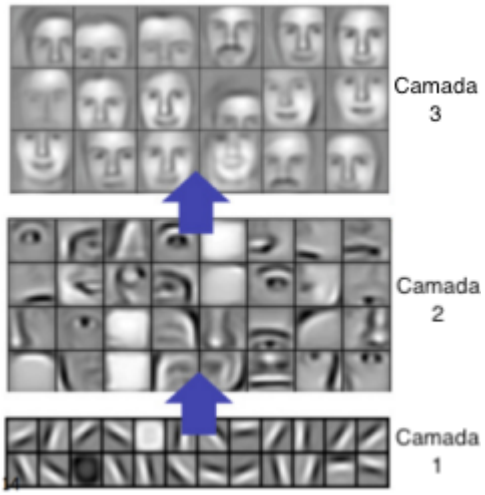


Figura 2.20: CNN extraindo características mais complexas a cada camada. [34]

De forma a detectar a mesma característica em todas as localizações possíveis da entrada, todos os neurônios de uma camada compartilham dos mesmos valores de pesos e

bias, de modo a analisar igualmente os campos da camada anterior [23]. Esse conjunto de pesos e *bias* compartilhados formam um *Kernel*, e sua saída é chamada de *Feature Maps* (em português, Mapa de Atributos) [21].

Pelo uso de pesos compartilhados, a rede consegue identificar determinada característica independente de translações, além de possuir uma menor necessidade de memória e ter uma maior velocidade de treinamento [19].

Outro parâmetro que pode ser modificado na rede é o *Stride*, que determina o "passo" do campo em uma entrada. Nos exemplos apresentados até o momento, o *Stride* utilizado tem valor 1, mas diferentes valores podem ser testados de forma a modificar o desempenho da rede.

Para que a extração de mais de uma característica ocorra, a camada $n + 1$ é formada por um conjunto de vários Mapas de Atributos, gerando assim um tensor.

Dessa maneira, cada camada da rede fica responsável por extrair uma determinada característica, conforme mostra a Figura (2.20), com características mais básicas nas camadas iniciais, como bordas e cantos numa imagem. Conforme avançam-se as camadas, a combinação dessas características básicas aumenta a complexidade das novas características descobertas. Por exemplo, extraindo então olhos, nariz, boca, chegando em faces completas na camada de saída [23] em problema de reconhecimento de faces.

Pooling

Após a camada de convolução, descrita anteriormente, pode existir uma camada de *Pooling*, responsável por diminuir a resolução do Mapa de Atributos. Com isso, a rede se torna menos sensível a translações e outras transformações [23].

A relevância disso dá-se por ser de maior importância uma determinada característica estar presente na imagem e estar próxima a outras características, do que identificar a sua exata localização. Isto é, é mais importante identificar, por exemplo, que na imagem existe uma boca em uma face, e que acima dela se encontra um nariz, do que saber, com exatidão de pixel, em que posição a boca se encontra [20].

Existem diversas formas de realizar *Pooling*. Uma das mais comuns é o uso do *Max*

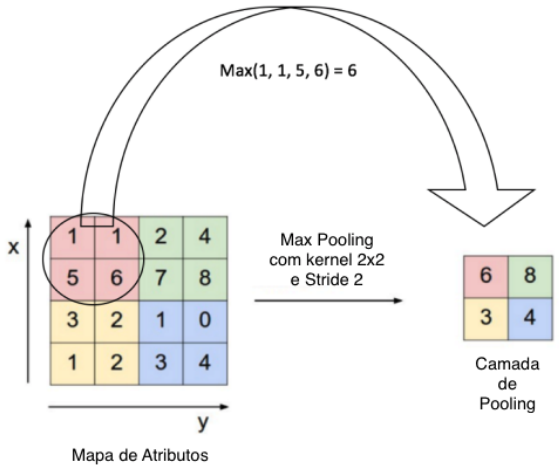


Figura 2.21: Exemplo de Max Pooling aplicado em uma entrada 4×4 com saída 2×2. [35]

Pooling (Figura (2.21)), no qual apenas o maior valor dentro de um determinado *Kernel* é repassado para a camada seguinte. Intuitivamente, pode-se pensar que apenas o neurônio que está mais excitado em relação a determinada característica, tendo assim o maior valor de saída no Mapa, será repassado para a próxima camada da rede.

Após as camadas de convolução e de *Pooling*, a CNN pode ser conectada a outras camadas totalmente conectadas, bem como outras arquiteturas de redes neurais.

Devido a sua estrutura, as CNNs são muito indicadas para problemas em que as entradas possuem representação estruturada em duas dimensões, como imagens, vídeos, e representações no domínio da frequência em áudios. Segundo Lecun et al. [23], seus usos diretos são em problemas como reconhecimento de escrita, reconhecimento óptico de caracteres, reconhecimento de fonemas, reconhecimento de falas, verificação de assinaturas, reconhecimento facial que é a principal utilização deste trabalho, entre outros. Ademais, podem ser utilizadas no Processamento Digital de Imagens e, por realizarem segmentação de imagens, também podem realizar extração de regiões de interesse.

2.4.3 *Generative Adversarial Networks*

Para problemas de classificação, os modelos apresentados até agora podem ser chamados de Discriminadores, pois os mesmos discriminam os dados de entrada entre as possíveis classes. Ou seja, a partir de um determinado exemplo de entrada, o discriminador deve mapear o exemplo a uma das possíveis classes. Nesse sentido, o discriminador é treinado utilizando técnicas de treinamento supervisionado, onde os dados de entrada e as suas respectivas classes são bem definidos.

Por outro lado, o treinamento não-supervisionado se refere a extração de características - ou padrões - a partir dos dados de entrada, tratando assim de problemas que não são bem definidos, já que os padrões ainda não são conhecidos [36]. Com isso, os modelos que tratam de dessa gama de problemas são chamados de Generativos, pois com o uso desses modelos pode-se gerar novos dados sintéticos para o espaço das entradas.

Generative Adversarial Networks (GAN)s são um modelo arquitetural de redes Generativas baseadas em *Deep Learning* (DL) e um *framework* de treinamento para essas redes. Descritas inicialmente por Goodfellow et al. [37], o treinamento das GAN utiliza-se de uma rede Discriminativa em conjunto com a rede Generativa, em que ambas "competem" entre si. Enquanto a rede Generativa tenta gerar dados cada vez mais significativos em sua saída, a rede Discriminativa, usando de um *dataset* conhecido acrescentado das saídas da rede Generativa, tenta classificar corretamente os exemplos do *dataset* e as saídas da rede em uma classificação binária. De outra forma, o trabalho da rede Generativa é "enganar" a rede Discriminativa, e o desta, por sua vez, é descobrir quais imagens são as reais e quais são as falsas. Após o treinamento, o Discriminador pode ser descartado, já que o objetivo é utilizá-lo unicamente para treinar a Generativa.

Desta forma, o treinamento da rede Generativa - que inicialmente era classificado como não-supervisionado - passa a ser supervisionado, já que o problema a ser resolvido pelo Discriminador é bem definido [38]. Outra característica importante desse tipo de arquitetura de redes neurais é que, devido à rede Generativa ter características de problemas não-supervisionados, diz-se que a mesma sofre de *Inceptionism* ou "alucinações" em seu

treinamento. Essas ditas alucinações são interpretações exageradas de padrões encontrados nos dados de entrada e das características conhecidas pela rede. Um exemplo de alucinações em GAN para imagens pode ser visto na Figura (2.22).

(a) (b)

Figura 2.22: Exemplo de alucinações em GANs. (a) é a imagem utilizada pelo Discriminador da rede DeepDream [39] durante o treinamento, e (b) é o *output* da Generativa. Pode-se perceber exageros nos padrões de características conhecidos pela rede.

Segundo a Teoria dos Jogos, o treinamento das redes GAN pode ser classificado como um jogo de soma-zero, em que as redes são adversárias uma da outra [38]. Cada vez que a Generativa engana a Discriminadora, a primeira é recompensada ou nada acontece, e a última é penalizada - tendo seus pesos modificados. Já quando a Discriminadora acerta a classificação da imagem da Generativa, o contrário ocorre. Com isso, no limite, a Generativa geraria réplicas tão perfeitas que a Discriminadora não teria como possívelmente acertar, fazendo com que essa tenha que adivinhar todas as imagens, com erro de $\sim 50\%$.

A função de custo da GAN é baseada na equação da *Cross-Entropy* (CE) (em português, Entropia Cruzada), descrita na Equação (2.24). CE mensura a diferença entre duas distribuições de probabilidades para um conjunto de eventos [40]. Para tal, a CE usa de um conceito originário da Teoria da Informação, em que a informação quantifica o número de *bits* necessários para codificar e transmitir um evento. A quantidade de informação contida em uma mensagem está relacionada com a raridade de ocorrência do mesmo. Eventos com maior probabilidade de ocorrência trazem menos informação que eventos

com menor probabilidade de ocorrer, já que os de maior probabilidade de acontecer já são esperados, trazendo assim pouca novidade, ou seja, pouca informação nova [41].

Entropia é o número de bits necessários para transmitir um evento aleatório em uma distribuição de probabilidade. Eventos com maior probabilidade tem uma menor entropia, enquanto eventos com menor probabilidade possuem maior entropia. A *Cross-Entropy* (CE) utiliza do conceito de entropia para calcular o número médio de *bits* necessários para transmitir um evento de uma distribuição P comparado com outra distribuição Q [40]. Tem-se então a Equação da CE (2.24).

$$H(P, Q) = \mathbb{E}_{x \sim P}[-\log(Q(x))] \tag{2.24}$$

Onde $P(x)$ é a probabilidade do evento x em P, $Q(x)$ é a probabilidade do evento x em Q, e, dada uma função $f(x)$, $\mathbb{E}_{x \sim P}[f(x)]$ é a somatória de $P(x) * f(x)$, que tomando $f(x) := \log(Q(x))$ equivale a Equação (2.25).

$$-\sum_{x}^{X} P(x) * \log(Q(x)) \tag{2.25}$$

A partir disso, a Equação (2.26) descreve a função de custo utilizada no treinamento das GANs [37].

$$min_G \; max_D \; V(D, G) = \mathbb{E}_{x \sim P_{data}(x)}[\log(D(x))] + \mathbb{E}_{z \sim P_z(z)}[\log(1 - D(G(z)))] \tag{2.26}$$

Onde $G(\cdot)$ é a rede Generativa, $D(\cdot)$ é a rede Discriminadora, $\mathbb{E}_{x \sim P_{data}(x)}[\cdot]$ é a operação de cálculo da média de todos os dados no mini-batch, e $V(D, G)$ é a função valor entre a Generativa e a Discriminativa.

Na prática, como nos estágios iniciais do treinamento a acurácia da Generativa é muito baixa e a Discriminativa acerta facilmente a maioria dos casos, $\log(1 - D(G(z)))$ satura. Então, ao contrário de treinar a Generativa para minimizar $\log(1 - D(G(z)))$, pode-se treinar a Generativa para maximizar $\log(D(G(z)))$, de forma a ter gradientes mais significativos nos primeiros estágios do treinamento [37].

Com isso, tem-se que a rede Generativa deve minimizar a Equação (2.27), e a rede Discriminadora deve minimizar a Equação (2.28).

$$L_G = \mathbb{E}_{x \sim p_{data}(x)}[-\log(1 - D(x))] + \mathbb{E}_{z \sim p_z(z)}[-\log(D(G(z)))] \tag{2.27}$$

$$L_D = \mathbb{E}_{x \sim p_{data}(x)}[-\log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[-\log(1 - D(G(z)))] \tag{2.28}$$

A rede Discriminadora estima a probabilidade que o *input* é real, enquanto a rede Generativa é treinada para maximizar a probabilidade que um dado falso é real. Segundo Jolicoeur-Martineau [42] a rede Generativa deveria também minimizar a probabilidade de um dado real ser real, e que, empiricamente, isso traria maior estabilidade e melhores resultados no treinamento da rede. Wang et al. [43] chegaram na mesma conclusão em seu trabalho com GAN para Super-Resolução, onde as imagens geradas pela rede teriam melhor qualidade comparado a imagens geradas por redes que que não utilizavam desse conceito.

Para esse fim, modifica-se na equação original 2.26 o parâmetro $D(x)$, onde $D(x) = sigmoid(C(x))$ torna-se $D(x) = sigmoid(C(x_r) - C(x_f))$, em que $C(x)$ é o *output* não transformado do Discriminador, x_r é o dado real e x_f é o dado falso criado pela rede Generativa [42]. Com isso, tem-se então o *Relativistic Standard Generative Adversarial Network* (RSGAN), cuja função de custo está descrita na Equação (2.29) [42].

$$\begin{aligned} L_G &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q}))}[\log(sigmoid(C(x_f) - C(x_r)))] \\ L_D &= -\mathbb{E}_{(x_r, x_f) \sim (\mathbb{P}, \mathbb{Q}))}[\log(sigmoid(C(x_r) - C(x_f)))] \end{aligned} \tag{2.29}$$

Pode-se então interpretar que o Discriminador estima a probabilidade de um dado real ser mais realista que um dado aleatório produzido pela rede Generativa [42]. Esse conceito será mais explorado na Seção 3.1.2, onde será tratado sobre Super-Resolução utilizando redes GAN.

2.5 Considerações finais

Neste capítulo foram apresentados conceitos fundamentais para a compreensão do trabalho proposto, tais como *Machine Learning*, *Neural Network*, *Digital Image Processing*, *Convolutional Neural Network*, e *Generative Adversarial Networks*. Esses conceitos serão utilizados como base para o conteúdo do capítulo seguinte, onde será tratado sobre *Face Recognition*, *Super-Resolution*, e como essa pode ajudar a reconhecer melhor os rostos em situações onde a qualidade de imagem não atinge os padrões desejáveis.

Capítulo 3

Reconhecimento Facial e Super-Resolução

Neste capítulo serão abordados os temas de Super-Resolução (Seção 3.1), Detecção Facial (Seção 3.2.1) e Reconhecimento Facial (Seção 3.2.2) com mais profundidade, sendo também apresentados modelos de redes importantes e estado da arte na literatura.

3.1 Super-Resolução

Segundo Milanfar [10], "Super-Resolução (SR) consiste em um conjunto de técnicas, utilizando uma ou mais imagens de baixa resolução de uma mesma cena, para reconstruir uma imagem em alta resolução, removendo artefatos degradantes e melhorando de forma geral a qualidade da imagem". No caso da utilização de várias imagens, utiliza informações não-redundantes de uma mesma cena para estimar a imagem final em alta resolução [10].

Tais técnicas podem ser aplicadas em variadas situações, nas quais a melhoria na qualidade de imagem se faz necessária, tais como imagens aéreas e de satélite, processamento de imagens médicas, e compressão de imagem ou vídeo [44]. Para o caso de biometrias, uma de suas limitações é a necessidade de uma curta distância entre o sujeito a ser analisado e o hardware responsável pela captura dessas informações. Como exemplos, tem-se a distância entre o olho humano e o sensor de iris, do dedo a um sensor biométrico, ou da

face em relação a câmera. Nestes casos, a SR auxilia tanto no aumento da resolução das imagens adquiridas à distância, quanto na melhoria da qualidade das mesmas [4].

Os algoritmos de SR podem ser aplicados no domínio do espaço ou da frequência. No primeiro caso, modifica-se a intensidade dos pixels, de forma a modelar uma ampla gama de movimentações e degradações, e incluem conhecimento prévio para regularizações. Dessa forma, são algoritmos mais flexíveis, porém tem maior complexidade computacional. Já para o segundo, utiliza-se da representação no espaço da frequência para modelar de forma mais fácil as frequências geradoras de *aliasing*, sendo esse processo mais simples, melhor paralelizável, porém mais limitado [4].

A SR é caracterizada como um problema mal-posto, pois várias imagens de alta resolução podem corresponder a uma mesma imagem de baixa resolução [44], e pode ser dividida em três tarefas a serem realizadas [4]:

- Design do modelo de baixa resolução a partir da imagem de alta resolução;
- Alinhamento da imagem;
- Reconstrução da imagem em alta resolução.

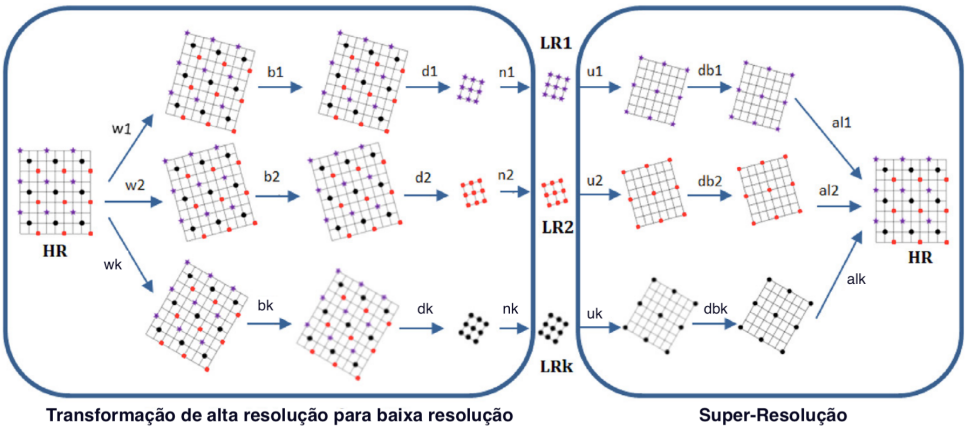


Figura 3.1: Ilustração da geração de imagens de baixa resolução (LR) a partir de alta resolução (HR) para serem utilizadas nos algoritmos de Super-Resolução, onde LRk representa as diferentes imagens de baixa resolução geradas. [4]

Para o modelo de baixa resolução utiliza-se, como base, uma imagem de alta resolução X, que terá sua resolução diminuída pela função $D_k(\cdot)$, sofrerá distorções $W_k(\cdot)$ e será

acrescida de *blur* $B_k(\cdot)$ e ruído N_k, gerando assim uma ou mais imagens de baixa resolução Y_k, conforme descreve a Equação (3.1) e ilustra a Figura (3.1) [4].

$$Y_k = D_k(B_k(W_k(X))) + N_k \tag{3.1}$$

O processo de alinhamento de imagem utiliza-se de duas ou mais imagens da mesma cena, geradas em tempos diferentes e/ou com diferentes pontos de vista e alinha-as pelo uso de um mapeamento nos pixels. Esse mapeamento ocorre em três passos. No primeiro, realiza-se a extração de características da imagem e identificação. Após, estima-se os parâmetros da transformação e, finalmente, utiliza-se a função de distorção na imagem [4].

Na reconstrução da imagem, segundo [4], pode-se dividir os métodos em duas categorias, sendo elas:

- **Métodos de reconstrução**: são técnicas para recuperar as componentes de alta frequência da imagem em múltiplas imagens complementares. Dessa forma, quando suprido com os detalhes necessários, realizam transformações extremamente precisas;

- **Métodos de aprendizado**: essas técnicas realizam o aprendizado das componentes de alta frequência da imagem, baseando-se em um *dataset*, por isso correm o risco de aprender de forma errônea, melhorando a qualidade da imagem para a percepção humana mas diminuindo a taxa de acerto da máquina, por exemplo em problemas de identificação de padrões, como identificado por Cheng, Zhu e Gong [5].

3.1.1 Super-Resolução utilizando Redes Convolucionais

Redes Convolucionais podem ser utilizadas para a reconstrução da imagem, sendo a VDSR [45] uma das redes estado da arte e base de comparação para uma grande quantidade de artigos, além de ser uma das primeiras a modificar o modelo inicial *Super-Resolution Convolutional Neural Network* (SRCNN) proposto por Dong, Loy, He et al. [46] [47].

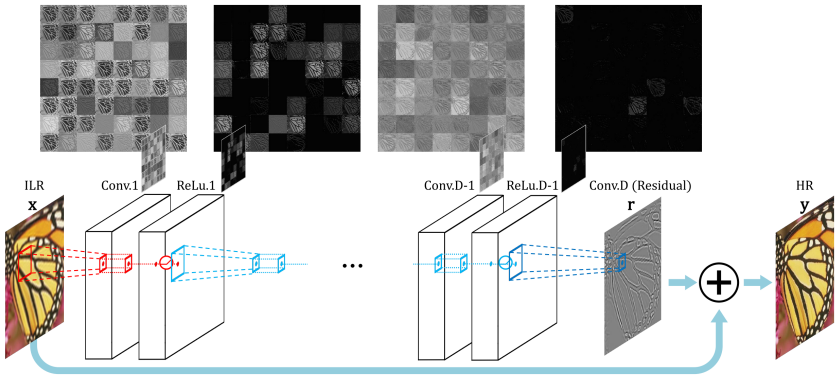


Figura 3.2: Arquitetura da rede VDSR, onde a imagem interpolada de baixa resolução (LR) passa por pares de camadas de convolução e ReLU, passando por uma última camada de *Residual-learning*, até gerar na saída da rede a imagem em alta resolução (HR). [45]

A VDSR (Figura (3.2)), baseada na VGGNet [24], possui d camadas, onde todas as camadas ocultas contém 64 filtros de tamanho $3 \times 3 \times 64$ (o filtro opera numa região de 3×3 em 64 mapas de característica). Nas camadas intermediárias, são intercalados uma camada de convolução e uma não-linear com a função de ativação ReLU (Seção 2.4) [45].

Em todas as camadas é utilizado o Zero Padding (Equação (3.3)), onde dada a matriz de entrada A (Equação (3.2)), acrescenta-se zeros no entorno da matriz, de modo a fazer com que, quando o filtro percorrer a matriz, o mesmo consiga captar os detalhes nas bordas da imagem. Além do mais, faz com que todos os mapas de características tenham o mesmo tamanho, de forma que a imagem final não tenha seu tamanho reduzido em comparação a imagem original [45].

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ h & j & k \end{bmatrix} \tag{3.2}$$

$$Pad(1, A) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a & b & c & 0 \\ 0 & d & e & f & 0 \\ 0 & h & j & k & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

Outra técnica empregada pelos autores é o *Residual-Learning* [48], de forma a combater o problema da dissipação/explosão do gradiente, onde a imagem residual r segue a equação $r = y - x$, onde x é a imagem em baixa resolução e y a imagem em alta resolução original. Ou seja, a rede deve aproximar não a imagem em alta resolução, mas sim a diferença entre a imagem em alta resolução original e a imagem em baixa resolução. Um usuário do modelo deve somar a saída da rede com a imagem de baixa resolução para obter a reconstrução em alta resolução. Combinando com a equação de MSE (Equação (2.11)), tem-se o custo conforme descrito na Equação (3.4), onde $f(x)$ representa a predição da rede sobre a entrada x [45].

$$C(\vec{w_k}) \equiv \frac{1}{2} \sum_{l=0}^{j} (r - f(x))^2 \tag{3.4}$$

Segundo Hayat [44], conforme citado por Kim et al. [49], camadas de *pooling* e *sub-sampling* em redes convolucionais para Super-Resolução não são indicadas, pois detalhes importantes da imagem podem ser descartados no processo. Por essa razão não existem camadas de *pooling* na rede VDSR.

3.1.2 Super-Resolução utilizando Redes GAN

Realizar a minimização de funções de custo baseadas na MSE é conveniente pois ao mesmo tempo ocorre a maximização do *Peak Signal-to-Noise Ratio* (PSNR), que é uma medida de acurácia comum em problemas de SR [50]. O problema dessa abordagem é que a habilidade dessas métricas em capturar as características perceptuais da imagem (como texturas em alta qualidade) é deveras limitada por serem baseadas na diferença entre os

pixels, resultado assim em texturas com muito *blur* [50]. Ledig et al. [50] propuseram o uso de uma nova função de custo que leva em conta a Distância Euclidiana (Equação (3.18)) entre os *feature maps* das imagens de alta resolução e SR extraídos após a passagem pela função de ativação na rede VGG19 pré-treinada, resultando assim em imagens com maior qualidade perceptual. Essa função está descrita na Equação (3.5) [50].

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \tag{3.5}$$

Onde $\phi_{i,j}$ é o *feature map* obtido pela j-ésima convolução antes da i-ésima camada de *max-pooling*, $W_{i,j}$ e $H_{i,j}$ referem-se à largura e altura da dimensão do *feature map* da rede VGG19. I^{HR} é a imagem em alta qualidade, e I^{SR} é a imagem de resolução aumentada.

Outra contribuição dos autores foi empregar pela primeira vez as técnicas de GAN para o problema da SR, aproveitando das características inerentes a esse modelo de redes, e assim superando as redes existentes e se tornando SOTA. Essa nova rede é denominada SRGAN.

A arquitetura da rede Generativa (Figura (3.3)) utiliza *residual blocks*, com duas camadas de convolução de *kernel* 3×3 e 64 *feature maps*, seguido de camadas de *Batch Normalization* e *ParametricReLU* como função de ativação. Após isso, apresenta também camadas de deconvolução para aumentar a resolução da imagem [50].

Na rede Discriminadora são utilizados oito camadas convolucionais com *kernel* de 3×3, seguidas por *Batch Normalization*, aumentando o número de *feature maps* a cada dois blocos de convolução por um fator de 2, partindo de 64 até 512 *maps*. A função de ativação utilizada foi a *LeakyReLU* e não foi utilizado *max-pooling* na rede. As convoluções são seguidas por duas camadas totalmente conectadas, com função Sigmoid (Equação (2.7)) na saída para obtenção da probabilidade de classificação da imagem [50].

A partir das ideias propostas por Ledig, Theis, Huszar et al. [50], Wang, Yu, Wu et al. [43] propuseram modificar a arquiterura da rede Generativa, introduzindo o conceito de *Residual-in-Residual Dense Block* (RRDB), que traria maior capacidade e mais velocidade no treinamento, tendo assim a denominada ESRGAN. Para tal, nos *residual blocks*

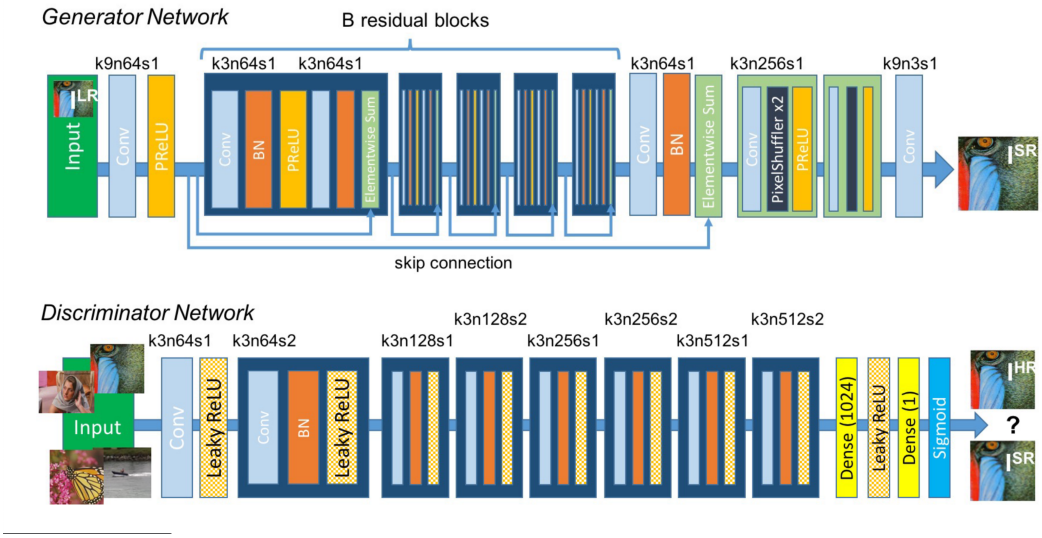


Figura 3.3: Arquitetura da rede SRGAN. [50]

foi removido a camada de *Batch Normalization* (BN), para reduzir a complexidade do treinamento e uso de memória, bem como remover artefatos que a camada causa devido as médias e desvios padrão das imagens do treinamento diferirem muito das imagens de teste, que por consequência diminui o poder de generalização da rede [43].

Outra modificação foi a utilização de *skip connections* entre as camadas de convolução nos *residual blocks*, ponderadas por um fator de escala entre 0 e 1 antes da adição - para prevenir instabilidades no treinamento [43]. Com isso, tem-se o denominado RRDB, ilustrado na Figura (3.4).

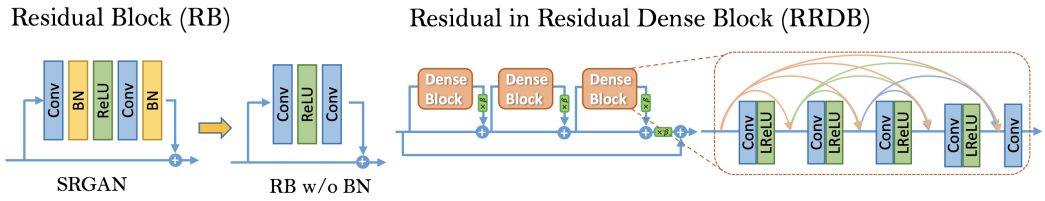


Figura 3.4: Exemplo de *Residual-in-Residual Dense Block*. [43]

Além da criação dos RRDB, os autores modificaram a rede Discriminadora para utilizar o conceito de *Relativistic Average Generative Adversarial Network* (RaGAN), onde a rede tenta predizer a probabilidade de uma imagem real ser relativamente mais realista

que as imagens em SR. O conceito de RaGAN é uma extensão do conceito de RSGAN (Equação (2.29)), em que a imagem de real é comparada com a média das imagens geradas pela rede Generativa, descrito na Equação (3.6) [43].

$$D_{Ra}(x_r, x_f) = \sigma(C(x_r) - \mathbb{E}_{x_f}[C(x_f)]) \tag{3.6}$$

Onde x_r é a imagem real, x_f é a imagem gerada pela rede Generativa, σ é a função sigmoid (Equação (2.7)), $C(\cdot)$ é a saída não transformada do Discriminador, e $\mathbb{E}_{x_f}$ é a função que realiza a média.

Com a introdução desse novo conceito, tem-se então a nova equação da rede Discriminadora, descrito na Equação (3.7) [43].

$$L_D^{Ra} = -\mathbb{E}_{x_r}[\log(D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f}[\log(1 - D_{Ra}(x_f, x_r))] \tag{3.7}$$

E, de forma simétrica, o custo da rede Generativa pode ser descrito como na Equação 3.8 [43].

$$L_G^{Ra} = -\mathbb{E}_{x_r}[\log(1 - D_{Ra}(x_r, x_f))] - \mathbb{E}_{x_f}[\log(D_{Ra}(x_f, x_r))] \tag{3.8}$$

Com isso, tem-se então que a função de custo da rede Generativa é conforme descrito na Equação 3.9 [43].

$$L_G = L_{percep} + \lambda L_G^{Ra} + \eta L_1 \tag{3.9}$$

Onde L_{percep} é conforme descrito na Equação (3.5) aplicada antes da função de ativação da rede VGG19, L_G^{Ra} é a Equação (3.8) descrita anteriormente, e L_1 é a *1-norm distance* entre a imagem real x_r e a imagem gerada pela rede Generativa $G(x_i)$ a partir da entrada x_i, conforme descreve a Equação 3.10. λ e η são coeficientes para balancear os diferentes termos.

$$L_1 = \mathbb{E}_{x_i} \parallel G(x_i) - x_r \parallel_1 \tag{3.10}$$

A partir desses avanços a rede ESRGAN se tornou SOTA em SR, e sua comparação com outras redes pode ser visto na Figura (3.5).

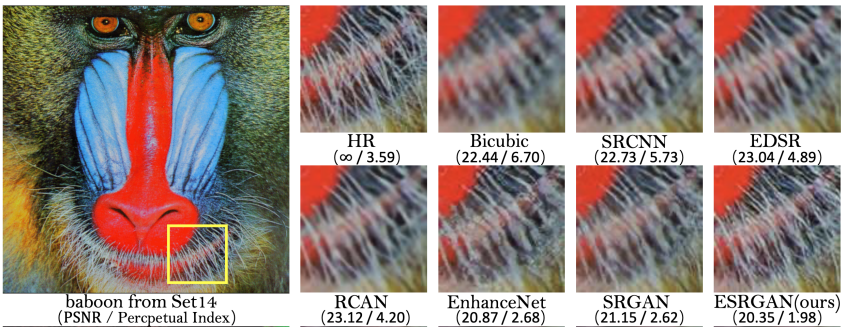


Figura 3.5: Comparação da rede ESRGAN com outras redes de SR. [43]

A utilização de SR pode trazer muitos benefícios quando em conjunto com Reconhecimento Facial, e esse conceito será melhor exposto na Seção 3.3, junto com os trabalhos relacionados na área.

3.2 Reconhecimento Facial

O *Face Recognition* (FR) é de grande importância para uma série de aplicações em segurança, onde a identificação de indivíduos se faz necessária, como entrada em estabelecimentos, monitoramento interno, monitoramento de espaços públicos, dentre outros.

É possível dividir o problema de reconhecer faces em três partes: detecção da face, extração das características biométricas e reconhecimento do indivíduo. A detecção da face ocorre no estágio inicial, onde procura-se identificar e extrair os rostos contidos em determinada imagem. Após a detecção, ocorre a extração das características do rosto, onde leva-se em conta propriedades como regiões do rosto, expressões faciais, orientação da face, dentre outras. Essas características são utilizadas para reconhecimento do indivíduo, utilizando algoritmos de classificação [51].

3.2.1 Detecção Facial

A *Face Detection* (FD), um subproblema de detecção de objetos, pode parecer uma tarefa simples, já que rostos possuem formato bem-modelado. Porém, segundo [52], alguns desafios precisam ser solucionados, tais como uso de óculos, presença barba, escala da imagem, expressões faciais, oclusão, iluminação da imagem, ângulo da face em relação a fotografia, dentre outros.

Tem-se como trabalho pioneiro para FD o método de Haar Cascade proposto por [53], onde divide-se a detecção em quatro etapas [54]:

- Seleção de características tipo-Haar;
- Cálculo da imagem integral;
- Treinamento utilizando AdaBoost;
- Utilização de classificadores em cascata.

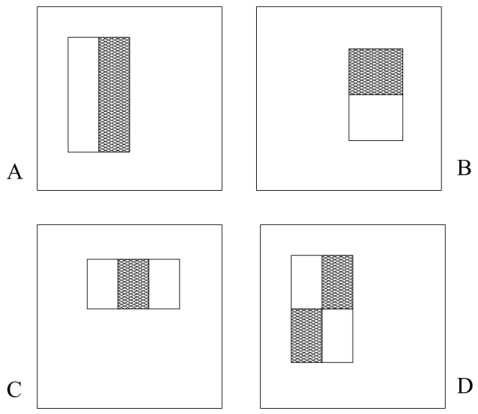


Figura 3.6: Exemplos de características tipo-Haar utilizadas por Viola e Jones, sendo (A) e (B) com dois retângulos, (C) com três retângulos e (D) com quatro retângulos. [53]

A seleção de características ocorre baseando-se em propriedades comuns nos rostos humanos: região dos olhos ser mais escura que a região do nariz, a região mais alta das bochechas ser mais clara que a região dos olhos, a localização dos olhos, nariz e boca,

dentre outras. De forma a identificar essas regiões, utiliza-se de características tipo-Haar, ilustradas nas Figura (3.6), onde o valor da região é calculado pela subtração dos valores dos pixels da região branca pelos pixels da região preta. Se o valor da região estiver acima de um determinado limiar, a região é selecionada.

Para realizar o cálculo do valor dos pixels com rapidez, os autores propuseram um método de cálculo da imagem integral, descrito na Equação (3.11), onde $ii(x, y)$ é a imagem integral no pixel (x, y), e $i(x', y')$ é a imagem original (Figura (3.7)) [53].

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.11}$$

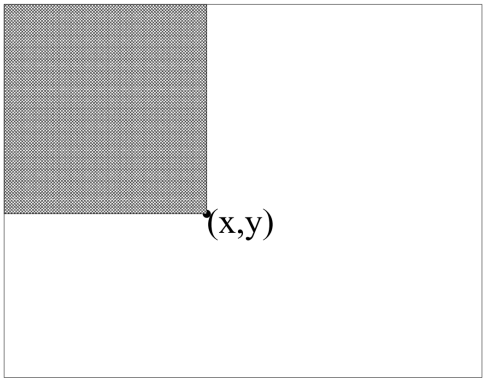


Figura 3.7: O valor da imagem integral no ponto (x, y) é o somatório dos pixels acima e à esquerda do ponto. [53]

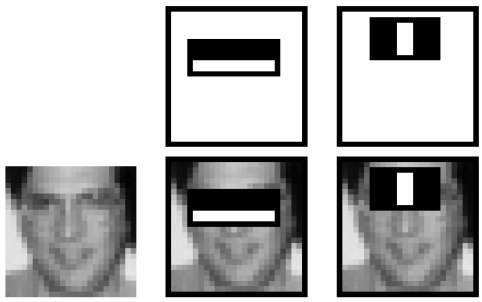


Figura 3.8: Exemplos de características tipo-Haar aplicadas em uma face. Acima temos a característica e abaixo a sua aplicação, sendo que a primeira característica seleciona a diferença de tons de cinza entre a região das bochechas e os olhos, e a segunda seleciona a diferença entre as regiões dos olhos e a região da ponte entre os olhos. [53]

Técnicas de *Boosting* são uma maneira de aprendizado de classificadores, utilizando a combinação de um conjunto de classificadores fracos obter uma solução otimizada. Dessa forma, cada característica tipo-Haar pode ser considerada um classificador fraco. Por exemplo, uma característica pode ser ótima em determinar a região dos olhos, já que a região dos mesmos é mais escura que a região do nariz, mas essa mesma propriedade poder determinar outras regiões na imagem que não fazem parte do rosto ou não ser otimizada para encontrar diferentes regiões da face. Por isso, os autores utilizaram o *Adaptive Boosting* (AdaBoost) como forma de agrupar esses classificadores, de modo que se uma característica falhar, a mesma não será passada para o próximo classificador (Figura (3.8)) [52].

Finalizando o algoritmo, os classificadores otimizados são então selecionados e combinados em cascata, sendo então descartados os classificadores que reconhecerem outras regiões diferentes da face na imagem [55]. Com esse método proposto, os autores conseguiram um algoritmo que possui uma boa relação entre precisão e eficiência em cenários simples e fixos [56].

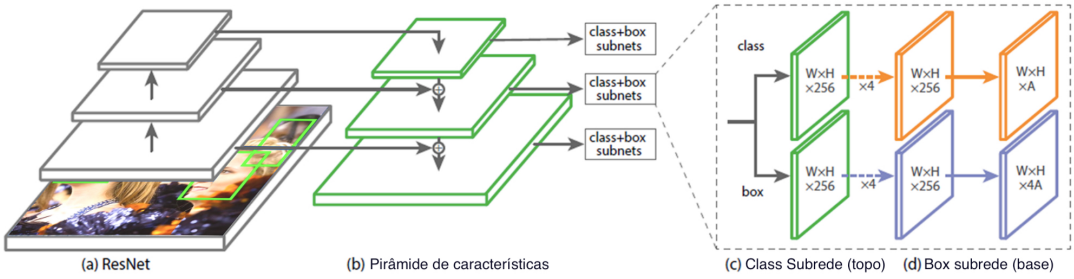


Figura 3.9: Arquitetura da rede AInnoFace, baseada na ResNet-152 (a), com seis níveis de FPN (b), anexada a uma subrede de classificação (c) e uma subrede de regressão (d). [56]

Pode-se também utilizar de Redes Convolucionais (Seção 2.4.2) para a detecção das faces. A AInnoFace [56] (Figura (3.9)), proposta por Zhang et al., atual estado da arte (Junho de 2019) no *dataset* WIDER FACE, foi criada com base na RetinaNet [57] (proposta por Tsung et al.), que utiliza a ResNet-152 [48] como base, em que essa possui 152 camadas entre convoluções e *poolings*.

A RetinaNet é uma rede de detecção de objetos que traz uma nova função de custo, denominada de *Focal Loss*, que é baseada na *Cross-Entropy* $CE(\cdot)$, que está descrita na Equação (2.24), cuja versão adaptada para classificação binária está descrita na Equação (3.12) [57].

$$CE(p, y) = \begin{cases} -\alpha_t \log(p) & \text{se } y = 1 \\ -\alpha_t \log(1-p) & \text{se } y \neq 1 \end{cases} \tag{3.12}$$

Sendo que $y \in \{\pm 1\}$ especifica a classe verdadeira, $p \in [0, 1]$ é a probabilidade estimada para a classe com rótulo $y = 1$, e $\alpha_t \in [0, 1]$ para classe 1 e $1 - \alpha_t$ para classe -1, de forma a balancear a importância de exemplos positivos e negativos. Nota-se na Figura (3.10), que mesmo para classes facilmente preditas ($p_t \gg 0.5$), a Cross Entropy possui um custo não-negligenciável, que, quando somado ao longo de vários exemplos, pode subjugar uma classe rara (como um rosto pequeno em uma imagem de alta resolução). Para resolver esse problema, a *Focal Loss* $FL(\cdot)$ modifica a Cross Entropy, adicionando o termo $(1 - p_t)^\gamma$, conforme descrito na Equação (3.13) [57].

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t) \tag{3.13}$$

Com isso, quando um exemplo é classificado erroneamente e p_t possui valor pequeno, o fator modulador $(1 - p_t)^\gamma$ tende a 1, fazendo o custo não ser afetado. Conforme $p_t \to 1$, o fator vai para 0 e o custo para exemplos classificados corretamente tem seu peso diminuído. Pode-se modificar o valor de γ, de forma a ajustar razão com que exemplos fáceis tem seu peso diminuído, e caso $\gamma = 0$, tem-se a Cross Entropy original [57].

Outra característica da RetinaNet é o uso de *Feature Pyramid Network* (FPN) [58], onde, tendo como base a convolução, cria-se uma pirâmide, indo do topo para a base, utilizando também de conexões laterais, de forma a fazer com que a rede construa uma pirâmide de características rica e multi-escala baseada em uma única imagem [57].

De forma a mitigar alguns dos problemas citados por [52], utiliza-se dois agrupamentos de técnicas de *data augmentation*, sendo eles [59]:

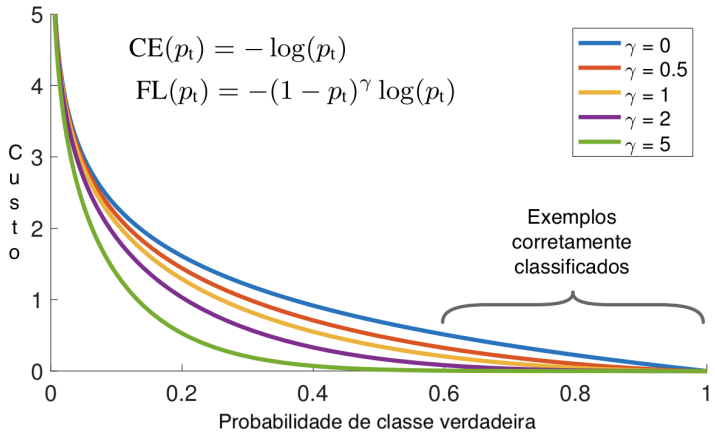


Figura 3.10: Gáfico das funções *Cross-Entropy* (CE) e *Focal Loss*. [57]

- **Incremento de um para muitos**: realiza a geração de novas imagens baseadas no *dataset* atual com rotações e translações na face, de forma a fazer com que a rede se torne invariante a poses;
- **Normalização de muitos para um**: realiza a transformação a partir de uma ou mais imagens de visão não-frontal da face para uma de visão frontal, para que o RF possa ser realizado em condições não controladas.

Além da estrutura da ResNet e outras inovações, a AInnoFace utiliza de algumas técnicas de *data augmentation*, como a expansão e diminuição das imagens com adição de distorções fotométricas aleatórias e espelhamento. Outra técnica empregada por [56] é a *anchor-based sampling*, onde uma face de tamanho S_{face} é escolhida aleatoriamente em um *batch*, então é selecionada uma âncora de escala S_{anchor}, depois calcula-se uma escala aleatória S_{random} em torno da âncora, e, por fim, redimensiona-se a imagem seguindo a equação $S^* = S_{random}/S_{face}$.

3.2.2 Extração de características e Classificação

Após o processamento das faces, extrai-se o vetor de *embedding*, que é um mapeamento das características dos rostos em valores reais para um vetor de dimensão d. Pelo uso desse vetor, pode-se realizar o cálculo da distância entre o mesmo e outro vetor de *embedding*,

buscando quantificar quão similar são os mesmos. Como esse vetor é uma representação das características de uma face, vetores com curta distância entre si representam faces parecidas (ou seja, uma mesma classe), enquanto vetores com grande distância respresentam faces de pessoas diferentes (ou seja, classes diferentes).

Para que o sistema de FR tenha bons resultados, *datasets* volumosos devem ser utilizados para o treinamento das redes neurais. Empresas como Google e Facebook utilizam *datasets* na ordem de $10^6 - 10^7$ indivíduos, sendo que os *datasets* disponíveis publicamente variam entre $10^3 - 10^5$ indivíduos. De forma melhorar os resultados, utiliza-se de redes mais profundas e funções de custo mais efetivas, para extrair características mais significativas e discriminativas. Algumas funções para tal são: Euclidian-distance-based Loss, Angular/Cosine-margin-based Loss, Softmax Loss e Triplet Loss [59].

O Estado da Arte em Junho de 2019, proposto por Deng et al. [60], utiliza a função de custo *Additive Angular Margin Loss* (ArcFace) Φ, baseada na função Softmax Loss σ que está descrita na Equação (3.14).

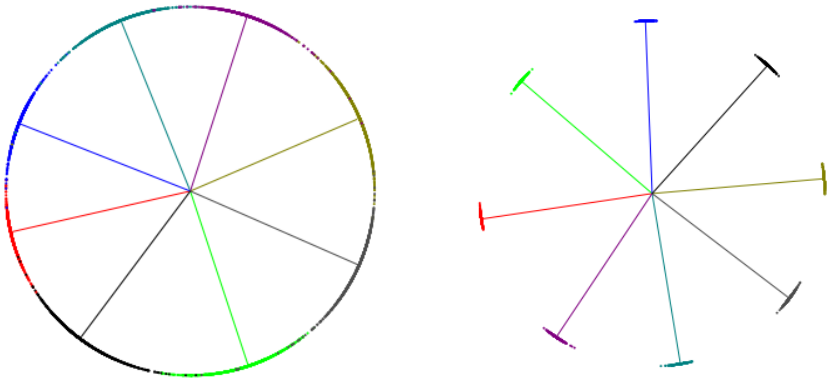
$$\sigma = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}} \tag{3.14}$$

Onde $x_i \in \mathbb{R}^d$ representa a característica da i-ésima instância, pertencendo a y_i-ésima classe, $W_j \in \mathbb{R}^d$ representa a j-ésima coluna da matriz de pesos $W \in \mathbb{R}^{d \times n}$ e $b_j \in \mathbb{R}^n$ é o *bias*. O tamanho do lote é representado por N, o número da classe é representado por n, e o vetor de características d é usualmente de tamanho 512.

Como a *Softmax Loss* não otimiza o vetor de *embedding* para reforçar uma maior similaridade em exemplos intra-classe e discrepância em exemplos entre diferentes classes, resultando assim em uma baixa performance, os autores propuseram mudanças na função original, fixando *bias* $b_j = 0$, transformando $W_j^T x_i = \| W_j \| \| x_i \| \cos\theta_j$ (onde θ_j é o ângulo entre W_j e a característica x_j), fixando $\| W_j \| = 1$ pela normalização l_2, fixando $\| x_i \|$ também pela normalização l_2 e reescalando-o para s, resultando na Equação (3.15)

[60].

$$\sigma_2 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos\theta_j}} \tag{3.15}$$



(a) Softmax Loss (b) ArcFace Loss

Figura 3.11: Comparação entre as funções Softmax Loss e ArcFace para um exemplo de 8 classes com vetor de características 2D. Pontos indicam instâncias e linhas referem-se a direção do centro de cada classe. Nota-se claramente que as instâncias intra-classe estão mais agrupadas e há também uma maior separação entre diferentes classes no resultado da função ArcFace Loss. [60]

A normalização faz com que as predições dependam somente do ângulo entre a característica e o peso. Como o *embedding* aprendido é distribuído em uma hiperesfera de raio s, adicionando um parâmetro angular m entre x_i e W_{y_i} faz com que a distância intra-classe seja minimizada e a discrepância entre diferentes classes seja aumentada (como ilustrado na Figura (3.11)), resultando assim na Equação final 3.16 [60].

$$\Phi = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos(\theta_{y_i+m})}}{e^{s\cos(\theta_{y_i+m})} + \sum_{j=1,j\neq y_i}^{n} e^{s\cos(\theta_j)}} \tag{3.16}$$

Pode-se dividir a classificação da face em dois tipos, identificação e verificação. A

verificação compara o rosto de entrada com cada um dos rostos contidos no *database*, realizando comparações $1:1$, de forma a determinar se ambas as imagens representam a mesma pessoa. Uma utilização é em *logins* em sistemas como forma de confirmar se determinado indivíduo é realmente quem diz ser. Já a identificação realiza comparações $1:N$, onde N representa o número de faces contidas no *database*, onde busca-se identificar se determinada face está contida naquele sistema ou não, e a quem pertence aquela face caso esteja contida no mesmo [59].

Após a extração do vetor de características, calcula-se a distância entre as mesmas, de forma a calcular a similaridade entre características e utilizar esses valores para realizar a classificação. Para esse cálculo, algumas funções possíveis são Similaridade do Cosseno (Equação (3.17)) e Distância Euclidiana L_2 (Equação (3.18)) [59].

$$\cos(\theta) = \frac{A \cdot B}{\parallel A \parallel \parallel B \parallel} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.17}$$

$$L_2 = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2} \tag{3.18}$$

Onde A e B são os *embedding* de tamanho n, e A_i e B_i as componentes do vetor.

Com a distância calculada, realiza-se a classificação dos rostos, buscando identificar nos tipos de classificação apresentados anteriormente. Para os testes, utiliza-se então uma comparação de *threshold* e algoritmo *k-Nearest Neighbors* (k-NN) para classificar os rostos tanto em verificação quanto em identificação das faces [59].

O algoritmo k-NN utiliza de k vizinhos mais próximos da nova entrada x para calcular a qual classe n essa entrada pertence. Para isso, utiliza-se da distância de x até as outras instâncias para encontrar os k vizinhos mais próximos e usa voto plural (nos casos de classificação) ou média (nos casos de regressão) para selecionar a classe de pertencimento [14].

Conforme demonstrado por Cheng et al. [5], mesmo as redes estado da arte em RF não conseguem atingir boas taxas de reconhecimento em imagens nativamente em baixa resolução, sendo que, no melhor dos casos, a rede CenterFace conseguiu 32.1% de acerto

no *dataset* TinyFace. Como forma de melhorar essas taxas de acerto, redes de Super-Resolução podem ser utilizadas.

3.3 Reconhecimento Facial com Super-Resolução

Vários trabalhos demonstram que a baixa resolução em imagens faciais causam uma redução muito significativa nas taxas de reconhecimento. Lemieux e Parizeau [61] mostraram redução na taxa de reconhecimento baseado em *eigenface* (*Principal Component Analysis* (PCA)) quando a distância entre os olhos é próxima a 7 pixels. Lin [62] e Fookes et al. [63] testaram a performance de PCA e *Elastic Bunch Graph Matching* para distâncias entre os olhos de 111, 56, 28, 14 e 7 pixels, e observaram uma queda severa na performance conforme a distância entre os olhos diminuia. Hennings-Yeomans et al. [64] propôs três estratégias para contornar essa problemática: (i) reduzir as imagens do banco de dados para uma resolução compatível com as imagens de entrada; (ii) aplicar Super-Resolução na imagem de entrada para aumentar a imagem até uma resolução compatível com as imagens no banco de dados; (iii) aplicar simultaneamente Super-Resolução e extração de características biométricas.

Shermeyer e Etten[47] utilizaram SR para melhoria na detecção de objetos em imagens de satélite. Os dois métodos de SR utilizados foram VDSR e o *Random-Forest Super-Resolution* - criado pelos autores nesse trabalho para comparar um método simples e leve com uma rede convolucional estado da arte - em conjunto com os métodos de detecção YOLT e SSD do *framework* SIMRDWN. Utilizando o *dataset* xView, os autores conseguiram melhorias na detecção entre 13% e 36%.

Já Rast et al. [65] utilizaram da SRCNN para aumentar a resolução das imagens - extraídas de imagens de alta resolução utilizando Haar Cascade e depois reduzidas em quatro vezes o tamanho original, de 60×60 pixels para 15×15 - separadamente do reconhecimento, realizado neste trabalho pelos algoritmos HMM e SVD. Para tal, utilizaram os *datasets* Essex, HP, FERET e iCV-F, esse ultimo criado pelos autores, num total de 18.651 imagens, e conseguiram melhorias no reconhecimento de até 46.96% em

relação ao reconhecimento feito diretamente em imagens de baixa resolução.

No trabalho de Wang et al. [66], foram criadas quatro redes convolucionais diferentes, testadas para reconhecimento facial no *dataset* UCSS, atingindo o resultado de 59.03% de acerto no melhor modelo.

Apesar do método de treinamento mais utilizado se dar pela geração artificial de imagens de baixa resolução a partir de imagens de alta resolução, o FR com SR possui taxas de acerto mais baixas em imagens nativamente de baixa resolução, conforme mostrado por Cheng et al. [5]. Outro problema relacionado aos sistemas de FR é que quando utilizado em conjunto com redes de SR em datasets nativamente em baixa resolução, o FR tem suas taxas de acerto pioradas, possivelmente pela adição de ruídos e artefatos introduzidos pela SR, tendo sido demonstrado nesse mesmo trabalho.

Tendo em vista essa problemática, os autores propuseram a *Complement-Super-Resolution and Identity* (CSRI), um modelo de rede convolucional que realiza tanto a SR quanto o FR pelo uso de *joint learn* e que possui duas ramificações, sendo a primeira de SR sintética - que utiliza pares de imagens de alta e baixa resolução - e a segunda de SR nativa - utilizando apenas imagens de baixa resolução nativa, de forma a fazer com que a rede adapte o aprendizado dos pares sintéticos para as imagens nativas. A rede por eles testada é uma junção da rede de SR VDSR e da rede de FR CenterFace, treinada com base nos *datasets* CelebA e TinyFace e testada nos *datasets* TinyFace e MegaFace2 (reduzido sinteticamente para baixa resolução).

Para tal, as funções de custo utilizadas foram MSE (Equação (2.11)) para as duas ramificações de SR e Cross Entropy CE (Equação (2.24)) para o FR. Com isso, a função de custo C final segue, conforme a Equação (3.19).

$$C = (CE_{SYN} + CE_{NAT}) + \lambda MSE_{SR} \tag{3.19}$$

Onde CE_{SYN} é o erro do RF na rede sintética, CE_{NAT} é o erro do RF na rede nativa, MSE_{SR} é o erro da SR e λ é um escalar que pondera o erro da SR.

Com isso, os autores conseguiram uma melhoria na taxa de acerto de 12.7% no *dataset*

TinyFace em comparação aos outros métodos, sendo o melhor método utilizado até o presente momento (Junho de 2019) para grandes *datasets*.

3.4 Considerações finais

Neste capítulo foram abordados os temas de *Face Detection*, *Face Recognition*, Super-Resolução, e como essa última pode auxiliar no reconhecimento realizado em baixa resolução. Alguns trabalhos correlatos, técnicas e redes foram também apresentados, e servirão como base para a realização do trabalho proposto.

Capítulo 4

Materiais e Métodos

Após introduzidas as bases teóricas para o trabalho proposto, neste capítulo serão apresentados os materiais utilizados e a metodologia seguida para o atingimento dos objetivos propostos anteriormente. Também serão expostos a rede neural *joint-learning*, *datasets*, metodologias de treinamento e de avaliação do desempenho.

4.1 Materiais

Nesta seção são apresentados os materiais que serão utilizados, como máquinas de desenvolvimento e de treinamento, linguagem de programação, *frameworks*, bibliotecas e *datasets*.

4.1.1 Hardware

Para o desenvolvimento do presente trabalho, foi utilizado um notebook Apple MacBook Pro (15", 2017), equipado com processador Intel Core i7-7700HQ, *quad-core* e oito *threads* com clock de 2.8GHz, 16GB de memória RAM LPDDR3, com clock de 2133 MHz, e placa de vídeo AMD Radeon Pro 555, com 2GB de memória GDDR5.

Como os treinamentos de redes neurais são deveras custosos em termos de poder computacional, uma máquina específica para treinamento das redes faz-se necessária.

Para tal, foram utilizadas duas máquinas. A primeira, uma *workstation* equipada com processador Intel i7-7700K *quad-core* e 8 *threads* @ 4.20GHz, 16GB de memória RAM DDR4, e duas placas de vídeo Nvidia, sendo uma GeForce GTX 1080 Ti (11GB de memória GDDR5X e 3584 CUDA Cores) e uma GeForce GTX 1070 (8GB de memória GDDR5 e 1920 CUDA Cores). No decorrer do trabalho, foi adicionada uma segunda máquina, um servidor equipado com processador AMD EPYC 7351 *16-core* 32 *threads* @ 2.4GHz, 128GB de memória RAM DDR4, e um par de placas de vídeo Nvidia GeForce RTX 2080 Ti (11 GB GDDR6, 4352 CUDA Cores, 544 Tensor Cores).

4.1.2 Software

O notebook de desenvolvimento possui sistema operacional macOS Movaje[1] versão 10.14.5. As máquinas de treinamento possuem sistemas operacionais baseados em Linux, sendo que a *workstation* roda Fedora[2] v30 com *kernel* v5.2.17 e o servidor roda Ubuntu[3] v18.04 com *kernel* v5.4.0.

Python foi a linguagem de programação escolhida, na versão 3.6.9. O ambiente de desenvolvimento utilizado foi o Visual Studio Code[4] rodando dentro de um container Docker[5], para garantir que o código seja o mais reprodutível possível.

Para a criação das *Neural Networks*, o *framework* utilizado foi o TensorFlow[6] ($v2.2$) [67], por possuir uma grande e ativa comunidade de desenvolvedores, ter suporte a processamento nas GPUs com CUDA e ter suporte nativo a API Keras[7], que facilita a prototipagem das redes. Também foram utilizada as bibliotecas OpenCV[8] ($v4.4.0.46$) e scikit-image[9] ($v0.17.2$) para conversão de imagens e pré-processamento das mesmas,

[1] `https://www.apple.com/lae/macos/mojave/`
[2] `https://getfedora.org/`
[3] `https://ubuntu.com/`
[4] `https://code.visualstudio.com/`
[5] `https://www.docker.com/`
[6] `https://www.tensorflow.org`
[7] `https://keras.io/`
[8] `https://opencv.org`
[9] `https://scikit-image.org/`

scikit-learn[10] (v0.23.2) para algoritmos de *Machine Learning*, e scikit-optimize[11] (v0.8.1) para o otimizador bayesiano.

4.1.3 *Datasets*

Conforme descrito anteriormente, o treinamento de redes de reconhecimento de faces necessita de grandes *datasets*, de forma a fazer com que a rede se torne mais generalista, separando melhor as diferentes classes (ou indivíduos), e realizando um melhor agrupamento intra-classe (ou seja, agrupando melhor as diferentes imagens de um mesmo indivíduo).

Para tal, foram selecionados os seguintes datasets:

- CASIA-WebFace;
- *Labeled Faces in the Wild* (LFW);
- TinyFaces;
- VggFace2.

Em 2014, Yi, Lei, Liao et al. [68] propuseram uma metodologia semi-automatizada para criação de grandes *datasets* para reconhecimento de faces. Visto que para realizar treinamentos de reconhecimento de faces são necessários muitos dados, na época grandes *datasets* para esse tipo de treinamento eram apenas privados, como o SFC [69] do Facebook e o CelebFaces [70] da Microsoft. A partir dessa premissa, Yi, Lei, Liao et al. [68] criaram uma metologia de extração semi-automatizada de imagens e labels e montaram o *dataset* CASIA-WebFace, contendo 494.414 imagens de 10.575 celebridades extraídas da base de dados do IMDb[12], que pode ser visto na Figura (4.1). Esse dataset foi escolhido pelo bom balanço entre número de imagens totais e tempo de treinamento, sendo utilizado durante a pesquisa dos *hyperparâmetros* da rede.

[10] `https://scikit-learn.org/stable/`
[11] `https://scikit-optimize.github.io/stable/index.html`
[12] `https://www.imdb.com/`

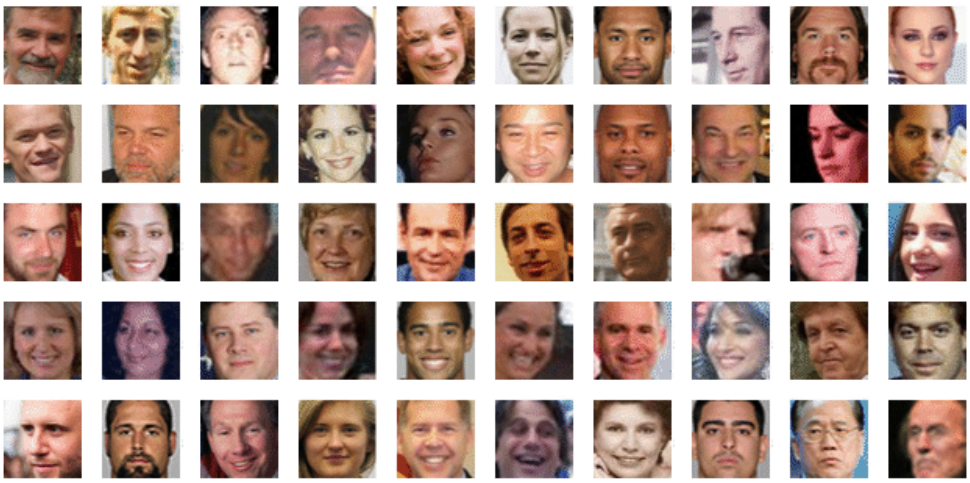


Figura 4.1: Exemplos de imagens do *dataset* CASIA-WebFace. [68]

VggFace2[13] [71] é um *dataset* com grande número de imagens tanto de diferentes indivíduos quanto de um mesmo indivíduo, contando com diferentes poses, idades, etnias, profissões, iluminações e *backgrounds*. Esse *dataset*, que pode ser visualizado na Figura (4.2), possui 3.31 milhões de imagens de 9131 indivíduos (variando entre 80 e 843 imagens de cada, com média de 362.6 imagens por indivíduo), com $59,3\%$ de homens e $40,7\%$ de mulheres, divididas entre 8631 classe na parcela de treinamento e 500 classes na parcela de testes.

VggFace2 foi escolhido para ser o principal *dataset* de treinamento por, no nosso conhecimento, ser o maior dataset aberto em número de imagens e classes, auxiliando no potencial de generalização da rede. Com esse dataset e a rede SE-ResNet-50, os autores conquistaram o status de SOTA nos *benchmarks* IJB [71].

Além do VggFace2, TinyFaces[14] [43] também foi selecionado para o treinamento (mais especificamente para a fase de *fine-tuning* da rede), por ser um *dataset* com imagens nativamente em baixa resolução, como mostra a Figura (4.3). Essas imagens variam entre 6×6 até 32×32 pixels, com média de 20×20, sendo num total 169.403 imagens, divididas entre 15.975 imagens rotuladas de 5.139 indivíduos, e 153.428 imagens não rotuladas. As imagens não rotuladas, em conjunto com as imagens de 2.569 classes, são utilizadas na

[13] `http://www.robots.ox.ac.uk/~vgg/data/vgg_face2/`
[14] `https://qmul-tinyface.github.io/`

Figura 4.2: Exemplos de imagens do *dataset* VggFace2. [71]

parcela do dataset de validação, enquanto as 2.570 classes restantes formam a parcela de treinamento [43].

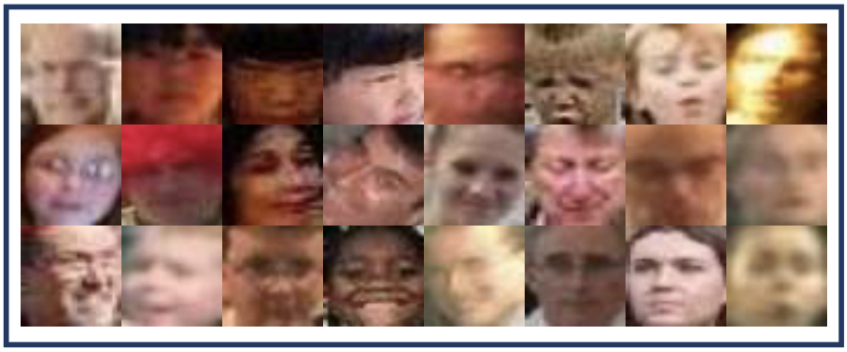


Figura 4.3: Exemplos de imagens em baixa resolução nativa do *dataset* TinyFaces. [5]

Para a validação, foi selecionado o *Labeled Faces in the Wild* (LFW)[15], [72] por ser o mais clássico *dataset* para validação de reconhecimento de faces na literatura. Esse *dataset* foi criado com base no problema de *pair matching*, que é o caso de, dadas duas imagens com rostos, identificar se os rostos pertencem a mesma pessoa ou a pessoas diferentes. Para isso, o LFW conta com 13.233 imagens de 5.749 indivíduos, sendo que desses, 1.680 possuem duas ou mais imagens, enquanto os 4.069 restantes possuem apenas uma imagem. Um exemplo das imagens constantes no *dataset* pode ser visto na Figura (4.4).

[15] `http://vis-www.cs.umass.edu/lfw/`

Figura 4.4: Exemplos de imagens do *dataset* LFW. [72]

4.2 Métodos

Nesta seção são apresentados os métodos que foram seguidos para a realização do trabalho, bem como a metodologia de análise de desempenho.

4.2.1 Arquitetura da rede

Seguindo os achados de Ledig, Theis, Huszar et al. [50] com a rede SRGAN, de Wang, Yu, Wu et al. [43] com a rede ESRGAN, e o proposto por Cheng, Zhu e Gong [5] com o CSRI, arquitetou-se a rede com os seguintes princípios:

- A rede deve ser uma junção entre SR e FR, de forma a ser treinada conjuntamente;
- A rede deve ser treinada nos moldes de uma rede GAN;
- A parcela da rede relativa a SR deve ser uma rede Generadora deGAN;
- A parcela da rede relativa a FR deve ser uma rede CNN;
- A rede Discriminadora deve somente discriminar sobre as imagens de SR geradas pela rede de SR e FR;
- A rede Discriminadora deve ser treinada à parte da rede de SR e FR, i.e., possuir gradientes próprios;

- A rede deve possuir duas ramificações, a primeira que lida com imagens de baixa resolução sintética, e a segunda que lida com imagens de baixa resolução nativa;

A partir dos princípios descritos acima, desenvolveu-se então a rede *Super-Resolution Face Recognition - Generative Adversarial Network* (SRFR-GAN), cuja arquitetura pode ser visualizada na Figura (4.5).

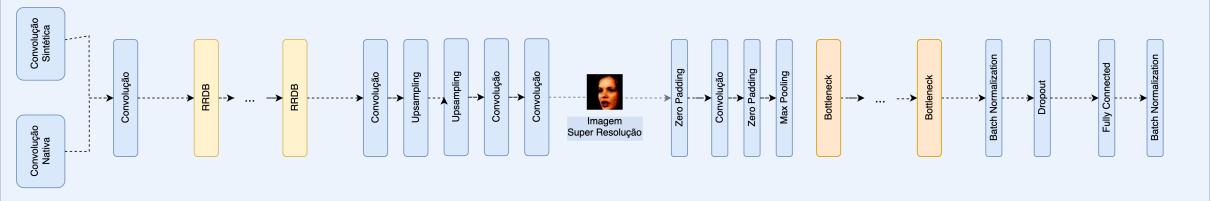


Figura 4.5: Arquitetura da rede SRFR-GAN.

Essa rede conta com, na sua parcela relativa a SR, uma rede Generativa nos moldes apresentados por Wang, Yu, Wu et al. [43], com 23 RRDBs. Cada RRDB é composto por 3 *residual dense blocks*, que são blocos de 5 camadas convolutivas onde ocorrem convoluções de 32 *filters*, com *kernels* de 3×3, *stride* de 1×1, e *padding* de 1. As saídas de cada camada de convolução são concatenadas com as anteriores e com a entrada, realizando assim a adição com o resíduo das entradas. Ao final de cada *residual dense block* e de cada RRDB, as saídas são escaladas pelo parâmetro β de escala de resíduo e então são adicionadas ao resíduo da rede.

Após as camadas de RRDB ocorre mais uma convolução e, em seguida, duas camadas de aumento de resolução intercaladas com convoluções. Por fim, ocorrem mais duas convoluções, gerando assim na saída a imagem de SR, com tamanho de 112×112 pixels. Todas as convoluções nessa parcela da rede seguem o mesmo padrão das convoluções apresentadas anteriormente. Também, ao longo de toda a rede, foi utilizada a função de ativação Mish (Equação (2.17)).

Para a tarefa do reconhecimento das faces, foi utilizada a ResNet 50, tendo na camada de entrada *Zero Padding* de 3×3, seguido por uma camada de convolução com 64 *filters*, *kernel* de 7×7 e *stride* de 2. Em sequência, tem-se uma segunda camada de *Zero*

Padding de 1×1, uma camada de *Max Pooling* de 3×3 e *stride* 2×2, e então as camadas *Bottleneck*. Após as camadas de *Bottleneck*, tem-se uma camada de *Batch Normalization* com *momentum* de 0.9, seguida de *Dropout* com *rate* de 0.4, uma camada totalmente conectada cujas saídas refletem o tamanho do vetor de *embeddings*, e, por fim, uma última camada de *Batch Normalization*, também de *momentum* de 0.9.

A entrada da rede SRFR-GAN é composta por uma camada convolucional de *kernel* 3×3, *stride* 1×1, e 62 *filters*. Essa camada de convolução é duplicada, sendo que uma camada atua somente nas imagens de baixa resolução sintética (convolução sintética), enquanto a outra fica especializada em imagens de baixa resolução nativa (convolução nativa). Dessa forma, a camada de convolução sintética ajuda a treinar a convolução nativa, fazendo com que essa, além de se tornar treinável (já que não existem imagens *ground truth* para as imagens nativamente em baixa resolução), fique especializada em descobrir as características pertinentes a imagens de baixa qualidade nativas.

Essa camada de convolução liga-se com a parcela de SR descrita anteriormente, que gera a primeira saída da rede, i.e., a imagem de resolução aumentada pela rede. Essa imagem propaga-se através da parcela de FR descrita anteriormente, gerando assim a segunda saída da rede, i.e., o vetor de *embeddings* da face contida na dada imagem. O tamanho do vetor de *embeddings* utilizado foi 512, seguindo o proposto na literatura.

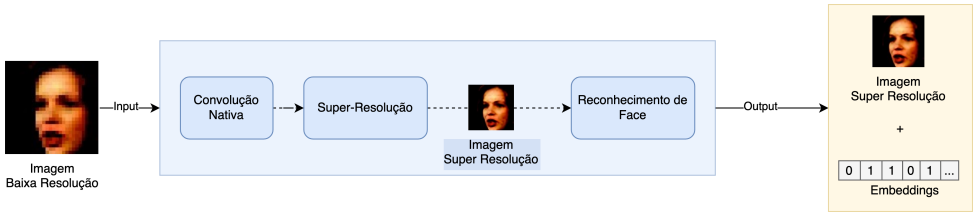


Figura 4.6: Arquitetura da rede SRFR-GAN em inferência.

Destaca-se que as imagens de baixa resolução nativa são apresentadas somente a camada de convolução nativa. De mesmo modo, as imagens de baixa resolução sintética só são apresentadas a camada de convolução sintética. Ambas as camadas são conectadas na primeira camada de convolução da parcela de SR.

Como a camada de convolução sintética serve apenas ao propósito de auxiliar no

treinamento da convolução nativa, em um cenário de inferência, essa camada não faria parte da rede e todos os *inputs* seriam apresentados a camada de convolução nativa, conforme ilustra a Figura (4.6).

4.2.2 Função de Custo

Para realizar o treinamento, extendeu-se a função de custo de Cheng, Zhu e Gong [5], tendo a $L_{SRFR-GAN}$ descrita na Equação (4.1).

$$L_{SRFR-GAN} = (L_{FR}^{SYN} + \eta L_{FR}^{NAT}) + \lambda L_{SR} \tag{4.1}$$

Onde L_{FR}^{SYN} e L_{FR}^{NAT} são a função de custo da parcela de reconhecimento da face aplicadas sobre as imagens de baixa qualidade sintéticas (SYN) e nativas (NAT) respectivamente. η e λ são escalares que ponderam os termos do reconhecimento de faces nativa e da super-resolução, respectivamente.

Para L_{FR}, testou-se duas possibilidades, a primeira é a ArcLoss (Equação (3.16)), a segunda é a função de *Cross-Entropy* (CE) (Equação (2.24)) aplicada na função Softmax σ (Equação (3.14)). Para $L_{FR} := CE(\cdot)$, foi adicionada uma camada totalmente conectada após a saída da rede, que tem como entrada o vetor de *embeddings* e como saída a classe a qual o determinado rosto pertence. Ou seja, a entrada dessa camada é 512, e a saída é o número de classes do dataset de treinamento. Com a função Softmax aplicada nessa camada, obtem-se a probabilidade de cada uma das saídas ser a classe que melhor representa o rosto que consta na imagem. A partir disso, o erro foi calculado utilizando a CE aplicada na saída da Softmax.

Já a função de custo L_{SR}, foi seguido o proposto por Wang, Yu, Wu et al. [43], conforme descrito na Equação (3.9), alterando apenas alguns escalares, como descrito na Equação (4.2).

$$L_G = \alpha L_{percep} + \beta L_G^{Ra} + \gamma L_1 \tag{4.2}$$

Onde L_{percep} é função de custo perceptual com base nas características extraídas da

rede VGG19 (conforme descrito na Equação (3.5)), L_G^{Ra} é a função de custo da rede generativa no seu modelo *Relativistic Average Generative Adversarial Network* (RaGAN) (conforme descrito na Equação (3.8)), e L_1 é a função de *1-norm distance* entre a imagem em alta resolução a imagem de resolução aumentada (conforme descrito na Equação (3.10)). α, β e γ são escalares que ponderam os três termos da equação, ajustando conforme o necessário para o melhor aprendizado.

Para a função de custo da rede Discriminadora, foi aderido o proposto por Wang, Yu, Wu et al. [43], conforme descrito na Equação (3.7).

4.2.3 Análise de desempenho

Para analisar o desempenho do *Super-Resolution* foi escolhida a métrica de *Peak Signal-to-Noise Ratio* (PSNR). Essa métrica, que é baseada no *Mean Squared Error*, demonstra a razão entre a magnitude do sinal e a magnitude do ruído que afeta a qualidade da imagem. Sua fórmula para imagens de apenas um canal de cor, ou seja, imagens em preto e branco, está descrita na Equação (4.3), onde a métrica para uma imagem g é calculada com base em uma imagem de referência f, ambas matrizes de $M \times N$ [73].

$$PSNR(f,g) = 10\log_{10}\left(\frac{255^2}{MSE(f,g)}\right) \tag{4.3}$$

$$MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij} - g_{ij})^2 \tag{4.4}$$

Para as análises de desempenho do *Face Recognition* foram escolhidas as seguintes métricas:

- ***Accuracy***: Mensura a habilidade da rede em classificar corretamente os exemplos. Tem sua fórmula como $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, onde TP são os verdadeiros positivos, TN são os verdadeiros negativos, FP e FN são os falsos positivos e os falsos negativos, respectivamente.

- ***Area Under Curve (AUC)***: Essa métrica mostra quão bem o modelo é capaz de distinguir entre as classes. Quanto mais alto seu valor (quanto mais o valor se aproxima de 1) maior é a habilidade do classificador em predizer Positivos como Positivos, e Negativos como Negativos.

- ***False Alarm Ratio (FAR)***: Mede a quantidade de vezes que a rede classificou um exemplo erroneamente como positivo, isto é, é a razão de falsos positivos do classificador. Sua fórmula se dá por $FAR = \frac{FP}{FP+TN}$, onde FP são os falsos positivos, e TN são os verdadeiros negativos.

- ***Equal Error Rate (EER)***: Mede a taxa de erro do classificador quando o FAR e o *False Rejection Rate (FRR)* - que mensura a quantidade de vezes que a rede classificou um exemplo erroneamente como negativo - possuem mesmo valor, na tentativa de minimizar ambas as taxas de erro.

- ***Validation Rate***: Outra métrica de *Accuracy*, mostra quão bem a rede classifica os exemplos enquanto mantém $FAR = 1 * 10^{-3}$.

Para o dataset LFW utilizou-se a validação como um problema binário, onde duas imagens com rostos são apresentadas para a rede, a fim de que a mesma classifique esses rostos como de uma mesma pessoa ou de pessoas diferentes. As saídas da rede, isto é, os vetores de *embedding* gerados pela mesma, passam por uma função de distância euclidiana, e, a partir de um determinado *threshold*, consegue-se classificar os rostos. Como o valor do *threshold* é uma escolha arbitrária, diferentes *thresholds* são utilizados, e por isso são calculadas as médias dos valores das métricas descritas anteriormente. Com isso, tem-se também os valores do Desvio Padrão de *Accuracy* e de *Validation Rate*.

4.2.4 Pré-Processamento

Antes de iniciar o treinamento, os *datasets* passaram por pré-processamento, que consistiu em detecção das faces, alinhamento das mesmas, e redução da resolução das imagens, de forma a gerar os datasets de baixa resolução sintética.

Para a detecção das faces e geração dos pontos para alinhamento, utilizamos a rede MTCNN[16] [74] implementação[17] no TensorFlow. Essa rede foi escolhida por ser o padrão utilizado na literatura [60], [75], [76]. Após a detecção, para as imagens que continham mais de uma face, utilizamos a métrica de distância do ponto médio do *bounding box* do rosto em relação ao centro da imagem, e selecionamos o rosto que mais se aproximava do ponto central da mesma.

Para o alinhamento, utilizamos os 5 pontos faciais (os dois olhos, nariz, e dois cantos da boca) e aplicamos funções do scikit-image e do OpenCv, realizando então o alinhamento da face e o *crop* da região do rosto. Para o *crop* das imagens em alta resolução, padronizamos em 112×112 pixels. A partir desse *crop* temos as imagens de *ground truth* em alta resolução para o treinamento.

Durante a fase de detecção e alinhamento das faces, foram encontrados problemas quanto a vazamentos de memória no TensorFlow, que impossibilitavam a paralelização do pré-processamento a fim de realizar essa operação em tempo hábil em datasets grandes, como o caso do VggFace2. Como o TensorFlow, por padrão, aloca praticamente toda a memória da GPU durante sua inicialização[18] e somente desaloca quando o processo é finalizado (ao invés de desalocar após a realização da operação de inferência sobre a imagem), alguns ponteiros não eram capturados pelo *Garbage Collector* do Python e sobravam na memória RAM, fazendo com que o uso dessa crescesse conforme fosse realizado o pré-processamento até ser totalmente utilizada, chegando ao ponto de receber o sinal SIGKILL por parte do sistema operacional por falta de memória. Para superar essa dificuldade, as operações de pré-processamento foram realizadas em *batches* de 256 imagens, realizando a operação toda em processos filhos do programa principal. Esses processos, após a execução do algoritmo, não são reaproveitados, mas sim terminados, e então novos processos são instanciados com o novo *batch* para realizar a operação. Com isso, foi assegurado a limpeza da memória RAM, não possibilitanto o estouro do máximo de sua capacidade, ao mesmo tempo que paralelizamos o pré-processamento para realizá-lo

[16] `https://kpzhang93.github.io/MTCNN_face_detection_alignment/`
[17] `https://github.com/ipazc/mtcnn`
[18] `https://www.tensorflow.org/guide/gpu`

em tempo hábil.

Com as imagens já alinhadas e *croppadas*, ocorreu então a redução da resolução das mesmas para 28×28 pixels utilizando funções do OpenCv com a interpolação bicúbica. Essa resolução foi escolhida por ser exatamente 4 vezes menor que as imagens de *ground truth*. Essas imagens foram salvas no formato TFRecords[19] para aumento de performance no treinamento[20].

Antes de realizar o treinamento, foi realizada a filtragem das classes que apareciam em ambos os dataset de treinamento e de validação[21]. Depois, realizou-se o *flip* das imagens como forma de *data augmentation*, para que a rede se torne mais generalista, e com isso o tamanho dos datasets foi dobrado. Por fim, a partir da matriz da imagem com valores inteiros entre [0, 255], foi realizada a normalização de seus valores, subtraindo 125,7 e dividindo por 128, para que cada valor da matriz se encontre entre (-1, 1), conforme sugerido na literatura [60], [75], [76].

4.2.5 Treinamento

O processo de treinamento ocorre em duas fases.

Na primeira, a rede é treinada somente com imagens de baixa resolução sintética - aproveitando do fato dessas imagens terem sua contrapartida em alta resolução - de modo a fazer com que a rede aprenda a mapear uma imagem de baixa resolução em alta resolução apropriadamente.

Na segunda, são apresentadas a rede ambas as imagens de baixa resolução nativa e sintética, de forma a realizar o processo de *fine-tuning*. Com isso, pela camada de convolução nativa, a rede tende a aprender a mapear as características relevantes que somente são encontradas em imagens nativamente em baixa resolução. Essa arquitetura de treinamento pode ser visualizada na Figura (4.7).

Em ambas as fases uma camada *Fully Connected* é adicionada após a geração dos

[19] `https://www.tensorflow.org/tutorials/load_data/tfrecord`
[20] `https://www.tensorflow.org/guide/data_performance`
[21] `https://github.com/happynear/FaceDatasets`

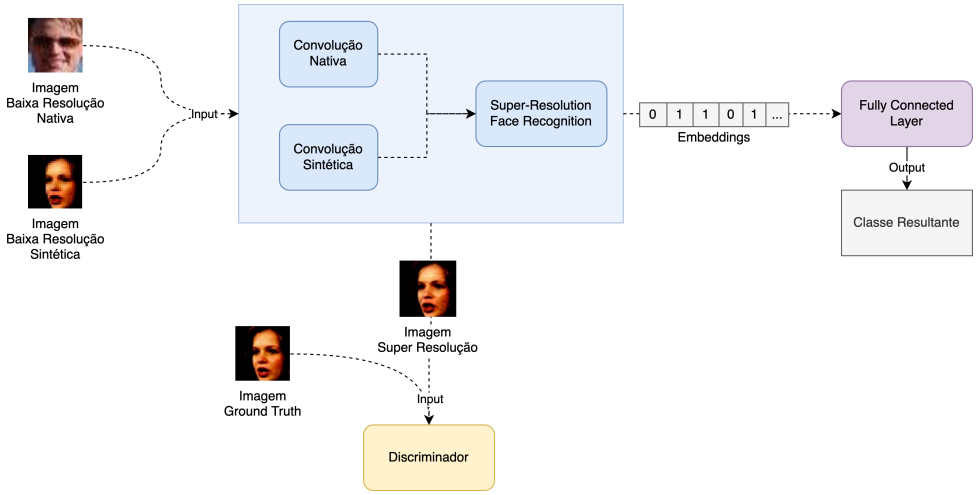


Figura 4.7: Arquitetura do treinamento da rede SRFR-GAN

embeddings. Essa camada tem como entrada o próprio vetor de *embeddings* e como saída o número de classes do treinamento, que, após passar pela função de ativação Softmax, dará como output a probabilidade de ser cada uma das classes presentes no *dataset*. O resultado dessa camada totalmente conectada serve então como *input* para a função de custo durante o treinamento.

Os experimentos realizados junto com seus resultados, seguindo a metodologia exposta, são apresentados no capítulo seguinte.

Capítulo 5

Experimentos

Após o pré-processamento foram realizados os treinamentos utilizando a técnica de *mixed-precision*[1], de forma a reduzir o tamanho das redes dentro das GPUs e possuir mais espaço para um *batch size* maior a cada iteração. Essa técnica possibilitou dobrar o *batch size*, passando de 8 para 16 imagens por GPU.

No início dos treinamentos foram enfrentados, mais uma vez, problemas relacionados ao uso exacerbado de memória, pois a memória RAM não comportava alocar todo o dataset. Esse problema foi resolvido utilizando métodos de *caching* do *dataset* em HD.

Nos primeiros testes foram utilizados o otimizador Adam e ArcLoss como função de custo correspondente a parcela de FR, com os hyperparâmetros conforme Tabela (5.1).

Os resultados, que podem ser vistos na Tabela (5.2) e Figura (5.1), mostram que a rede não aprendeu a classificar bem diferentes rostos, acertando 50% das faces apresentadas no LFW por pura sorte, já que em uma validação binária - como a utilizada - se a rede simplesmente "chutar" todas as faces como "mesma pessoa" ou como "pessoas diferentes" a mesma acertará na metade das vezes.

Nesse momento, foram descobertos erros quanto ao treinamento distribuído no TensorFlow. Como a função de treinamento utilizada é customizada, as agregações e divisões por *batch size* realizadas na função de custo do treinamento devem ser escritas pelo usuário do *framework*, não existindo ainda um tratamento automatizado para esse caso de uso

[1] `https://www.tensorflow.org/guide/mixed_precision`

ID	*Learning Rate*	*Momentum*	*Weight Decay*	λ	η	α	β	γ	*Scale*	*Margin*
01	0,001	0,9	0,005	0,100	1,000	0,100	1,000	0,100	64	0,5
02	0,0001	0,99	0,005	0,100	1,000	0,100	1,000	0,100	64	0,5

Tabela 5.1: Hyperparâmetros utilizados no treinamento da rede com otimizador Adam. λ corresponde a *Super-Resolution Weight*, η a *Face Recognition Weight*, α a *Perceptual Weight*, β a *Generator Weight*, e γ a L_1 *Distance Weight*. Os dois últimos parâmetros são, respectivamente, *ArcLoss Scale*, *ArcLoss Angular Margin*.

Figura 5.1: Exemplos de imagens de resolução aumentada geradas pela rede no treinamento que utilizou ArcLoss na função de custo e otimizador Adam. As imagens são do dataset de validação *Labeled Faces in the Wild*.

ID do Treinamento	Accuracy Mean	Accuracy Std.	AUC	EER	FAR	Validation Rate	Validation Std.
01	50,01	0,025	0,500	0,500	1,000	1,000	0
02	50,01	0,025	0,500	0,500	1,000	1,000	0

Tabela 5.2: Acurácia do treinamento, utilizando otimizador Adam e ArcLoss na função de custo, em relação ao dataset *Labeled Faces in the Wild*.

da ferramenta. A documentação apontava para utilizar a função "tf.nn.compute_average_loss", mas os valores de erro da rede eram muito mais altos do que quando comparado com o treinamento utilizando apenas uma GPU, pois a função primeiro soma todos os valores do tensor da imagem que tem forma [*batch size*, altura, largura, profundidade] e depois divide esse resultado pelo *batch size* utilizado. O comportamento esperado (que é o padrão quando utilizado com apenas uma GPU) é o de realizar a média dos valores

do tensor da imagem, e não a soma. O erro foi resolvido fazendo a agregação dos valores do tensor utilizando a função "tf.math.reduce_mean" e dividindo então pelo número de GPUs utilizadas no treinamento, já que os gradientes gerados após a fase *backwards* são somados para atualizar os pesos da rede.

Após a resolução do erro nas funções de custo, o otimizador foi trocado de Adam para NovoGrad (buscando melhorar a performance do treinamento e poder de generalização da rede) e a função de ativação para a Mish (buscando um melhor poder de regularização da rede), além de testar duas redes em paralelo, uma utilizando camadas de *Batch Normalization* na parcela de reconhecimento da face, e a outra sem as mesmas camadas. Esse teste parte da possibilidade levantada por Wang, Yu, Wu et al. [43] que redes de SR com camadas de BN geravam imagens com menor nível de detalhamento, e queríamos testar se o mesmo era válido para a arquitetura proposta. Os hyperparâmetros utilizados são mostrados na Tabela (5.3) e as imagens de resolução aumentada são mostradas na Figura (5.2).

ID	*Learning Rate*	β_1	β_2	*Weight Decay*	λ	η	α	β	γ	*Scale*	*Margin*	BN
03	0,001	0,9	0,999	0	0,001	1	0,100	1,000	0,100	64	0,5	Sim
04	0,001	0,9	0,999	0	0,001	1	0,100	1,000	0,100	64	0,5	Não
05	0,004	0,9	0,9	0,005	0,001	1	0,100	1,000	0,100	64	0,5	Sim
06	0,004	0,9	0,9	0,005	0,001	1	0,100	1,000	0,100	64	0,5	Não
07	0,001	0,4	0,999	0,005	0,001	1	0,100	1,000	0,100	64	0,5	Sim
08	0,001	0,4	0,999	0,005	0,001	1	0,100	1,000	0,100	64	0,5	Não

Tabela 5.3: Hyperparâmetros utilizados no treinamento da rede com otimizador NovoGrad e ArcLoss na função de custo. β_1 e β_2 são os parâmetros do otimizador, λ corresponde a *Super-Resolution Weight*, η a *Face Recognition Weight*, α a *Perceptual Weight*, β a *Generator Weight*, e γ a L_1 *Distance Weight*. Os parâmetros *Scale* e *Margin* são, respectivamente, *ArcLoss Scale*, *ArcLoss Angular Margin*.

Pode-se notar pelas imagens que, por mais que a rede tenha aprendido o formato do rosto de modo geral, algumas partes do rosto ainda estão desfiguradas, além das imagens possuírem muitos artefatos.

Os resultados de acuracidade no dataset LFW estão descritos na Tabela (5.4).

A partir da não convergência da rede e da análise dos gráficos das funções de custo, decidiu-se simplificar a função de custo na parcela de reconhecimento da face, na tentativa

Figura 5.2: Exemplos de imagens do treinamento utilizando ArcLoss na função de custo e otimizador NovoGrad. Na esquerda, a imagem de baixa resolução; ao centro, a imagem *ground truth* em alta resolução; na direita, a imagem de resolução aumentada gerada pela rede.

ID do Treinamento	*Mean Accuracy*	*Accuracy Std.*	AUC	EER	FAR	*Validation Rate*	*Validation Std.*
03	–	–	–	–	–	–	–
04	50,01	0,025	0,500	0,500	1,000	1,000	0
05	–	–	–	–	–	–	–
06	50,01	0,025	0,500	0,500	1,000	1,000	0
07	–	–	–	–	–	–	–
08	50,01	0,025	0,500	0,500	1,000	1,000	0

Tabela 5.4: Acurácia do treinamento, utilizando otimizador NovoGrad e ArcLoss na função de custo, em relação ao dataset *Labeled Faces in the Wild*. As linhas nulas representam treinamentos interrompidos pela não convergência da rede.

de buscar uma convergência facilitada. Para tal, a ArcLoss foi substituída pela *Cross-Entropy*, e os treinamentos foram iniciados utilizando os hyperparâmetros da Tabela (5.5).

Nos treinamentos 11 ao 14, as camadas de *upsampling* foram trocadas de de UpSampling2D, seguidas de Conv2D, para a Conv2DTranspose.

A camada de UpSampling2D realiza o esticamento da imagem utilizando algum algoritmo de interpolação. Para esse algoritmo foi utilizado o *nearest neighbours*, que, na

ID	*Learning Rate*	*Learning Rate Decay Steps*	β_1	β_2	*Weight Decay*	λ	η	α	β	γ	BN	# Iterações
09	0,007	–	0,9	0,25	0,002	0,001	1	0,100	1,000	0,100	Sim	$18k$
10	0,007	–	0,9	0,25	0,002	0,001	1	0,100	1,000	0,100	Não	$56k$
11	0,007	1000	0,9	0,25	0,002	0,001	1	0,100	1,000	0,100	Sim	$50k$
12	0,007	1000	0,9	0,25	0,002	0,001	1	0,100	1,000	0,100	Não	$54k$
13	0,007	1000	0,9	0,25	0,002	0,1	1	0,001	0,050	0,010	Sim	$--$
14	0,007	1000	0,9	0,25	0,002	0,1	1	0,001	0,050	0,010	Não	$40k$
15	0,007	1000	0,5	0,25	0,002	0,1	0,1	0,001	0,050	0,010	Sim	$12k$
16	0,007	1000	0,5	0,25	0,002	0,1	0,1	0,001	0,050	0,010	Não	$9k$
17	0,002	1000	0,5	0,25	0,002	0,1	0,1	0,001	0,050	0,010	Sim	$--$
18	0,002	1000	0,5	0,25	0,002	0,1	0,1	0,001	0,050	0,010	Não	$16k$

Tabela 5.5: Hyperparâmetros utilizados no treinamento da rede com otimizador NovoGrad e *Cross-Entropy* na função de custo. β_1 e β_2 são os parâmetros do otimizador, λ corresponde a *Super-Resolution Weight*, η a *Face Recognition Weight*, α a *Perceptual Weight*, β a *Generator Weight*, e γ a L_1 *Distance Weight*. As linhas nulas na coluna de Número de Iterações são treinamentos que ocorreram erros e, por isso, foram desconsiderados.

Figura 5.3: Exemplos de imagens do treinamento utilizando *Cross-Entropy* na função de custo e otimizador NovoGrad. Na esquerda, a imagem de baixa resolução; ao centro, a imagem *ground truth* em alta resolução; na direita, a imagem de resolução aumentada gerada pela rede.

prática, duplica cada linha e coluna da imagem. Quando seguida de uma convolução essa camada pode ser utilizada em redes GAN, pois o UpSampling2D atuará aumentando a resolução da imagem e a convolução se encarregará de interpretar a imagem aumentada

e capturar os detalhes da mesma. Já a camada Conv2DTranspose realiza a operação transposta da convolução, fazendo com que a imagem aumente dessa forma.

Figura 5.4: Exemplos de imagens do treinamento utilizando Conv2DTranspose nas camadas de *upsampling*.

Foi testada a utilização dessa camada como forma de tentar uma melhoria perceptual na qualidade das imagens geradas pela parcela de SR, mas, conforme pode ser visto na Figura (5.4), a alteração dessa camada trouxe um maior número de defeitos nas imagens, principalmente pontos pretos circundado por artefatos. Possivelmente, após treinamentos mais longos com maior grau de *fine-tuning* na rede, utilizar a Conv2DTranspose ao invés de UpSampling2D seguida por convolução pode gerar imagens com maior nível de detalhamento, mas não é algo que se demonstrou verdadeiro nos testes realizados utilizando poucas iterações/épocas.

Os resultados, que podem ser vistos na Tabela (5.6) e Figura (5.5), mostram que a rede não aprendeu a classificar bem diferentes rostos, com exceção do treinamento número 13, que apresentou um começo de convergência na classificação de rostos, mas ainda assim com resultados pouco expressivos.

Após essa série de treinamentos sem convergência, decidiu-se então testar a utilização de um otimizador bayesiano para auxiliar na busca pelos melhores hyperparâmetros.

Dada uma função $y = f(x)$, *black-box* e custosa de ser calculada (como são as caracterísicas de uma *Neural Network*), o otimizador bayesiano constrói um modelo probabilístico

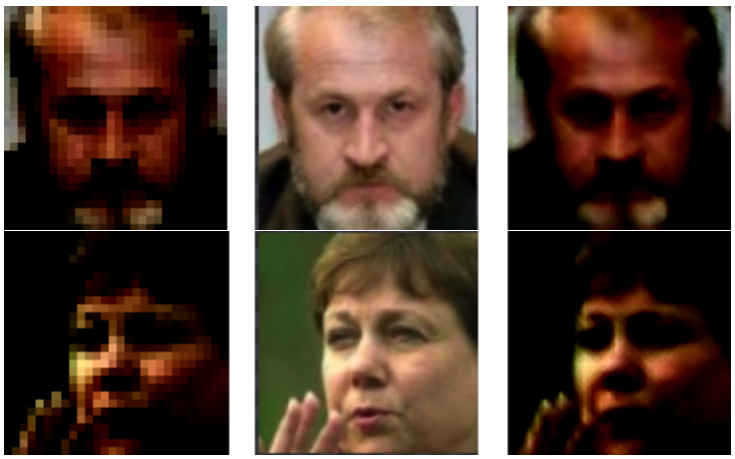


Figura 5.5: Exemplos de imagens de validação utilizando *Cross-Entropy* na função de custo e otimizador NovoGrad. Na esquerda, a imagem de baixa resolução; ao centro, a imagem *ground truth* em alta resolução; na direita, a imagem de resolução aumentada gerada pela rede.

ID do Treinamento	*Mean Accuracy*	*Accuracy Std.*	AUC	EER	FAR	*Validation Rate*	*Validation Std.*
9	50,16	0,952	0,504	0,497	0	0	0
10	50,01	0,025	0,500	0,500	1,000	1,000	0
11	50,01	0,025	0,500	0,500	1,000	1,000	0
12	–	–	–	–	–	–	–
13	51,07	3,270	0,510	0,491	0,587	0,608	0,032
14	–	–	–	–	–	–	–
15	50,01	0,025	0,500	0,500	1,000	1,000	0
16	50,01	0,025	0,500	0,500	1,000	1,000	0
17	–	–	–	–	–	–	–
18	50,01	0,025	0,500	0,500	1,000	1,000	0

Tabela 5.6: Acurácia do treinamento, utilizando otimizador NovoGrad e *Cross-Entropy* na função de custo, em relação ao dataset *Labeled Faces in the Wild*. As linhas nulas são treinamentos que não convergiram ou que ocorreram erros, e, por isso, foram desconsiderados.

a priori que tenta capturar o comportamento da função. Baseando-se na observação do resultado y dessa função, o otimizador então atualiza esse modelo probabilístico utilizando-se de um processo Gaussiano, determinando o próximo conjunto de entradas x que serão processadas pela função, tentando assim convergir mais facilmente a um ótimo global [77].

Para a implementação desse otimizador foi utilizada a função "gp_minimize" da biblioteca *scikit-optimize*. Os hyperparâmetros buscados foram os seguintes:

- *Learning Rate*;

- *Learning Rate Decay Steps*;
- β_1;
- *Face Recognition Weight* λ;
- *Super-Resolution Weight* η;
- *Perceptual Weight* α;
- *Generator Weight* β;
- L_1 *Weight* γ.

O resultado da rede a ser otimizado pela função é a acurácia da rede em detectar faces. β_2 foi travado em 0.999, e os demais parâmetros, que foram encontrados pelo otimizador, podem ser vistos na Tabela (5.7).

ID	*Learning Rate*	*Learning Rate Decay Steps*	β_1	λ	η	α	β	γ	BN	# Iterações	*Mean Accuracy*
19	$0,00354$	$1,52 \cdot 10^3$	$0,724$	$0,202$	$0,506$	$0,00136$	$0,021$	$0,0103$	Sim	$20k$	0
20	$0,00164$	$1,58 \cdot 10^3$	$0,659$	$0,32$	$0,437$	$0,00944$	$0,0132$	$0,0116$	Sim	$20k$	0
21	$0,00398$	$4,56 \cdot 10^3$	$0,779$	$0,945$	$0,127$	$0,0012$	$0,0114$	$0,0417$	Sim	$20k$	0
22	$0,00468$	$4,12 \cdot 10^3$	$0,799$	$0,135$	$0,128$	$0,00579$	$0,0428$	$0,0181$	Sim	$20k$	0
23	$0,00333$	$2,79 \cdot 10^3$	$0,52$	$0,182$	$0,682$	$0,00146$	$0,0232$	$0,0772$	Sim	$20k$	0
24	$0,00501$	$3,13 \cdot 10^3$	$0,558$	$0,155$	$0,5$	$0,00972$	$0,052$	$0,0389$	Sim	$20k$	0
25	$0,0049$	$2,44 \cdot 10^3$	$0,503$	$0,402$	$0,725$	$0,00148$	$0,0486$	$0,0587$	Sim	$20k$	0
26	$0,00297$	$2,62 \cdot 10^3$	$0,646$	$0,381$	$0,707$	$0,00332$	$0,0145$	$0,0569$	Não	$3,5k$	0
27	$0,00627$	$4,53 \cdot 10^3$	$0,813$	$0,493$	$0,857$	$0,00199$	$0,0158$	$0,0686$	Não	$20k$	0
28	$0,00287$	$2,66 \cdot 10^3$	$0,514$	1	$0,103$	$0,00292$	$0,0609$	$0,0132$	Não	$20k$	0
29	$0,00221$	$1,27 \cdot 10^3$	$0,536$	$0,114$	$0,341$	$0,00733$	$0,012$	$0,0239$	Não	$20k$	0
30	$0,00404$	$1,92 \cdot 10^3$	$0,86$	$0,239$	$0,14$	$0,00132$	$0,0264$	$0,0584$	Não	$20k$	0
31	$0,00251$	$4,2 \cdot 10^3$	$0,515$	$0,106$	$0,353$	$0,00794$	$0,0405$	$0,0218$	Não	$20k$	0
32	$0,0022$	$2,5 \cdot 10^3$	$0,853$	$0,12$	$0,961$	$0,00456$	$0,017$	$0,015$	Não	$20k$	0

Tabela 5.7: Hyperparâmetros encontrados pelo otimizador bayesiano no treinamento da rede com otimizador NovoGrad, *Cross-Entropy* na função de custo, e dataset VggFace2. β_1 corresponde ao parâmetro do otimizador NovoGrad, λ corresponde a *Super-Resolution Weight*, η a *Face Recognition Weight*, α a *Perceptual Weight*, β a *Generator Weight*, e γ a L_1 *Distance Weight*. As linhas nulas na coluna de Número de Iterações são treinamentos que ocorreram erros e, por isso, foram desconsiderados.

Após a não convergência da rede, tendo sua acurácia sempre em 0, decidiu-se trocar o dataset VggFace2 pelo CASIA-WebFace, por ser um dataset menor, fazendo com que o treinamento fosse realizado em menor tempo, possibilitando assim acelerar o processo

ID	*Learning Rate*	*Learning Rate Decay Steps*	β_1	λ	η	α	β	γ	BN	# Iterações	*Mean Accuracy*
33	$0,00354$	$1,52 \cdot 10^3$	$0,724$	$0,202$	$0,506$	$0,00136$	$0,021$	$0,0103$	Sim	$20k$	0
34	$0,00164$	$1,58 \cdot 10^3$	$0,659$	$0,32$	$0,437$	$0,00944$	$0,0132$	$0,0116$	Sim	$20k$	0
35	$0,00468$	$4,12 \cdot 10^3$	$0,799$	$0,135$	$0,128$	$0,00579$	$0,0428$	$0,0181$	Sim	$20k$	0
36	$0,00333$	$2,79 \cdot 10^3$	$0,52$	$0,182$	$0,682$	$0,00146$	$0,0232$	$0,0772$	Sim	$20k$	0
37	$0,00501$	$3,13 \cdot 10^3$	$0,558$	$0,155$	$0,5$	$0,00972$	$0,052$	$0,0389$	Sim	$20k$	0
38	$0,0049$	$2,44 \cdot 10^3$	$0,503$	$0,402$	$0,725$	$0,00148$	$0,0486$	$0,0587$	Sim	$20k$	0
39	$0,0024$	$2,6 \cdot 10^3$	$0,722$	$0,722$	$0,707$	$0,00192$	$0,0656$	$0,049$	Não	$25k$	0

Tabela 5.8: Hyperparâmetros encontrados pelo otimizador bayesiano no treinamento da rede com otimizador NovoGrad, *Cross-Entropy* na função de custo, e dataset CASIA-WebFace. β_1 corresponde ao parâmetro do otimizador NovoGrad, λ corresponde a *Super-Resolution Weight*, η a *Face Recognition Weight*, α a *Perceptual Weight*, β a *Generator Weight*, e γ a L_1 *Distance Weight*. As linhas nulas na coluna de Número de Iterações são treinamentos que ocorreram erros e, por isso, foram desconsiderados.

de *hyperparameter tuning*. Os valores dos parâmetros dos novos treinamentos podem ser vistos na Tabela (5.8).

Após mais uma série de treinamentos sem convergência, optou-se por separar as redes, realizando primeiro o treinamento da parcela de *Super-Resolution*, depois da parcela de *Face Recognition*, e, por fim, realizar o *fine tuning* da rede em *joint-learn*. Essa escolha veio da necessidade de identificar de onde partia o problema de convergência.

Também, após avaliação dos resultados gerados nos treinamentos e dos gráficos do erro da rede, percebeu-se que a presença de camadas de BN auxiliou a rede na tentativa de convergência - já que a rede proposta é deveras profunda - sendo que sem essas camadas a rede por vezes seguia na direção oposta à convergência, gerando assim imagens de mais baixa qualidade quando comparado as imagens geradas pelas redes com camadas de BN.

Com isso, iniciaram-se os treinamentos da parcela de SR, com os hyperparâmetros conforme Tabela (5.9), sendo que o primeiro treinamento deu-se seguindo os hyperparâmetros apontados por Wang, Yu, Wu et al. [43]. Vale salientar que nesse treinamento abdicou-se do uso de *weight decay*, de forma a facilitar a descoberta dos melhores hyperparâmetros pelo otimizador bayesiano. Também, trocou-se o otimizador de NovoGrad para Adam com *weight decay decoupled*.

A rede chegou a resultados convincentes no treinamento 43, como os exemplificados na Figura (5.6), com convergência relativamente rápida e com poucos artefatos.

Logo após, deu-se início aos treinamentos da parcela de reconhecimento de face. Para

ID	*Learning Rate*	β_1	α	β	γ	# Iterações
40	$0,0002$	$0,9$	1	$0,005$	$0,01$	$62k$
41	$0,00349$	$0,811$	$0,0146$	$0,0336$	$0,0717$	$47k$
42	$0,0002$	$0,9$	$0,0146$	$0,005$	$0,01$	$34k$
43	$0,0002$	$0,9$	$0,0146$	$0,005$	$0,02$	$50k$
44	$0,00252$	$0,9$	$0,0392$	$0,00506$	$0,0571$	$22k$
45	$0,00300$	$0,9$	$0,0400$	$0,00500$	$0,0580$	$33k$

Tabela 5.9: Hyperparâmetros encontrados pelo otimizador bayesiano no treinamento da rede de super-resolução, com otimizador Adam com *weight decay decoupled*, e dataset CASIA-WebFace. β_1 corresponde ao parâmetro do otimizador Adam, α a *Perceptual Weight*, β a *Generator Weight*, e γ a L_1 *Distance Weight*.

(a) PSNR: 19.89

(b) PSNR: 37.95

(c) PSNR: 15.78

Figura 5.6: Exemplos de imagens de validação do treinamento isolado de *Super-Resolution*. Na esquerda, a imagem de baixa resolução; ao centro, a imagem *ground truth* em alta resolução; na direita, a imagem de resolução aumentada gerada pela rede.

tal, utilizou-se também do otimizador Adam com *weight decay decoupled*. A acurácia foi calculada com base no dataset LFW. Os parâmetros *ArcLoss Scale* e *ArcLoss Angular*

Margin receberam os valores 64 e $0,5$, respectivamente, conforme Deng, Guo e Zafeiriou [60]. β_2 recebeu o valor de $0,999$. Os demais hyperparâmetros encontrados pelo otimizador bayesiano podem ser vistos na Tabela (5.10).

ID	*Learning Rate*	β_1	*Weight Decay*	# Iterações	*Mean Accuracy*
46	$0,0002$	$0,9$	$0,0005$	$74k$	$0,5$
47	$0,000925$	$0,79$	$0,000758$	$74k$	$0,5$
48	$0,00485$	$0,608$	$0,000333$	$74k$	$0,5$
49	$0,00177$	$0,573$	$0,0009$	$74k$	$0,5$

Tabela 5.10: Hyperparâmetros encontrados pelo otimizador bayesiano no treinamento da rede de reconhecimento de face, com otimizador Adam com *weight decay decoupled*, ArcLoss na função de custo, e dataset CASIA-WebFace. β_1 corresponde ao parâmetro do otimizador Adam.

Após a não convergência da rede, trocou-se o otimizador de Adam para *Stochastic Gradient Descent*, também com *weight decay decoupled*, de modo fazer uso do mesmo otimizador utilizado pelos autores. Também, para o primeiro treinamento, utilizou-se dos mesmos hyperparâmetros que os autores propuseram.

A partir disso, pode-se visualizar na Tabela (5.11) os hyperparâmetros encontrados pelo otimizador bayesiano.

ID	*Learning Rate*	*Momentum*	*Weight Decay*	# Iterações	*Mean Accuracy*
50	$0,1$	$0,9$	$0,0005$	$74k$	$0,5$
51	$0,00282$	$0,654$	$0,000253$	$74k$	$0,5$
52	$0,0028$	$0,757$	$0,000753$	$74k$	$0,5$
53	$0,98$	$0,504$	$0,000394$	$74k$	$0,5$
54	$0,0282$	$0,756$	$0,000371$	$74k$	$0,5$
55	$0,026$	$0,583$	$0,000323$	$74k$	$0,5$
56	$0,015$	$0,675$	$0,000104$	$74k$	$0,5$
57	$0,0467$	$0,801$	$0,000547$	$74k$	$0,5$

Tabela 5.11: Hyperparâmetros encontrados pelo otimizador bayesiano no treinamento da rede de reconhecimento de face, com otimizador *Stochastic Gradient Descent* com *weight decay decoupled*, ArcLoss na função de custo, e dataset CASIA-WebFace.

Como a rede não teve aprendizado significativo, após análises do gráfico do erro da rede, decidiu-se implementar o *learning rate decay* conforme proposto por Deng, Guo e Zafeiriou [60]. Para o datast CASIA-WebFace, nas iterações $20k$ e $28k$, ocorreu a divisão da *learning rate* por 10.

Os resultados desses treinamentos e os hyperparâmetros encontrados pelo otimizador bayesiano podem ser visualizados na Tabela (5.12).

ID	*Learning Rate*	*Momentum*	*Weight Decay*	# Iterações	*Mean Accuracy*
58	$0,1$	$0,9$	$0,0005$	$74k$	$0,5$
59	$0,00692$	$0,73$	$0,000785$	$74k$	$0,5$
60	$0,00522$	$0,729$	$0,000216$	$74k$	$0,5$
61	$0,997$	$0,605$	$0,000129$	$74k$	$0,5$

Tabela 5.12: Hyperparâmetros encontrados pelo otimizador bayesiano no treinamento da rede de reconhecimento de face, com otimizador *Stochastic Gradient Descent* com *weight decay decoupled*, *learning rate decay*, ArcLoss na função de custo, e dataset CASIA-WebFace.

Mesmo com os hyperparâmetros originais e com mesmo método de treinamento que o utilizado por Deng, Guo e Zafeiriou [60], não foi possível encontrar convergência na rede de reconhecimento de faces. A função de custo ArcLoss provou-se ser muito instável durante os treinamentos realizados, com o valor do erro indo a 0 em diversos momentos.

A partir desse ponto, não foram realizadas novas tentativas de convergência da rede, devido ao tempo limitado do presente trabalho.

De forma a facilitar a reprodutibilidade dos experimentos, todos os códigos estão disponíveis no GitHub[2] em repositório público[3].

[2] `https://github.com/`

[3] `https://github.com/OliRafa/SRFR-GAN`

Capítulo 6

Conclusão

Dos obstáculos existentes para o *Face Recognition*, um dos mais desafiadores é o reconhecimento em imagens de baixa resolução e/ou qualidade. A utilização de algoritmos de *Super-Resolution* parece ser um caminho promissor para a melhoria da classificação dos rostos e, sendo as redes GAN o atual *State of the Art* em SR, decorre então em novas possibilidades de pesquisas na tentativa de resolver o problema exposto.

A partir dos resultados obtidos por Cheng, Zhu e Gong [5], decidiu-se unir a ideia arquitetural e de treinamento exposta pelos citados com o poder generalizador das redes generativas, a fim de mensurar os resultados no desempenho de reconhecimento de faces, e então realizar comparações dos mesmos com outras redes SOTA. Para tal, foi desenvolvida a rede de *joint-learn*, contando com uma rede generativa nos moldes da especificada por Wang, Yu, Wu et al. [43] para realizar o aumento da resolução, acoplada a uma ResNet50 para realização do reconhecimento da face, e treinada de forma conjunta com a rede discriminadora de super-resolução.

Percebeu-se com o decorrer do trabalho que as redes GAN de SR atingem resultados satisfatórios na melhoria da resolução e qualidade das imagens, e o fazem de forma relativamente rápida, guardadas as condições iniciais das mesmas serem deveras adversas. Percebeu-se também que as redes de FR testadas, que utilizam funções de custo baseadas em Softmax, são de difícil treinamento, exigindo hyperparâmetros precisos para não acrescentar instabilidades durante a fase de treinos.

Como não foi encontrado convergência no reconhecimento das faces, não foi possível concluir alguns dos objetivos destacados anteriormente. Por consequência, não foi comparada a rede proposta com os atuais SOTA em reconhecimento de faces de baixa qualidade de imagem.

Evidencia-se que, tanto a utilização da rede GAN quanto o treinamento conjunto entre SR e FR não foram as causas da não convergência da rede, já que a mesma não convergiu mesmo quando treinada com os hyperparâmetros propostos no trabalho original de Deng, Guo e Zafeiriou [60], apesar das consequências do treinamento conjunto na parcela de FR ainda serem incertas devido a não convergência.

6.1 Trabalhos Futuros

Como o trabalho realizado não cumpriu parte dos objetivos propostos, algumas soluções podem ser testadas de modo e atingir esses objetivos, sendo elas:

- **Modificar a arquitetura da rede de *Face Recognition***: ResNet50 foi a rede utilizada para a parcela de reconhecimento das faces, mas outras redes podem ser testadas para verificar a convergência do modelo como um todo, como as versões mais profundas da ResNet (ResNet101 e ResNet152), as versões atualizadas da mesma (ResNet50V2, ResNet101V2 e ResNet152V2), ou mesmo outras arquiteturas;

- **Modificar função de custo na parcela de reconhecimento de faces**: Como a função ArcLoss testada adiciona uma instabilidade considerável ao treinamento, propõe-se testar a utilização de outras funções de custo, como a Center Loss [76] ou a SphereFace Loss [75]. Ambas as funções propostas são baseadas na Softmax e são consideradas SOTA.

  Outra possibilidade é utilizar uma função de custo baseada em *triplet loss*, que, apesar de não possuir os melhores resultados atualmente, foi muito testada ao longo do tempo, sendo considerada muito sólida;

- **Adicionar mais camadas de *Batch Normalization* e *Dropout***: Como as camadas de *Batch Normalization* e *Dropout* atuam como regularizadores na rede, as camadas de BN auxiliaram na convergência do modelo, conforme foi mostrado anteriormente nos experimentos, mesmo que ao custo de imagens não tão perceptivelmente agradáveis quando em comparação a imagens geradas pelas redes sem camadas de BN. Propõe-se então a utilização de mais dessas camadas a fim de testar a convergência do modelo como um todo, auxiliando assim na classificação dos rostos;

- **Aplicar métodos de redução de dimensionalidade**: Como a dificuldade de um classificador está em selecionar quais são as características mais relevantes em um dado de forma a classificá-lo corretamente, e como as imagens de resolução aumentada podem possuir artefatos que possam induzir o classificador erroneamente, propõe-se a utilização de algum algoritmo de redução de dimensionalidade entre a parcela de SR e o FR, de forma a facilitar a classificação dos rostos. Algumas possibilidades são a utilização de *Principal Component Analysis (PCA)* ou de *Haar Scattering Networks* [78]–[80].

Além das possibilidade expostas na tentativa de buscar a convergência, também propõe-se aumentar a resolução da imagem de entrada em 8x o tamanho original - passando de $28 \times 28px$ para $224 \times 24px$ - e posteriormente reduzir a sua resolução para 4x, tendo então seu resultado final como $112 \times 112px$. Com uma maior resolução de imagem pode haver o mapeamento de mais características da face, gerando assim imagens mais fieis ao rosto do indivíduo, e o posterior condensamento da resolução pode trazer uma maior qualidade para a imagem final gerada.

Ademais, espera-se que as contribuições do presente trabalho possam embasar futuras pesquisas realizadas pela comunidade científica no presente tópico e, para tal, todo o código utilizado se encontra em repositório aberto[1].

[1] `https://github.com/OliRafa/SRFR-GAN`

Bibliografia

[1] L. A. Addington, «Cops and Cameras: Public School Security as a Policy Response to Columbine», *American Behavioral Scientist*, vol. 52, n.º 10, pp. 1426–1446, 2009. DOI: 10.1177/0002764209332556. eprint: https://doi.org/10.1177/0002764209332556. URL: https://doi.org/10.1177/0002764209332556.

[2] B. W. Fisher, E. M. Higgins e E. M. Homer, «School Crime and Punishment and the Implementation of Security Cameras: Findings from a National Longitudinal Study», *Justice Quarterly*, vol. 0, n.º 0, pp. 1–25, 2019. DOI: 10.1080/07418825.2018.1518476. eprint: https://doi.org/10.1080/07418825.2018.1518476. URL: https://doi.org/10.1080/07418825.2018.1518476.

[3] Y. Zhou, D. Liu e T. Huang, «Survey of face detection on low-quality images», *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, pp. 769–773, 2018. DOI: 10.1109/FG.2018.00121.

[4] K. Nguyen, C. Fookes, S. Sridharan, M. Tistarelli e M. Nixon, «Super-resolution for biometrics: A comprehensive survey», *Pattern Recognition*, vol. 78, pp. 23–42, 2018, ISSN: 00313203. DOI: 10.1016/j.patcog.2018.01.002. URL: https://doi.org/10.1016/j.patcog.2018.01.002.

[5] Z. Cheng, X. Zhu e S. Gong, «Low-Resolution Face Recognition», *CoRR*, vol. abs/1811.08965, 2018.

[6] Á. Persechino e M. Albuquerque, «Digital image processing: fundamental concepts», *Monografia-CBPF*, vol. 1, n.º 4, pp. 1–41, out. de 2015. DOI: 10.7437/mo24471119/2015.04.001. URL: https://doi.org/10.7437/mo24471119/2015.04.001.

[7] R. C. Gonzalez e R. E. Woods, *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008, ISBN: 9780131687288 013168728X 9780135052679 013505267X. URL: http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X.

[8] M. Sonka, *Image Processing, Analysis, and Machine Vision*. CL Engineering, jan. de 2014, ISBN: 1133593607. URL: https://www.xarg.org/ref/a/1133593607/.

[9] R. Pascanu, T. Mikolov e Y. Bengio, «Understanding the exploding gradient problem», *CoRR*, vol. abs/1211.5063, 2012. arXiv: 1211.5063. URL: http://arxiv.org/abs/1211.5063.

[10] P. Milanfar, *Super-Resolution Imaging*, sér. Digital Imaging and Computer Vision. CRC Press, 2010, ISBN: 9781439819319. URL: https://books.google.com.br/books?id=fjTUbMnvOkgC.

[11] F. Crété-Roffet, T. Dolmiere, P. Ladret e M. Nicolas, «The Blur Effect: Perception and Estimation with a New No-Reference Perceptual Blur Metric», em *SPIE Electronic Imaging Symposium Conf Human Vision and Electronic Imaging*, vol. XII, San Jose, United States, jan. de 2007, EI 6492-16. URL: https://hal.archives-ouvertes.fr/hal-00232709.

[12] G. V. S. Kumar, «REVIEW ON IMAGE SEGMENTATION TECHNIQUES», *International Journal of Scientific Research Engineering & Technology (IJSRET)*, vol. 3, pp. 992–997, set. de 2014.

[13] D. Poole, A. Mackworth e R. Goebel, *Computational Intelligence: A Logical Approach*. New York, NY, USA: Oxford University Press, Inc., 1997, ISBN: 0-19-510270-3.

[14] S. Russell e P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009, ISBN: 0136042597, 9780136042594.

[15] T. M. Mitchell, *Machine learning*, sér. McGraw Hill series in computer science. McGraw-Hill, 1997, ISBN: 978-0-07-042807-2. URL: http://www.worldcat.org/oclc/61321007.

[16] K. Faceli, J. Gama, A. Lorena e A. De Carvalho, *Inteligência artificial: uma abordagem de aprendizado de máquina*. Grupo Gen - LTC, 2011, ISBN: 9788521618805. URL: https://books.google.com.br/books?id=4DwelAEACAAJ.

[17] A. Machado e L. Haertel, *Neuroanatomia funcional*. Atheneu, 2014, ISBN: 9788538804574. URL: https://books.google.com.br/books?id=QL4CDQEACAAJ.

[18] W. S. McCulloch e W. Pitts, «A logical calculus of the ideas immanent in nervous activity», *The Bulletin of Mathematical Biophysics*, vol. 5, n.º 4, pp. 115–133, dez. de 1943. DOI: 10.1007/bf02478259. URL: https://doi.org/10.1007/bf02478259.

[19] S. S. Haykin, *Neural networks and learning machines*, Third. Upper Saddle River, NJ: Pearson Education, 2009.

[20] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[21] M. A. Nielsen, *Neural Networks and Deep Learning*, 2015. URL: http://neuralnetworksanddeeplea com/.

[22] K. Hornik, «Approximation capabilities of multilayer feedforward networks», *Neural Networks*, vol. 4, n.º 2, pp. 251–257, 1991, ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T. URL: http://www.sciencedirect.com/science/article/pii/089360809190009T.

[23] Y. Lecun, L. Bottou, Y. Bengio e P. Haffner, «Gradient-based learning applied to document recognition», *Proceedings of the IEEE*, vol. 86, n.º 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791. URL: https://doi.org/10.1109/5.726791.

[24] K. Simonyan e A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014. arXiv: 1409.1556 [cs.CV].

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke e A. Rabinovich, «Going Deeper with Convolutions», *CoRR*, vol. abs/1409.4842, 2014. arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[26] D. Misra, «Mish: A self regularized non-monotonic neural activation function», *arXiv preprint arXiv:1908.08681*, 2019.

[27] L. Wright. (2019). Meet Mish New State of the Art AI Activation Function. The successor to ReLU?, URL: https://medium.com/@lessw/meet-mish-new-state-of-the-art-ai-activation-function-the-successor-to-relu-846a6d93471f (acedido em 02/10/2020).

[28] D. Sussillo, «Random Walks: Training Very Deep Nonlinear Feed-Forward Networks with Smart Initialization», *CoRR*, vol. abs/1412.6558, 2014. arXiv: 1412.6558. URL: http://arxiv.org/abs/1412.6558.

[29] B. Ginsburg, P. Castonguay, O. Hrinchuk, O. Kuchaiev, V. Lavrukhin, R. Leary, J. Li, H. Nguyen, Y. Zhang e J. M. Cohen, *Stochastic Gradient Methods with Layer-wise Adaptive Moments for Training of Deep Networks*, 2020. arXiv: 1905.11286 [cs.LG].

[30] E. Hazan, K. Y. Levy e S. Shalev-Shwartz, *Beyond Convexity: Stochastic Quasi-Convex Optimization*, 2015. arXiv: 1507.02030 [cs.LG].

[31] D. P. Kingma e J. Ba, *Adam: A Method for Stochastic Optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[32] A. W. Yu, L. Huang, Q. Lin, R. Salakhutdinov e J. Carbonell, *Block-Normalized Gradient Method: An Empirical Study for Training Deep Neural Network*, 2018. arXiv: 1707.04822 [cs.LG].

[33] I. Loshchilov e F. Hutter, *Decoupled Weight Decay Regularization*, 2019. arXiv: 1711.05101 [cs.LG].

[34] H. Lee, R. Grosse, R. Ranganath e A. Y. Ng, «Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations», *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pp. 1–8, 2009. DOI: 10.1145/1553374.1553453. URL: http://portal.acm.org/citation.cfm?doid=1553374.1553453.

[35] U. Karn, *An intuitive explanation ofconvolutional neural networks*, Disponível em: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/. Acesso em: 02 maio 2019, 2016.

[36] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012, ISBN: 0262018020.

[37] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville e Y. Bengio, *Generative Adversarial Networks*, 2014. arXiv: 1406.2661 [stat.ML].

[38] J. Brownlee. (2019). A Gentle Introduction to Generative Adversarial Networks (GANs), URL: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (acedido em 03/10/2020).

[39] A. Mordvintsev, C. Olah e M. Tyka. (2015). Inceptionism: Going Deeper into Neural Networks, URL: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html (acedido em 03/10/2020).

[40] J. Brownlee. (2019). A Gentle Introduction to Cross-Entropy for Machine Learning, URL: https://machinelearningmastery.com/cross-entropy-for-machine-learning/ (acedido em 04/10/2020).

[41] V. Kulkarni. (2019). Cross-Entropy for Dummies: A simple and intuitive explanation of information, entropy, and cross-entropy for data scientists, URL: https://towardsdatascience.com/cross-entropy-for-dummies-5189303c7735 (acedido em 04/10/2020).

[42] A. Jolicoeur-Martineau, *The relativistic discriminator: a key element missing from standard GAN*, 2018. arXiv: 1807.00734 [cs.LG].

[43] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao e X. Tang, *ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks*, 2018. arXiv: 1809.00219 [cs.CV].

[44] K. Hayat, «Super-Resolution via Deep Learning», *CoRR*, vol. abs/1706.09077, 2017. arXiv: 1706.09077. URL: http://arxiv.org/abs/1706.09077.

[45] J. Kim, J. K. Lee e K. M. Lee, «Accurate Image Super-Resolution Using Very Deep Convolutional Networks», *CoRR*, vol. abs/1511.04587, 2015. arXiv: 1511.04587. URL: http://arxiv.org/abs/1511.04587.

[46] C. Dong, C. C. Loy, K. He e X. Tang, «Image Super-Resolution Using Deep Convolutional Networks», *CoRR*, vol. abs/1501.00092, 2015. arXiv: 1501.00092. URL: http://arxiv.org/abs/1501.00092.

[47] J. Shermeyer e A. V. Etten, «The Effects of Super-Resolution on Object Detection Performance in Satellite Imagery», *CoRR*, vol. abs/1812.04098, 2018. arXiv: 1812.04098. URL: http://arxiv.org/abs/1812.04098.

[48] K. He, X. Zhang, S. Ren e J. Sun, «Deep Residual Learning for Image Recognition», *CoRR*, vol. abs/1512.03385, 2015. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[49] J. Kim, J. Kwon Lee e K. Mu Lee, «Deeply-Recursive Convolutional Network for Image Super-Resolution», em *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jun. de 2016.

[50] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang e W. Shi, *Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network*, 2017. arXiv: 1609.04802 [cs.CV].

[51] B. Prihasto, S. Choirunnisa, M. I. Nurdiansyah, S. Mathulaprangsan, V. C.-M. Chu, S.-H. Chen e J.-C. Wang, «A survey of deep face recognition in the wild», em *2016 International Conference on Orange Technologies (ICOT)*, IEEE, dez. de 2016. DOI: 10.1109/icot.2016.8278983. URL: https://doi.org/10.1109/icot.2016.8278983.

[52] A. Almadhor, «Comparative Analysis of Face Detection Algorithms: Novice to Novel», *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1642–1647, 2018.

[53] P. Viola e M. Jones, «Robust Real-time Object Detection», *International Journal of Computer Vision*, vol. 57, n.º 2, pp. 137–154, 2002.

[54] A. Srivastava, S. Mane, A. Shah, N. Shrivastava e B. Thakare, «A survey of face detection algorithms», em *2017 International Conference on Inventive Systems and Control (ICISC)*, IEEE, jan. de 2017. DOI: 10.1109/icisc.2017.8068607. URL: https://doi.org/10.1109/icisc.2017.8068607.

[55] K. Dang e S. Sharma, «Review and comparison of face detection algorithms», em *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*, IEEE, jan. de 2017. DOI: 10.1109/confluence.2017.7943228. URL: https://doi.org/10.1109/confluence.2017.7943228.

[56] F. Zhang, X. Fan, G. Ai, J. Song, Y. Qin e J. Wu, «Accurate Face Detection for High Performance», *CoRR*, vol. abs/1905.01585, 2019. arXiv: 1905.01585. URL: http://arxiv.org/abs/1905.01585.

[57] T. Lin, P. Goyal, R. B. Girshick, K. He e P. Dollár, «Focal Loss for Dense Object Detection», *CoRR*, vol. abs/1708.02002, 2017. arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[58] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan e S. J. Belongie, «Feature Pyramid Networks for Object Detection», *CoRR*, vol. abs/1612.03144, 2016. arXiv: 1612.03144. URL: http://arxiv.org/abs/1612.03144.

[59] M. Wang e W. Deng, «Deep Face Recognition: A Survey», *CoRR*, vol. abs/1804.06655, 2018. arXiv: 1804.06655. URL: http://arxiv.org/abs/1804.06655.

[60] J. Deng, J. Guo e S. Zafeiriou, «ArcFace: Additive Angular Margin Loss for Deep Face Recognition», *CoRR*, vol. abs/1801.07698, 2018. arXiv: 1801.07698. URL: http://arxiv.org/abs/1801.07698.

[61] A. Lemieux e M. Parizeau, «Experiments on eigenfaces robustness», em *Object recognition supported by user interaction for service robots*, IEEE Comput. Soc, 2002. DOI: 10.1109/icpr.2002.1044743. URL: https://doi.org/10.1109/icpr.2002.1044743.

[62] F. C.-H. Lin, «Super-resolution image processing with application to face recognition», tese de doutoramento, Queensland University of Technology, 2008. URL: https://eprints.qut.edu.au/16703/.

[63] C. Fookes, F. Lin, V. Chandran e S. Sridharan, «Evaluation of image resolution and super-resolution on face recognition performance», *Journal of Visual Communication and Image Representation*, vol. 23, n.º 1, pp. 75–93, 2012, ISSN: 1047-3203. DOI: https://doi.org/10.1016/j.jvcir.2011.06.004. URL: http://www.sciencedirect.com/science/article/pii/S1047320311000721.

[64] P. H. Hennings-Yeomans, S. Baker e B. V. Kumar, «Simultaneous super-resolution and feature extraction for recognition of low-resolution faces», em *2008 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, jun. de 2008. DOI: 10.1109/cvpr.2008.4587810. URL: https://doi.org/10.1109/cvpr.2008.4587810.

[65] P. Rasti, T. Uiboupin, S. Escalera e G. Anbarjafari, «Convolutional Neural Network Super Resolution for Face Recognition in Surveillance Monitoring», em *Articulated Motion and Deformable Objects*, F. J. Perales e J. Kittler, eds., Cham: Springer International Publishing, 2016, pp. 175–184, ISBN: 978-3-319-41778-3.

[66] Z. Wang, S. Chang, Y. Yang, D. Liu e T. S. Huang, «Studying Very Low Resolution Recognition Using Deep Networks», *CoRR*, vol. abs/1601.04153, 2016. arXiv: 1601.04153. URL: http://arxiv.org/abs/1601.04153.

[67] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané,

Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu e Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015. URL: `https://www.tensorflow.org/`.

[68] D. Yi, Z. Lei, S. Liao e S. Z. Li, *Learning Face Representation from Scratch*, 2014. arXiv: `1411.7923 [cs.CV]`.

[69] Y. Taigman, M. Yang, M. Ranzato e L. Wolf, «DeepFace: Closing the Gap to Human-Level Performance in Face Verification», em *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708. DOI: `10.1109/CVPR.2014.220`.

[70] Y. Sun, X. Wang e X. Tang, *Deep Learning Face Representation by Joint Identification-Verification*, 2014. arXiv: `1406.4773 [cs.CV]`.

[71] Q. Cao, L. Shen, W. Xie, O. M. Parkhi e A. Zisserman, *VGGFace2: A dataset for recognising faces across pose and age*, 2018. arXiv: `1710.08092 [cs.CV]`.

[72] G. B. Huang, M. Ramesh, T. Berg e E. Learned-Miller, «Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments», University of Massachusetts, Amherst, rel. téc. 07-49, out. de 2007.

[73] A. Horé e D. Ziou, «Image Quality Metrics: PSNR vs. SSIM», em *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369. DOI: `10.1109/ICPR.2010.579`.

[74] K. Zhang, Z. Zhang, Z. Li e Y. Qiao, «Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks», *IEEE Signal Processing Letters*, vol. 23, n.º 10, pp. 1499–1503, out. de 2016, ISSN: 1070-9908. DOI: `10.1109/LSP.2016.2603342`.

[75] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj e L. Song, *SphereFace: Deep Hypersphere Embedding for Face Recognition*, 2018. arXiv: 1704.08063 [cs.CV].

[76] Y. Wen, K. Zhang, Z. Li e Y. Qiao, «A Discriminative Feature Learning Approach for Deep Face Recognition», em *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe e M. Welling, eds., Cham: Springer International Publishing, 2016, pp. 499–515, ISBN: 978-3-319-46478-7.

[77] J. Mokus, «On bayesian methods for seeking the extremum», em *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, G. I. Marchuk, ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 400–404, ISBN: 978-3-540-37497-8.

[78] X. Cheng, X. Chen e S. Mallat, *Deep Haar Scattering Networks*, 2015. arXiv: 1509.09187 [cs.LG].

[79] F. F. Neto, *Building Function Approximators on top of Haar Scattering Networks*, 2018. arXiv: 1804.03236 [stat.ML].

[80] F. F. Neto, A. A. Solomon, R. de Losso, C. Garcia e P. D. Cavalcanti, *Deep Haar Scattering Networks in Pattern Recognition: A promising approach*, 2018. arXiv: 1811.12081 [eess.SP].

Apêndice A

Proposta Original do Projeto

Proposta de tema para
Dissertação/Estágio/Projeto - Trabalho de Conclusão de Curso

Orientador da Instituição onde se realiza o trabalho:

Pedro João Soares Rodrigues	pjsr@ipb.pt

Instituição do orientador:

IPB	ESTiG

Co-orientador da Instituição parceira:

Arnaldo Candido Junior; Pedro Luiz de Paula Filho	arnaldocan@gmail.com; plpf200

Instituição do co-orientador:

UTFPR	Câmpus Medianeira

Curso ou cursos da Instituição do orientador onde se propõe que o trabalho seja realizado:

Mestrado em Sistema de Informação

Título do trabalho:

Reconhecimento Facial com Super-Resolução: uma abordagem utilizando Redes Generativas e Joint-Learn

Palavras chave:

Aprendizagem máquina; Deep Learning; Super-resolução; Reconhecimento de faces.

Objetivos:

Estabelecer arquiteturas neurais de aprendizagem automática para tratar e classificar imagens faciais de baixa resolução.