Relatório Final - Iniciação Científica

Restauração de imagens com técnicas de aprendizado de máquina

Submetido à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)

Bolsista: Guilherme Albuquerque Lizarzaburu
Orientador: Magno Teófilo Madeira da Silva
Coorientador: Renato Candido

Processo Nº: 2019/08636-8
Período de vigência: 01/08/2019 a 31/07/2020

Universidade de São Paulo (USP)
Escola Politécnica
Departamento de Engenharia de Sistemas Eletrônicos (PSI)
Av. Prof. Luciano Gualberto, 158, trav. 3
São Paulo - SP
05508-010

São Paulo

2020

Sumário

Resumo

Este relatório se inicia com o resumo, os objetivos, o cronograma e a metodologia do plano de pesquisa para uma comparação entre as atividades previstas e realizadas nesta Iniciação Científica. Na Seção 2, é apresentada a formulação dos problemas envolvendo a restauração de imagens degradadas por funções de espalhamento de ponto gaussianas. Na Seção 3, são descritas as arquiteturas de redes neurais utilizadas como soluções para os problemas propostos. As seguintes redes foram consideradas neste trabalho: *perceptron* multicamada, redes convolucionais, redes convolucionais residuais, mistura de especialistas e redes gerativas adversárias. Na Seção 4, são apresentados os resultados de simulação e na Seção 5, é apresentada a conclusão deste relatório comparando as atividades realizadas com as propostas no plano de pesquisa inicial. Cabe observar que conceitos fundamentais de redes neurais, já abordados no Relatório Parcial, foram organizados em três apêndices.

1 Resumo do Plano de Pesquisa

Recentemente, soluções baseadas em aprendizado de máquina para restauração de imagens têm sido propostas na literatura. Em geral, as distorções de imagens são modeladas por funções de espalhamento de ponto lineares e invariantes no espaço [1]. No entanto, essas distorções, na prática, são não lineares, o que justifica o uso de soluções não lineares para mitigar o efeito das mesmas.

Neste trabalho de Iniciação Científica, pretende-se utilizar redes neurais para melhorar a qualidade de imagens. Particularmente, serão considerados três tipos de redes: (I) *perceptron* multicamada (*Multilayer Perceptron* - MLP), (II) convolucional (*Convolutional Neural Network* - CNN) e (III) gerativa adversária (*Generative Adversarial Network* - GAN) [2]. Pretende-se estudar a viabilidade de se treinar essas redes para maximizar o índice de similaridade estrutural, medida que leva em conta características do sistema visual humano e tem sido amplamente usada na literatura para comparar a similaridade entre duas imagens [3, 4]. Por fim, para obter bons resultados independentemente do tipo de distorção, pretende-se ainda combinar essas redes em uma mistura de especialistas (*mixture of experts* – ME) [5, 6].

1.1 Objetivos

Os principais objetivos deste projeto de pesquisa são:

1. Realizar um estudo de técnicas de aprendizado de máquina aplicadas à restauração de imagens.
2. Estudar diferentes soluções baseadas em redes neurais, focando principalmente nas redes MLP, CNN e GAN.
3. Verificar a possibilidade de treinar as redes para maximizar o índice SSIM.
4. Estudar e implementar uma mistura de especialistas para o problema.

1.2 Cronograma de Atividades Proposto

As atividades a serem realizadas estão listadas a seguir:

1. Estudo de técnicas de aprendizado de máquina a partir do curso *Introduction to Deep Learning* ministrado por Andrew Ng, disponível no *Coursera*.
2. Estudo e implementação das redes MLP e CNN para restauração de imagens.

3. Extensão dos resultados com uma função custo baseada no índice SSIM.

4. Estudo e implementação de uma solução baseada em rede GAN.

5. Estudo e implementação de uma mistura de especialistas.

6. Análise dos resultados.

O cronograma de atividades previsto é mostrado na Tabela 1.

Tabela 1: Cronograma de atividades.

Atividade	2019			2020		
Meses	ago/set	out/nov	dez	jan/fev	mar/abr/mai	jun/jul
1	X	X				
2	X	X	X			
3			X	X	X	
4				X	X	
5				X	X	X
6					X	X

1.3 Metodologia

Durante um estudo inicial sobre o funcionamento de redes neurais, foram implementados exemplos didáticos usando somente ferramentas computacionais para álgebra linear, fornecidas pela biblioteca *Numpy* da linguagem *Python*. O Problema das Meia-luas tratado no Apêndice A foi realizado neste contexto de aprendizado e familiarização com redes neurais.

Posteriormente, nas simulações computacionais para a restauração de imagens, foram utilizadas as bibliotecas *Tensorflow* e *Keras*, especializadas na construção de redes neurais. A escolha por essas bibliotecas, ambas da linguagem *Python*, se deu pela praticidade oferecida por elas e pelas suas implementações eficientes para o uso de unidades de processamento do tipo GPU.

O ambiente computacional utilizado para a confecção dos modelos de restauração foi o *Jupyter Notebook* e foi usada a biblioteca *Matplotlib* para a visualização dos resultados. Os *notebooks* criados para a implementação das redes neurais restauradoras de imagens, assim como os conjuntos de dados usados para o treinamento e teste dessas redes, se encontram no repositório público *https://github.com/gui-lizarza/image-deblurring-neural-nets*.

2 Formulação do Problema

Os problemas de restauração de imagens tratados neste relatório se baseiam em imagens degradadas por meio de uma categoria específica de degradação, a Função de Espalhamento de Ponto (*Point Spread Function* - PSF) gaussiana [1]. Sendo assim, os experimentos listados neste relatório envolvem diversos cenários de degradação possíveis a partir de PSFs gaussianas.

O primeiro problema apresentado, o Problema dos 8 Níveis de Cinza, envolve a restauração de imagens formadas por *pixels* que podem assumir apenas 8 gradações distintas de cinza, incluindo preto e branco. Em seguida, as restaurações são generalizadas para imagens em 256 níveis de cinza e para imagens coloridas compostas por três canais distintos de cor (vermelho, verde e azul), com cada canal apresentando 256 gradações possíveis de sua cor correspondente. A fim de emular uma situação em que não se sabe ao certo o tipo de degradação da imagem, consideram-se também imagens degradadas por diferentes filtros gaussianos, alterando-se suas dimensões e variâncias.

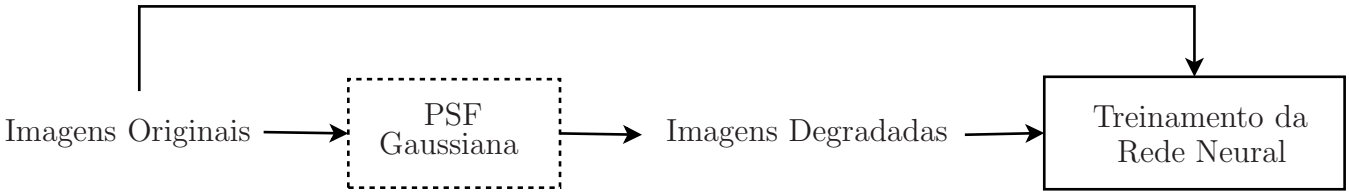
Para se medir o desempenho dos modelos concebidos, é considerado o índice de Similaridade Estrutural (*Structural Similarity* - SSIM) [3, 4] como forma de averiguar a similaridade entre duas imagens. No caso deste relatório, para se ter um controle da qualidade das restaurações feitas, são medidas as similaridades entre as imagens sem degradação e as imagens restauradas, quanto maior a similaridade melhor o desempenho do modelo.

Para a resolução dos problemas, são propostos diversos modelos baseados em técnicas de aprendizado de máquina, mais especificamente são usadas redes neurais, cuja fundamentação teórica se encontra nos Apêndices A, B e C. É considerada inicialmente uma fase de treinamento desses modelos, na qual se utiliza um conjunto específico de imagens não degradadas, ou originais, e suas respectivas degradações por PSFs gaussianas, chamado de *conjunto de treinamento*. Com os modelos já treinados, é feita uma verificação de seus desempenhos com um outro conjunto de imagens originais e degradadas, chamado de *conjunto de teste*. Vale lembrar que, em uma situação real, as imagens originais não estão disponíveis. No entanto, elas são consideradas no teste dos modelos treinados apenas para medir a qualidade dos mesmos. Na Figura 1, são esquematizados os processos de treinamento e teste dos modelos propostos.

2.1 Função de Espalhamento de Ponto

A Função de Espalhamento de Ponto é um modelo linear de distorção de imagens, isto é, a imagem degradada consiste na convolução entre duas matrizes, a imagem original e o filtro da

Treinamento:



Teste:

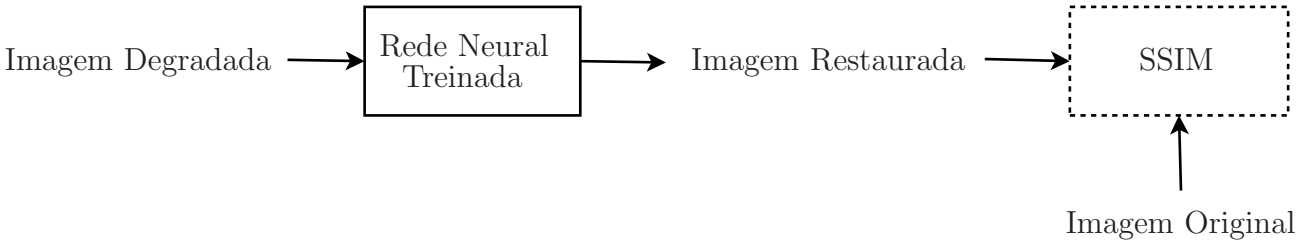


Figura 1: Esquemas de treinamento e teste dos modelos considerados.

PSF [1]. Por simplicidade, são considerados neste trabalho apenas filtros gaussianos.

Em duas dimensões, um filtro gaussiano $\mathbf{H}$ é dado pela matriz

$$\mathbf{H}(n_1, n_2) = Ke^{-\frac{{n_1}^2+{n_2}^2}{2\sigma^2}}, \tag{1}$$

em que σ é o desvio padrão do filtro e K é uma constante de normalização, escolhida para que $\sum_{n_1}\sum_{n_2}\mathbf{H}(n_1, n_2) = 1$. Sendo L_1 a quantidade de linhas do filtro, tem-se $n_1 = -(L_1 - 1)/2, \ldots, -1, 0, 1, \ldots, (L_1-1)/2$ caso L_1 seja ímpar, e $n_1 = -(L_1-1)/2, \ldots, -0{,}5, 0{,}5, \ldots, (L_1 - 1)/2$ caso L_1 seja par. Analogamente, sendo L_2 a quantidade de colunas do filtro, tem-se $n_2 = -(L_2 - 1)/2, \ldots, -1, 0, 1, \ldots, (L_2 - 1)/2$ caso L_2 seja ímpar, e $n_2 = -(L_2 - 1)/2, \ldots, -0{,}5, 0{,}5, \ldots, (L_2 - 1)/2$ caso L_2 seja par. Um filtro gaussiano, portanto, contém amostras de uma função gaussiana bidimensional com variância σ^2 em torno do centro da matriz. Neste trabalho, a dimensão de um filtro gaussiano é dada pela notação (L_1, L_2). Exemplos numéricos para filtros gaussianos com $\sigma = 3$ de dimensões $(3, 3)$ e $(3, 2)$, respectivamente, são

$$\mathbf{H}_{(3,3);\,\sigma=3} = \begin{array}{c} \\ n_1 = 1 \\ n_1 = 0 \\ n_1 = -1 \end{array} \begin{array}{c} \begin{array}{ccc} n_2 = -1 & n_2 = 0 & n_2 = 1 \end{array} \\ \begin{pmatrix} 0{,}1069973 & 0{,}11310982 & 0{,}1069973 \\ 0{,}11310982 & 0{,}11957153 & 0{,}11310982 \\ 0{,}1069973 & 0{,}11310982 & 0{,}1069973 \end{pmatrix} \end{array} \text{e}$$

$$\mathbf{H}_{(3,2);\,\sigma=3} = \begin{array}{c} \\ n_1 = 1 \\ n_1 = 0 \\ n_1 = -1 \end{array} \begin{array}{c} \begin{array}{cc} n_2 = -0{,}5 & n_2 = 0{,}5 \end{array} \\ \begin{pmatrix} 0{,}16355221 & 0{,}16355221 \\ 0{,}17289558 & 0{,}17289558 \\ 0{,}16355221 & 0{,}16355221 \end{pmatrix} \end{array}.$$

Denotando por $\mathbf{F}$ a imagem original e $\mathbf{G}$ a imagem distorcida pela PSF, temos que

$$\mathbf{G} = \mathbf{F} * \mathbf{H}, \tag{2}$$

em que $*$ representa a operação de convolução em duas dimensões.

Para que as dimensões de $\mathbf{F}$ e $\mathbf{G}$ sejam as mesmas, é comum, antes da convolução, completar a matriz $\mathbf{F}$ com uma quantidade conveniente de colunas e linhas de elementos nulos (*zero padding*) [7]. Na Figura 2(a), é mostrada uma imagem com 256 níveis de cinza, e nas Figuras 2(b) e 2(c) são mostradas suas correspondentes imagens degradadas por PSFs gaussianas de filtros de dimensão $(7, 7)$ com $\sigma = 2$ e $\sigma = 5$, respectivamente. Foi realizado o *zero padding* nas matrizes envolvidas.

(a) Imagem original (b) $\sigma = 2$ (c) $\sigma = 5$

Figura 2: Degradações por diferentes PSFs gaussianas para imagens em cinza.

2.2 Índice de Similaridade Estrutural

O índice de Similaridade Estrutural é usado para medir a similaridade entre duas imagens em nível de cinza, baseando-se em características do sistema visual humano [3,4]. O índice assume valores no intervalo $[-1, 1]$, sendo igual a 1 quando as imagens são idênticas.

O índice SSIM entre duas imagens $\mathbf{X}$ e $\mathbf{Y}$ é dado por uma combinação de três medidas distintas relacionadas à luminância ($l(\mathbf{X}, \mathbf{Y})$), ao contraste ($c(\mathbf{X}, \mathbf{Y})$) e à estrutura das imagens ($s(\mathbf{X}, \mathbf{Y})$). Tais medidas são dadas por [3]

$$l(\mathbf{X}, \mathbf{Y}) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}, \tag{3}$$

$$c(\mathbf{X}, \mathbf{Y}) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \text{ e} \tag{4}$$

$$s(\mathbf{X}, \mathbf{Y}) = \frac{2\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}, \tag{5}$$

em que μ_x e μ_y são as respectivas médias dos elementos das matrizes $\mathbf{X}$ e $\mathbf{Y}$, σ_x e σ_y são os respectivos desvios padrões e σ_{xy} é a covariância entre as matrizes. Os termos c_1, c_2 e c_3 são

definidos por [3]

$$c_1 = (k_1 P)^2, \;\; c_2 = (k_2 P)^2 \;\; \text{e} \;\; c_3 = \frac{c_2}{2}, \tag{6}$$

em que k_1 e k_2 são constantes, geralmente $k_1 = 0{,}01$ e $k_2 = 0{,}03$, e P equivale ao maior valor de *pixel* da imagem, tipicamente dado por $2^{\#\text{bits por pixel}} - 1$.

Dessa maneira, o índice SSIM é dado por [3]

$$\text{SSIM}(\mathbf{X}, \mathbf{Y}) = \left[l(\mathbf{X}, \mathbf{Y})^\alpha \cdot c(\mathbf{X}, \mathbf{Y})^\beta \cdot s(\mathbf{X}, \mathbf{Y})^\gamma \right]. \tag{7}$$

Considerando $\alpha = \beta = \gamma = 1$ e as definições (3), (4) e (5), a equação (7) se reduz a [3]

$$\text{SSIM}(\mathbf{X}, \mathbf{Y}) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \tag{8}$$

Neste relatório, considera-se apenas a definição (8) para o cálculo do SSIM. Vale ressaltar também que se utilizou a função `tensorflow.image.ssim()` da biblioteca *Tensorflow* como forma de se implementar o cálculo desse índice. Além de receber as imagens a serem comparadas e os valores de k_1 e k_2 como entradas, a função também possui um parâmetro denominado `filter_size`, que determina o tamanho das matrizes $\mathbf{X}$ e $\mathbf{Y}$ com as quais é calculado o SSIM individualmente. Para `filter_size` $=$ 11, por exemplo, temos que $\mathbf{X}$ e $\mathbf{Y}$ possuem dimensão $(11, 11)$. Sendo assim, ambas as imagens a serem consideradas para o cálculo do índice são subdivididas (sem sobreposição) em matrizes de dimensão definida por `filter_size`, e o valor final do índice SSIM é dado pela média aritmética de todos os valores SSIM calculados para cada uma dessas matrizes. No caso deste relatório, foi adotado $k_1 = 0{,}01$, $k_2 = 0{,}03$ e `filter_size` $= 11$.

2.3 Problema dos 8 Níveis de Cinza

A imagem da Figura 2(a) apresenta 256 níveis de cinza, isto é, os *pixels* da imagem apresentam até 256 gradações de cor entre preto e branco. Dessa forma, o Problema dos 8 Níveis de Cinza busca simplificar a tarefa de restauração de imagens reduzindo a quantidade de níveis de cinza das amostras analisadas.

As imagens sem degradação em 8 níveis de cinza são obtidas a partir do truncamento dos 256 níveis de cinza das imagens originais em apenas 8 níveis, e as imagens com degradação são obtidas a partir da aplicação de uma PSF gaussiana com um filtro de dimensão $(7, 7)$ e $\sigma = 2$ sobre as imagens sem degradação truncadas. Dessa maneira, as imagens degradadas não possuem apenas 8 níveis de cinza, uma vez que, devido à aplicação da PSF, os *pixels* dessas imagens não apresentam necessariamente valores inteiros. Sendo assim, o objetivo do Problema

dos 8 Níveis de Cinza é restaurar a imagem degradada de forma que ela se assemelhe o máximo possível à imagem sem degradação truncada em 8 níveis de cinza.

O fato de não se truncar também a imagem degradada não é realista. No entanto, considerando o Problema dos 8 Níveis de Cinza como um experimento inicial, decidiu-se torná-lo mais simples. Se houvesse o truncamento depois da aplicação da PSF, como são considerados poucos níveis de cinza, o problema acabaria sendo muito complicado, uma vez que tal truncamento representa uma transformação altamente não linear. Portanto, decidiu-se pelo truncamento apenas quando se considerou mais níveis (256 no caso). Vale ressaltar que a resolução de todas as imagens trabalhadas neste relatório é de 256×256.

Nas Figuras 3(a), 3(b) e 3(c), são mostradas, respectivamente, uma imagem em 256 níveis de cinza, a mesma imagem truncada em 8 níveis de cinza e a imagem degradada obtida a partir da aplicação da PSF gaussiana sobre a imagem truncada. Também é mostrado o índice SSIM da imagem degradada em relação à imagem sem degradação truncada.

(a) Original em 256 níveis de cinza

(b) Original em 8 níveis de cinza / SSIM = 1

(c) Degradada / SSIM = 0,68

Figura 3: Exemplo do tratamento de imagens para o Problema dos 8 Níveis de Cinza.

2.4 Problema dos 256 Níveis de Cinza

O Problema dos 256 Níveis de Cinza é similar ao Problema dos 8 Níveis de Cinza, a diferença é que se consideram diretamente as imagens com 256 gradações de cinza. As imagens degradadas são obtidas pela aplicação de uma PSF gaussiana, com um filtro de dimensão $(7, 7)$ e $\sigma = 2$, sobre as imagens originais em 256 níveis de cinza. Vale ressaltar que após a aplicação da PSF, os valores dos *pixels* da imagem degradada são truncados em valores inteiros no intervalo $[0, 255]$. O objetivo do problema, portanto, é obter uma restauração para a imagem degradada que se assemelhe o máximo possível à imagem original em 256 níveis.

2.5 Restauração para Vários Valores de σ

Após a discussão para os problemas envolvendo degradações com apenas um tipo de filtro gaussiano (com dimensão $(7, 7)$ e $\sigma = 2$), é avaliado o desempenho dos modelos propostos para a restauração de imagens degradadas por mais de um tipo específico de filtro. No caso, o conjunto de treinamento e teste das redes neurais passa a abranger imagens degradadas por PSFs com variados valores de σ. Vale lembrar que a dimensão dos filtros considerados ainda é $(7, 7)$.

Neste problema, além da restauração de imagens em 256 níveis de cinza, também foi incluída a restauração de imagens coloridas formadas por três canais de cores distintas (vermelho, verde e azul). Para a degradação destas imagens, considera-se a convolução do filtro da PSF com cada canal de cor individualmente. O índice SSIM entre duas imagens coloridas foi considerado como sendo a média aritmética entre os índices SSIM calculados entre cada canal de cor das duas imagens.

Na Figura 4(a), é mostrada uma imagem colorida sem degradação e nas Figuras 4(b) e 4(c), são mostradas suas degradações por PSFs gaussianas de filtro de dimensão $(7, 7)$ com $\sigma = 1$ e $\sigma = 5$, respectivamente. Os valores de índice SSIM indicados foram calculados em relação à imagem original.

(a) Original / SSIM = 1 (b) $\sigma = 1$ / SSIM = 0,88 (c) $\sigma = 5$ / SSIM = 0,70

Figura 4: Degradações por diferentes valores de σ para imagens coloridas. Índices SSIM em relação à imagem original.

2.6 Restauração para Várias Dimensões de Filtros

Diferentemente das restaurações para mais de um valor de σ, neste problema são consideradas imagens degradadas por filtros gaussianos de diferentes dimensões. No caso, são consideradas degradações com $\sigma = 3$ e filtros com dimensão $(14, 2)$, resultando em uma degradação vertical,

e dimensão $(2, 14)$, resultando numa degradação horizontal. O experimento é realizado apenas para imagens em 256 níveis de cinza.

Na Figura 5(a), é mostrada uma imagem com 256 níveis de cinza sem degradação e nas Figuras 5(b) e 5(c), são mostradas as degradações dessa imagem por PSFs gaussianas de $\sigma = 3$ com filtros de dimensões $(14, 2)$ e $(2, 14)$, respectivamente. Os valores de índice SSIM indicados foram calculados em relação à imagem original.

(a) Original / SSIM = 1

(b) Degradação vertical / SSIM = 0,61

(c) Degradação horizontal / SSIM = 0,65

Figura 5: Degradações vertical e horizontal consideradas para o problema. Índices SSIM em relação à imagem original.

3 Arquiteturas das Redes Neurais

Nesta Seção, são expostos os modelos de redes neurais usados para solucionar os problemas descritos na Seção 2. Vale ressaltar que explicações teóricas mais aprofundadas sobre os fundamentos básicos de uma rede neural se encontram nos Apêndices A, B e C deste relatório.

3.1 MLP-Adam e CNN-Maior

As redes MLP-Adam e CNN-Maior foram redes usadas em um projeto anterior de Iniciação Científica orientado pelos mesmos orientadores deste trabalho e que também possuía a restauração de imagens com redes neurais como tema [8]. Neste outro projeto de Iniciação Científica, essas redes foram usadas para o Problema dos 8 Níveis de Cinza.

Enquanto a rede CNN-Maior possui camadas convolucionais em sua composição, a rede MLP-Adam é formada apenas por camadas densas totalmente conectadas. Ambas as redes tem como entrada uma região qualquer de resolução 7×7 da imagem a ser restaurada, todas as regiões 7×7 da imagem são consideradas, havendo sobreposição entre as regiões. Para a rede CNN-Maior, devido a suas camadas convolucionais, essa região é dada por uma matriz de

dimensão $(7, 7)$. Para a rede MLP-Adam, a região é transformada em um vetor de 49 elementos. As saídas das duas redes são compostas por 8 neurônios, com cada neurônio representando um nível de cinza diferente. Como é desejado que apenas um dos neurônios de saída seja ativado por vez, a função de ativação utilizada para a última camada é a função *softmax* [7].

Sendo assim, o propósito das redes é, a partir de uma região 7×7 da imagem degradada, determinar o nível de cinza original do *pixel* central desse recorte da imagem. Nas Tabelas 2 e 3, tem-se mais especificações sobre as duas redes, incluindo a disposição de suas camadas e os hiperparâmetros usados.

	Caracterização da rede MLP-Adam
Entrada	49 elementos
Camada oculta 1	40 neurônios
Camada oculta 2	40 neurônios
Camada de saída	8 neurônios
Inicialização de parâmetros	Inicialização de Glorot
Função de ativação	ReLU (ocultas); *Softmax* (saída)
Função custo	Entropia cruzada
Tamanho do mini batch	512
Passo de adaptação	0,001
Otimizador	Adam ($\beta_1 = 0{,}9$; $\beta_2 = 0{,}99$)

Tabela 2: Sumário da rede MLP-Adam.

	Caracterização da CNN-Maior	**Saída da camada**
Entrada	Tensor de dimensão $(7, 7, 1)$	$(7, 7, 1)$
Camada oculta 1	Convolução: 8 filtros de dimensão $(3, 3, 1)$	$(5, 5, 8)$
Camada oculta 2	Convolução: 16 filtros de dimensão $(3, 3, 8)$	$(3, 3, 16)$
Camada oculta 3	Convolução: 32 filtros de dimensão $(3, 3, 16)$	$(1, 1, 32)$
Camada oculta 4	Camada totalmente conectada	$(32, 1, 1)$
Camada de saída	8 neurônios	$(8, 1, 1)$
Inicialização de parâmetros	Inicialização de Glorot	–
Função de ativação	ReLU (ocultas); *Softmax* (saída)	–
Função custo	Entropia cruzada	–
Tamanho do mini batch	512	–
Passo de adaptação	0,001	–
Otimizador	Adam ($\beta_1 = 0{,}9$; $\beta_2 = 0{,}99$)	–

Tabela 3: Sumário da CNN-Maior.

3.2 Rede Neural Convolucional Residual

A rede neural convolucional (*Convolutional Neural Network* – CNN) residual proposta é esquematizada na Figura 6. Ela é composta por 10 camadas convolucionais, tendo a tangente hiperbólica como função de ativação. Todos os filtros convolucionais possuem dimensão $(7, 7)$.

À entrada da rede, são alimentadas as imagens degradadas normalizadas entre valores no intervalo $[-1, 1]$. São usadas as imagens inteiras e não apenas regiões da imagem, como nas redes MLP-Adam e CNN-Maior. Assim, uma vez que a função de ativação utilizada é a tangente hiperbólica, a saída da rede corresponde às imagens restauradas também normalizadas entre valores no intervalo $[-1, 1]$. Cabe lembrar que os tensores de entrada de todas as camadas da rede passam pelo processo de *zero padding* [7] para que os tensores de saída sempre apresentem resolução 256×256.

A quantidade de filtros por camada na CNN proposta aumenta gradativamente de 1 para 32 da primeira à quinta camada. Isso faz com que o tensor de saída da quinta camada tenha dimensão $(256, 256, 32)$, o que possibilita extrair diferentes características da imagem degradada, facilitando o processo de restauração. Da quinta à décima camada, a quantidade de filtros por camada diminui de 32 para 1. Assim, o tensor de saída da última camada representa a imagem restaurada com dimensão $(256, 256, 1)$, exatamente igual à dimensão da imagem degradada.

Além disso, considerou-se uma rede do tipo residual, em que existem atalhos entre camadas não consecutivas a fim de otimizar o processo de aprendizado da rede [9]. No caso da CNN proposta, como esquematizado na Figura 6, a entrada da décima camada é dada pela soma das saídas da nona com as saídas da primeira camada, a entrada da nona camada é dada pela soma das saídas da oitava com as saídas da segunda camada e assim sucessivamente.

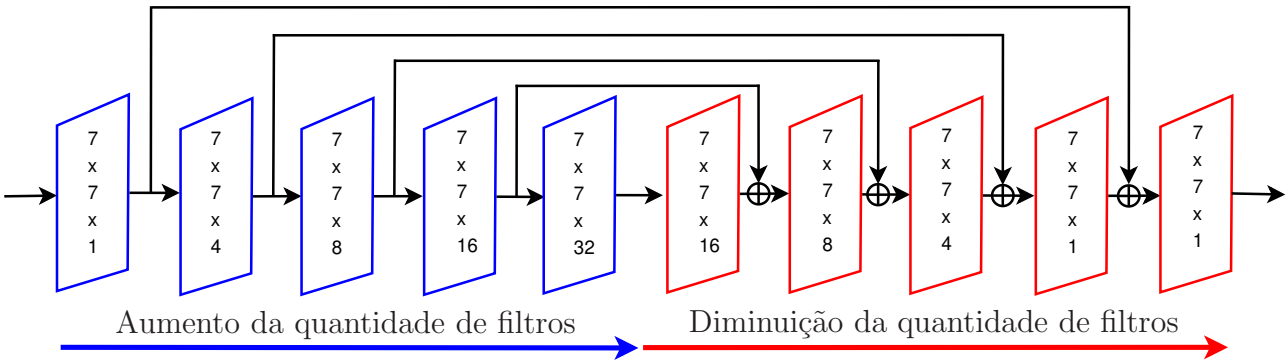


Figura 6: Esquema da CNN residual utilizada; quantidade de filtros 7×7 indicadas.

Neste relatório são mencionadas duas CNNs residuais diferentes. A primeira utiliza como função custo o Erro Quadrático Médio (*Mean Squared Error* – MSE), sendo denominada

Residual-MSE, e a segunda utiliza uma função custo baseada no índice SSIM dada por

$$J = 1 - \text{SSIM}(\mathbf{Y}, \mathbf{D}), \tag{9}$$

em que $\mathbf{Y}$ é a imagem restaurada e $\mathbf{D}$ é a imagem original. Utilizou-se a diferenciação automática (*autodiff*) [10] para determinar os gradientes dessa função custo. Essa segunda rede é chamada aqui de Residual-SSIM.

A grande vantagem dessa arquitetura quando comparada com as redes MLP-Adam e CNN-Maior é sua maior versatilidade em aplicações práticas. Enquanto estas são limitadas a uma certa quantidade de níveis de cinza intrínseca às redes, a CNN residual não possui essa limitação, sendo facilmente aplicada a problemas envolvendo grandes quantidades de gradações possíveis de cinza e até mesmo em restaurações de imagens coloridas.

3.3 Mistura de Especialistas

O propósito de uma mistura de especialistas é combinar em um mesmo sistema várias redes neurais especializadas na resolução de problemas específicos, sendo chamadas de redes especialistas. Logo, a mistura de especialistas pode ser usada para a resolução de tarefas mais gerais, como restaurar imagens degradadas por mais de um tipo de degradação [11]. Na Figura 7, é esquematizado o modelo de mistura de especialistas usado.

No caso, foram consideradas apenas duas redes especialistas na mistura, Especialista 1 e Especialista 2, previamente treinadas. Dessa maneira, o sistema pode ser aplicado a até duas classes distintas de degradação de imagens. A mistura de especialistas recebe em sua entrada a imagem degradada e define dois coeficientes, a_1 e a_2, que serão usados para realizar uma combinação linear convexa entre as saídas de ambas as redes especialistas. Ou seja, a imagem restaurada $\mathbf{Y}_m$ é dada por

$$\mathbf{Y}_m = a_1\mathbf{Y}_1 + a_2\mathbf{Y}_2, \tag{10}$$

em que $a_1 + a_2 = 1$, $0 \leq a_1, a_2 \leq 1$, e $\mathbf{Y}_1$ e $\mathbf{Y}_2$ são as saídas do Especialista 1 e Especialista 2, respectivamente.

A mistura de especialistas também é implementada como uma rede neural com duas camadas convolucionais após a entrada: a primeira camada com 64 filtros de dimensão $(5, 5)$ e a segunda com 128 filtros de dimensão $(5, 5)$, ambas as camadas apresentam um *stride* $s = 2$ [7]. Na saída das duas camadas convolucionais, é aplicada a função ReLU como ativação e a técnica de *dropout* com $p = 0{,}3$ [12].

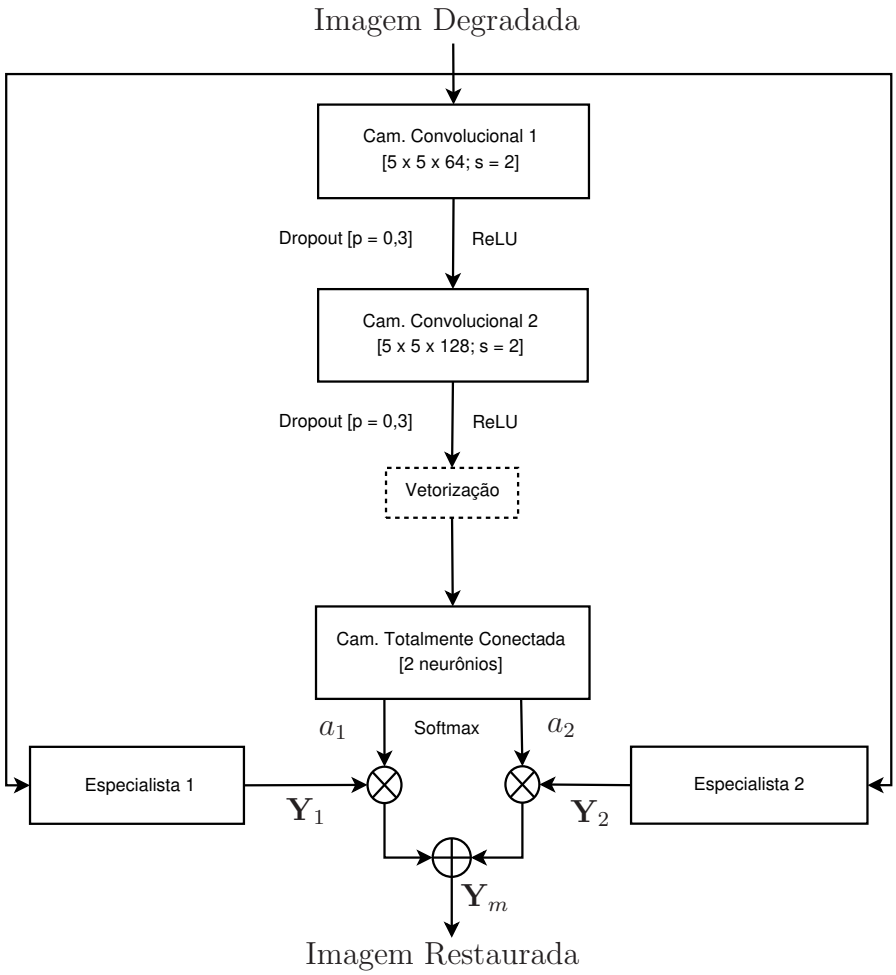


Figura 7: Modelo de mistura de especialistas usado.

O tensor de saída da segunda camada convolucional é posteriormente transformado em um vetor unidimensional e alimentado a uma camada densa totalmente conectada. A saída da camada totalmente conectada é composta por dois neurônios, representando a_1 e a_2. Como a função de ativação desta camada é a função *softmax*, garante-se que $a_1 + a_2 = 1$, com a_1 e a_2 não negativos. Finalmente, a saída da mistura de especialistas é dada pela Equação (10).

Para o treinamento da mistura de especialistas é usada a mesma função custo relacionada ao índice SSIM expressa na Equação (9). Vale ressaltar que os parâmetros das redes especialistas não são alterados, apenas os parâmetros das camadas responsáveis por determinar a_1 e a_2 são considerados no treinamento.

3.4 Rede Gerativa Adversária

Uma rede gerativa adversária (*Generative Adversarial Networks* – GAN) possui dois componentes principais: um gerador e um discriminador [2]. Para a restauração de imagens, o gerador representa a rede restauradora em si e o discriminador representa uma rede que busca diferenciar corretamente imagens originais de restaurações feitas pelo gerador [13]. Na Figura 8, temos

o diagrama da GAN utilizada para restauração de imagens.

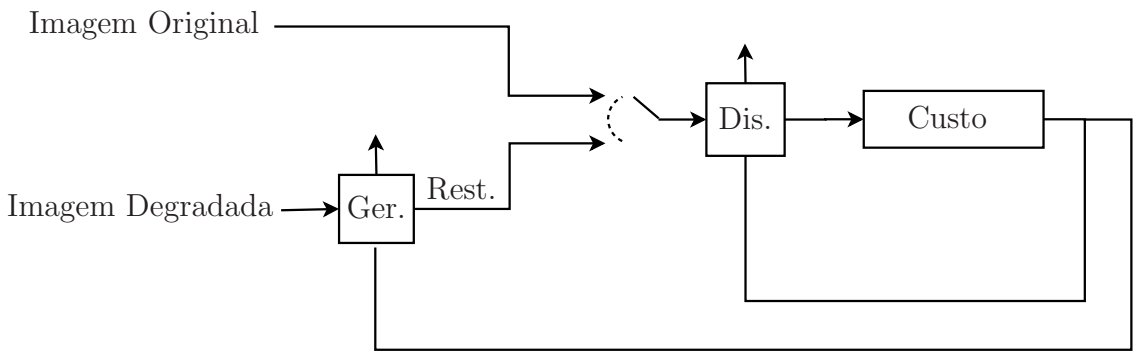


Figura 8: Modelo de rede gerativa adversária.

O objetivo do gerador é enganar o discriminador, isto é, fazer com que suas restaurações sejam classificadas como imagens originais pelo discriminador. Em contrapartida, o objetivo do discriminador é não ser enganado e classificar corretamente imagens originais e restaurações. Devido a esse aspecto de jogo entre o gerador e o discriminador, o treinamento de uma GAN pode se tornar instável caso algum de seus componentes tenha um desempenho muito superior ao outro, não convergindo para a solução do problema. Logo, o ideal é que gerador e discriminador tenham desempenhos similares, tornando o jogo entre eles o mais competitivo possível [2].

Sendo assim, no treinamento da GAN proposta, são consideradas tanto imagens originais quanto imagens degradadas. As imagens degradadas são restauradas pelo gerador e, em seguida, o discriminador busca realizar a distinção entre imagens originais e restaurações. Então, são calculados os valores para duas funções custos diferentes, a primeira leva em conta quantos acertos o discriminador obteve entre imagens originais e restauradas, e a segunda analisa quantos erros de classificação foram cometidos apenas entre imagens restauradas. Ambas as funções custo usam a função de entropia cruzada para realizar essas avaliações. Dessa forma, a primeira função custo está relacionada ao desempenho do discriminador (mede o sucesso das classificações) e é usada para atualizar os parâmetros do mesmo, enquanto a segunda função custo está relacionada ao desempenho do gerador (mede a quantidade de enganos do discriminador) e é usada para atualizar os parâmetros internos do gerador. Nas Figuras 9 e 10, são apresentados esquemas dos treinamentos do discriminador e do gerador, respectivamente.

Vale ressaltar que é possível realizar um treinamento não supervisionado com a arquitetura de uma GAN. Isto é, no caso de restauração de imagens, não é necessário que as imagens degradadas sejam obtidas diretamente a partir de degradações das imagens originais, como era requerido para as outras redes neurais consideradas neste trabalho. É possível treinar a GAN

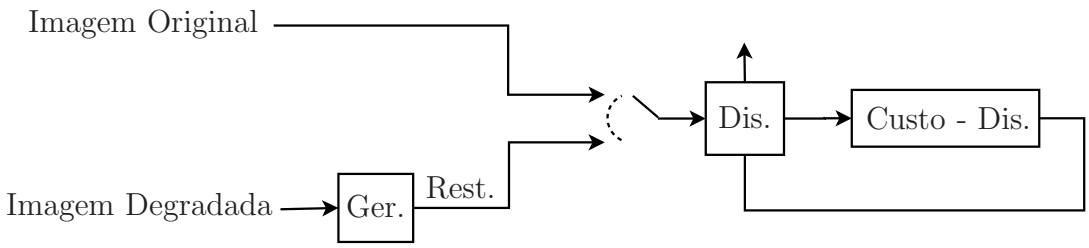


Figura 9: Modelo de treinamento do discriminador.

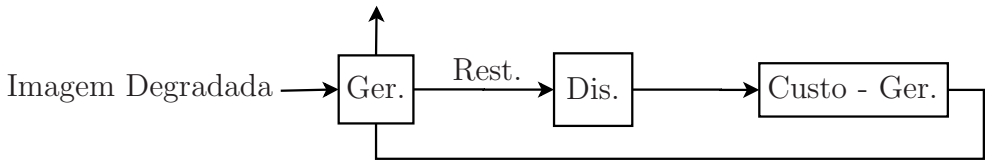


Figura 10: Modelo de treinamento do gerador.

com um conjunto de imagens degradadas sem nenhuma relação com o conjunto de imagens originais. Isso ocorre uma vez que as funções custo da GAN dependem apenas da habilidade do discriminador em diferenciar imagens originais de restauradas (obtidas a partir das imagens degradadas) sem se ter necessariamente uma relação direta entre elas [13]. Contudo, para efeito de comparação, neste trabalho, optou-se por treinar a GAN com o mesmo conjunto de dados usados para as outras redes apresentadas.

O discriminador é implementado como um rede neural de duas camadas convolucionais seguidas por uma camada densa totalmente conectada. A arquitetura escolhida é muito similar à arquitetura da rede usada na mistura de especialistas para gerar os coeficientes a_1 e a_2. A única diferença é que, na saída do discriminador, há apenas um neurônio ativado pela função sigmoidal, indicando 0 para imagens classificadas como originais e 1 para imagens classificadas como restaurações.

Como o desempenho do discriminador e do gerador precisam ser comparáveis para se ter um treinamento equilibrado, a rede restauradora usada como gerador precisou ser mais complexa que as outras redes restauradoras listadas neste relatório, uma vez que a tarefa de discriminação é significativamente mais simples que a tarefa de restauração. Dessa maneira, foi considerado o gerador usado em [13], em que também foi proposta uma arquitetura de GAN para restauração de imagens. A rede em questão é formada por 9 blocos residuais e possui 24 camadas convolucionais, sendo cada bloco residual composto por duas camadas convolucionais. Além disso, ao invés do *zero padding* para as operações de convolução é usado o *reflection padding*, em que as bordas dos tensores de saída das camadas convolucionais são preenchidas a partir da reflexão dos valores nas extremidades desses tensores sobre as áreas a serem completadas. Na Figura 11, é mostrado um esquema simplificado do gerador usado.

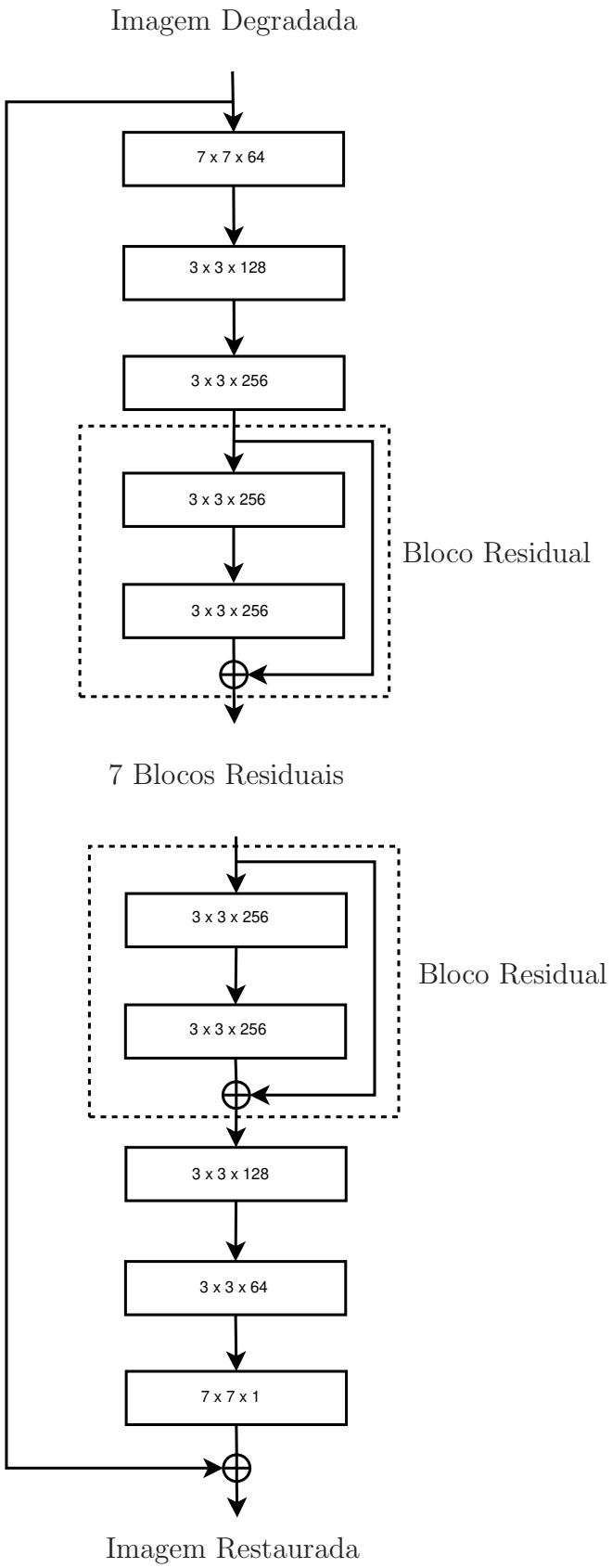


Figura 11: Esquema simplificado do gerador usado para a GAN. Adaptado de [13].

4 Resultados

Nesta Seção, são explorados os resultados obtidos para os problemas enunciados na Seção 2. O conjunto de dados usados para as redes MLP-Adam, CNN-Maior e para a GAN era composto por 20 imagens para treinamento e 1 imagem para teste, todas em 256 níveis de cinza antes do preprocessamento específico para cada problema.

Para as CNNs residuais, foi considerado esse mesmo conjunto de dados com 21 imagens para o Problema dos 8 Níveis de Cinza e para o Problema dos 256 Níveis de Cinza. Para a restauração de imagens coloridas, foi usado um conjunto de dados com 29 imagens coloridas para treino e 3 imagens coloridas para teste, todas com 256 gradações para cada canal de cor.

Finalmente, para a mistura de especialistas, foi considerado um conjunto maior de imagens em 256 níveis de cinza, com 196 imagens para treinamento. Para o teste, foram usadas as mesmas 21 imagens em cinza do conjunto de dados mencionado anteriormente. Foi escolhido um conjunto de imagens maior para esta última arquitetura, uma vez que o seu desempenho para um treinamento com conjunto de apenas 21 imagens em cinza não foi satisfatório. Todos os conjuntos de dados usados neste relatório estão listados no repositório público *https://github.com/gui-lizarza/image-deblurring-neural-nets*.

4.1 Problema dos 8 Níveis de Cinza

Para o Problema dos 8 Níveis de Cinza, utilizaram-se primeiramente as arquiteturas das redes MLP-Adam e CNN-Maior, experimento já realizado no Relatório Parcial. Após a averiguação do desempenho dessas arquiteturas, utilizou-se duas CNNs residuais (Residual-MSE e Residual-SSIM) para o mesmo problema.

A degradação das imagens foi obtida a partir de uma PSF gaussiana com $\sigma = 2$ e filtro de dimensão $(7, 7)$.

MLP-Adam e CNN-Maior

Os resultados obtidos foram compatíveis com os resultados expostos no projeto de Iniciação Científica citado previamente [8]. Após 100 épocas de treinamento, a partir da imagem degradada do conjunto de teste, com índice SSIM de 0,77 em relação à imagem original, obteve-se uma imagem restaurada com índice SSIM de 0,84 tanto para a rede MLP-Adam quanto para a CNN-Maior. Tais resultados são expostos na Figura 12. Na Figura 13, são expostos os gráficos das funções custo de ambas as redes ao longo das 100 épocas de treinamento.

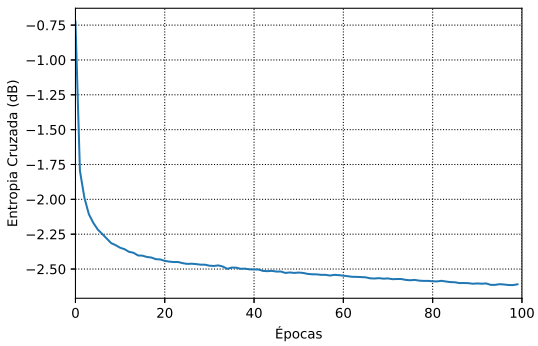
(a) Original / SSIM = 1

(b) Degradada / SSIM = 0,77

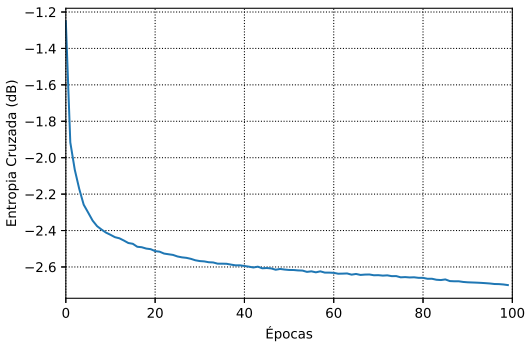
(c) MLP-Adam / SSIM = 0,84

(d) CNN-Maior / SSIM = 0,84

Figura 12: Restaurações para imagens em 8 níveis de cinza (imagens restauradas em (c) e (d)). Redes utilizadas foram MLP-Adam e CNN-Maior. Imagem degradada por PSF gaussiana com $\sigma = 2$ e filtro de dimensão $(7, 7)$.



(a) MLP-Adam



(b) CNN-Maior

Figura 13: Gráficos das funções custo de ambas as redes ao longo de 100 épocas de treinamento.

A melhoria na qualidade das imagens é nítida e o desempenho de ambas as redes foi similar. Entretanto, vale ressaltar que numa aplicação prática do problema, os valores dos *pixels* das imagens degradadas são inteiros, algo que não foi considerado neste experimento. Nessas condições, a tarefa de restauração seria mais complexa e o resultado não seria tão bom. O truncamento de *pixels* para valores inteiros foi realizado em problemas envolvendo 256 níveis de cinza.

CNN Residual

As CNNs residuais consideradas para o Problema dos 8 Níveis de Cinza foram a rede Residual-MSE e a rede Residual-SSIM, explicadas na Seção 3.2. Ambas as redes superaram, em termos de índice SSIM, os resultados obtidos com as redes MLP-Adam e CNN-Maior. Após 20000 épocas de treinamento, a partir da imagem degradada do conjunto de teste, com índice SSIM de 0,77 em relação à imagem original, gerou-se uma imagem restaurada com índice SSIM de 0,88 para a rede Residual-MSE e uma imagem restaurada com índice SSIM de 0,90 para a rede Residual-SSIM. Tais resultados são mostrados na Figura 14. Na Figura 15, são expostos os gráficos das funções custo de ambas as redes ao longo das 20000 épocas de treinamento.

A razão por serem consideradas 20000 épocas de treinamento para as CNNs residuais, enquanto apenas 100 épocas para as redes MLP-Adam e CNN-Maior, se deve ao fato da época de treinamento de uma CNN residual necessitar de menos tempo e custo computacional. Como a entrada das redes MLP-Adam e CNN-Maior não é composta pela imagem completa com resolução 256×256, mas por regiões 7×7 da imagem, o conjunto de treinamento dessas redes apresenta milhares de amostras, enquanto o conjunto de treinamento das CNNs residuais apresenta apenas 20 amostras, cada uma representando uma imagem 256×256 completa. Para se acelerar o tempo de treinamento das CNNs residuais, também se optou pela utilização de unidades de processamento do tipo GPU.

Como é possível notar, as CNNs residuais alcançaram maiores valores para o índice SSIM em comparação ao alcançado pelas redes MLP-Adam e CNN-Maior, mostrando uma capacidade de aprendizado superior. Ademais, quando considerada uma função custo baseada no índice SSIM, o desempenho da CNN residual também melhorou quando comparado ao desempenho da mesma rede com o MSE como função custo.

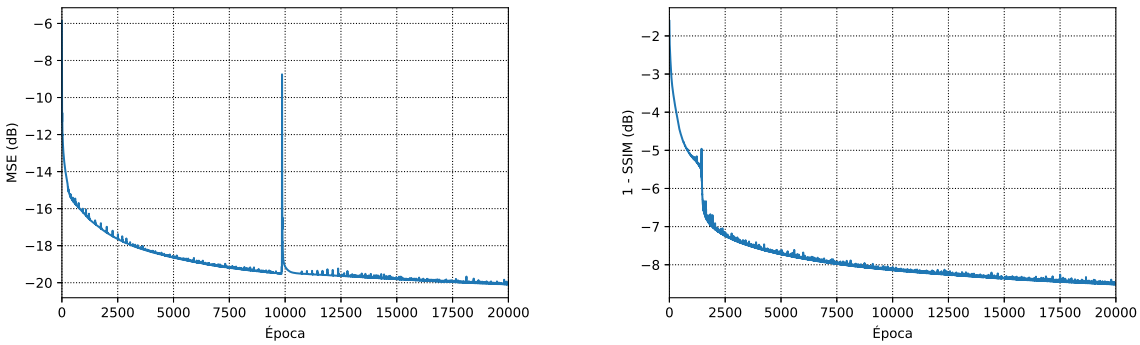
O melhor desempenho visual da rede Residual-SSIM corrobora com o observado em [4], que considera o MSE não adequado para tarefa de restauração de imagens. Além disso, vale a pena comentar que não é possível comparar os valores de função custo das duas redes, uma vez que mesmo alcançando -20dB, a rede Residual-MSE obteve uma restauração pior em termos de SSIM que a rede Residual-SSIM, cujo valor mínimo da função custo ficou em torno de -9dB.

O pico observado no gráfico da função custo da rede Residual-MSE ocorreu devido a um passo de aprendizado elevado. A instabilidade, porém, foi logo corrigida.

(a) Original / SSIM = 1 (b) Degradada / SSIM = 0,77

(c) R.-MSE / SSIM = 0,88 (d) R.-SSIM / SSIM = 0,90

Figura 14: Restaurações para imagens em 8 níveis de cinza (imagens restauradas em (c) e (d)). Redes utilizadas foras Residual-MSE e Residual-SSIM.



(a) Residual-MSE (b) Residual-SSIM

Figura 15: Gráficos das funções custo de ambas as redes ao longo de 20000 épocas de treinamento.

4.2 Problema dos 256 Níveis de Cinza

Após explorar o Problema dos 8 Níveis de Cinza, as CNNs residuais foram aplicadas para a restauração de imagens em 256 níveis de cinza. Também foi aplicada a GAN para a resolução deste problema.

Como os 256 níveis de cinza são capazes de fornecer mais informação à rede que apenas 8 níveis, a tarefa de restauração de imagens em 256 níveis de cinza, além de mais próxima de uma

situação cotidiana, também apresenta potencial de resultados melhores do que os apresentados até aqui. As imagens degradadas em 256 níveis de cinza eram obtidas a partir da aplicação sobre as imagens originais de uma PSF gaussiana com filtro de dimensão $(7, 7)$ e $\sigma = 2$, sendo posteriormente truncadas em valores inteiros no intervalo $[0, 255]$.

CNN Residual

A CNN residual escolhida para a resolução do Problema dos 256 Níveis de Cinza foi a rede Residual-SSIM, com os mesmos hiperparâmetros utilizados para a resolução do Problema dos 8 Níveis de Cinza. Os resultados obtidos após 20000 épocas de treinamento são expostos na Figura 16. Na Figura 17, é mostrado o gráfico dos valores da função custo da rede ao longo das épocas de treinamento.

A rede Residual-SSIM obteve um desempenho ainda melhor para degradações com $\sigma = 2$ quando comparado ao Problema dos 8 Níveis de Cinza, obtendo um índice SSIM para sua restauração de 0,97 em relação à imagem original.

(a) Original / SSIM = 1 (b) Degr. $\sigma = 2$ / SSIM = 0,89 (c) Rest. $\sigma = 2$ / SSIM = 0,97

Figura 16: Restauração para imagem degradada por filtro $\sigma = 2$ pela rede Residual-SSIM.

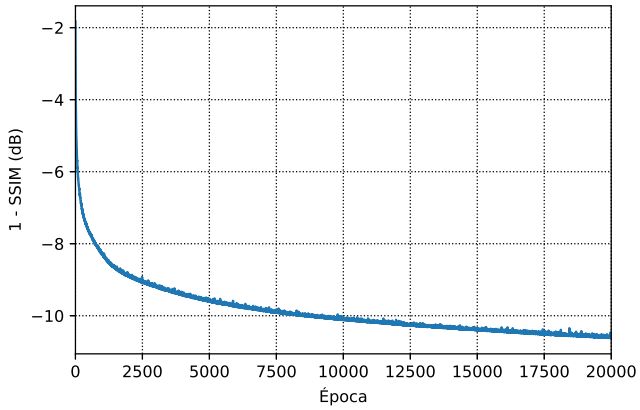


Figura 17: Gráfico da função custo da rede Residual-SSIM ao longo de 20000 épocas de treinamento.

Assim, de maneira a mensurar a capacidade de generalização dessa rede para outros tipos de degradação, foram analisadas as restaurações do modelo para imagens degradadas por filtros gaussianos com outros valores de σ. No caso, foram analisadas degradações gaussianas para filtros de dimensão $(7, 7)$ com $\sigma = 1$ e $\sigma = 3$. Na Figura 18, são expostas as restaurações realizadas. Como é possível notar, a rede não teve o mesmo desempenho na restauração de imagens degradadas por PSFs com outros valores de σ, piorando significativamente a qualidade da imagem degradada com $\sigma = 1$ e mantendo o mesmo índice SSIM em relação à imagem original para a degradação com $\sigma = 3$.

Portanto, o treinamento estritamente focado em um tipo de filtro gaussiano se mostrou insatisfatório para aplicações mais gerais de restauração de imagens. Na Subseção 4.3, é realizado um outro treinamento da rede Residual-SSIM considerando mais de um tipo de filtro gaussiano no conjunto de treinamento. Sendo assim, é possível averiguar se essa pobre capacidade de generalização é devida à arquitetura da rede em si ou ao conjunto de treinamento pouco abrangente.

(a) Original / SSIM = 1 (b) Degr. $\sigma = 1$ / SSIM = 0,96 (c) Rest. $\sigma = 1$ / SSIM = 0,38

(d) Original / SSIM = 1 (e) Degr. $\sigma = 3$ / SSIM = 0,86 (f) Rest. $\sigma = 3$ / SSIM = 0,86

Figura 18: Restauração para imagens degradadas por filtros com $\sigma = 1$ e $\sigma = 3$ pela rede Residual-SSIM. A rede foi treinada apenas $\sigma = 2$.

GAN

Na Figura 19, é mostrada a restauração realizada pela GAN para a imagem do conjunto de teste após um treinamento de 10000 épocas. É interessante notar como o índice SSIM associado à restauração foi igual ao índice SSIM associado à imagem degradada (0,89), mesmo com a qualidade visual da restauração sendo nitidamente melhor do que a qualidade da imagem degradada. Vale ressaltar também que o MSE associado à restauração (0,0033) chegou inclusive a ser maior que o MSE associado à imagem degradada (0,0028).

Como o treinamento da GAN é voltado a promover um jogo entre gerador e discriminador, desconsiderando qualquer índice específico, como o MSE ou o índice SSIM, nota-se que pode ocorrer uma melhoria da qualidade visual da imagem restaurada sem ser observado um aumento significativo em determinadas medidas de qualidade. Esse fenômeno aponta como o índice SSIM, mesmo sendo mais adequado que o MSE para tais tarefas, ainda não captura alguns elementos do sistema visual humano, indicando valores similares para imagens com diferentes níveis de nitidez.

Outra observação importante a ser feita é que, apesar de apresentar uma arquitetura mais complexa, o gerador da GAN não obteve o mesmo desempenho quando comparado com a rede Residual-SSIM, seja em termos de índice SSIM ou mesmo em termos de qualidade visual perceptível. Supõe-se que este fato se deve ao processo de treinamento extremamente delicado de uma GAN, exigindo equilíbrio entre gerador e discriminador, em que calibrar seus hiperparâmetros é uma tarefa significativamente mais desafiadora do que calibrar os hiperpârametros das outras redes apresentadas no relatório.

(a) Original / SSIM = 1 (b) Degr. $\sigma = 2$ / SSIM = 0,89 (c) Rest. $\sigma = 2$ / SSIM = 0,89

Figura 19: Restauração para imagem degradada por filtro $\sigma = 2$ pela GAN proposta.

Na Figura 20, é mostrada a evolução dos valores das funções custo da GAN ao longo das 10000 épocas de treinamento. Como é possível observar, os valores da função custo associada ao discriminador geralmente são menores do que os valores da função custo associada ao gerador,

indicando certo desequilíbrio no treinamento. Os resultados da GAN, portanto, ainda apresentam potencial de melhora caso se faça um estudo mais aprofundado sobre o discriminador e gerador usados, o que se pretende abordar em um trabalho futuro.

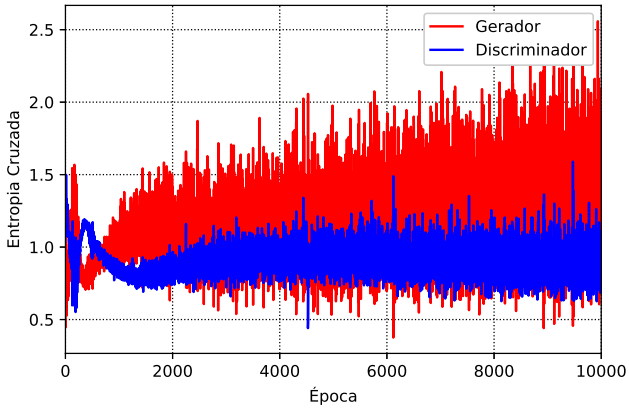


Figura 20: Gráfico das funções custo da GAN ao longo de 10000 épocas de treinamento.

4.3 Restauração para Vários Valores de σ

Como forma de mitigar a deficiência de generalização da rede Residual-SSIM usada para o Problema dos 256 Níveis de Cinza, foi realizado um novo treinamento para essa arquitetura, desta vez considerando amostras degradadas por PSFs com valores variados de σ. No caso, foram incluídas no conjunto de treinamento 20 imagens degradadas por um filtro com $\sigma = 1$ e 20 imagens degradadas por um filtro com $\sigma = 3$. Como é importante que a rede Residual-SSIM também consiga restaurar imagens degradadas por outros tipos de degradação não considerados no treinamento, filtros com $\sigma = 2$ não foram usados no treinamento da rede, apenas no teste. A dimensão de todos os filtros gaussianos considerados ainda era $(7, 7)$.

Novamente, após 20000 épocas de treinamento, foi observado o desempenho da rede para degradações com $\sigma = 1$, $\sigma = 2$ e $\sigma = 3$. Na Figura 21, são expostos esses resultados. Na Figura 22, é mostrada a evolução dos valores da função custo da rede ao longo das épocas de treinamento.

A rede Residual-SSIM foi capaz de restaurar satisfatoriamente tanto imagens degradadas por PSFs com valores de σ usados no treinamento ($\sigma = 1$ e $\sigma = 3$) quanto imagens degradadas por valores de σ não incluídos no conjunto de treino ($\sigma = 2$). Mesmo com a rede Residual-SSIM não obtendo o mesmo índice SSIM de 0,97 para a degradação com $\sigma = 2$ observado na Subseção 4.2, é constatado que, com um conjunto de treinamento mais abrangente, o poder de generalização da rede para degradações não incluídas no treino é significativamente maior.

(a) Original / SSIM = 1 (b) Degr. $\sigma = 1$ / SSIM = 0,96 (c) Rest. $\sigma = 1$ / SSIM = 0,98

(d) Original / SSIM = 1 (e) Degr. $\sigma = 2$ / SSIM = 0,89 (f) Rest. $\sigma = 2$ / SSIM = 0,92

(g) Original / SSIM = 1 (h) Degr. $\sigma = 3$ / SSIM = 0,86 (i) Rest. $\sigma = 3$ / SSIM = 0,95

Figura 21: Restauração para imagens degradadas por filtros com $\sigma = 1$, $\sigma = 2$ e $\sigma = 3$ pela rede Residual-SSIM. A rede foi treinada apenas para $\sigma = 1$ e $\sigma = 3$.

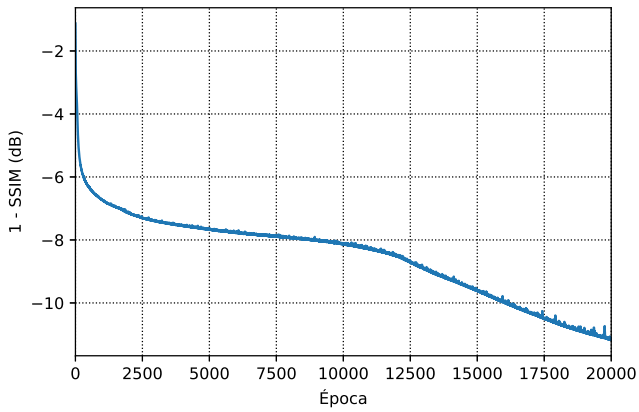


Figura 22: Gráfico da função custo da rede Residual-SSIM ao longo de 20000 épocas de treinamento.

Cabe lembrar que o valor mínimo da função custo deste treinamento acabou sendo similar ao valor mínimo alcançado para o conjunto de treinamento contendo apenas $\sigma = 2$ (-11dB).

Adaptação para Imagens Coloridas

O último teste realizado com as CNNs residuais foi a adaptação da Restauração para Vários Valores de σ para o tratamento de imagens coloridas. As imagens coloridas consideradas possuíam 3 canais de cores vermelho, verde e azul com resolução 256×256 cada, sendo assim os tensores de entrada e saída da rede agora tinham dimensão $(256, 256, 3)$. Logo, a arquitetura da rede Residual-SSIM usada até aqui precisou passar por pequenas modificações para a resolução deste novo problema. A primeira e as duas últimas camadas da rede passaram a apresentar 3 filtros convolucionais de dimensão $(7, 7)$ ao invés de um único filtro convolucional como é mostrado na Figura 6. Essas mudanças foram feitas para que a rede tivesse como entrada e saída tensores de dimensão $(256, 256, 3)$.

As imagens coloridas degradadas foram obtidas a partir da aplicação de uma PSF gaussiana com filtro de dimensão $(7, 7)$ em cada um dos canais de cor das imagens originais, truncando o valor final de cada elemento do tensor resultante em valores inteiros no intervalo $[0, 255]$. O índice SSIM entre imagens coloridas foi considerado como sendo a média entre os índices SSIM calculados entre cada canal de cor das imagens.

Para o conjunto de treinamento da rede Residual-SSIM, foram consideradas 29 imagens degradadas por um filtro gaussiano com $\sigma = 3$ e 29 imagens degradadas por um filtro gaussiano com $\sigma = 5$. Dessa forma, a rede poderia ser generalizada para a restauração de imagens degradadas por variados valores de σ. Como feito previamente, após um treinamento de 20000 épocas, testou-se o desempenho da rede tanto para valores de σ presentes no conjunto de treinamento ($\sigma = 3$ e $\sigma = 5$) quanto para valores de σ não incluídos no treinamento ($\sigma = 4$). Na Figura 23, são expostos esses resultados de restauração para as três imagens que compunham o conjunto de teste. Na Figura 24, é mostrado o gráfico da evolução dos valores da função custo da rede ao longo das épocas de treinamento.

A partir desses resultados, observou-se que a rede Residual-SSIM também apresentou desempenho satisfatório para a restauração de imagens coloridas. Os índices SSIM de todas as imagens restauradas em relação às imagens originais foram significativamente maiores do que os índices SSIM relacionados às imagens degradadas. Outra observação importante sobre o desempenho geral da arquitetura proposta é que a rede Residual-SSIM apresentou restaurações adequadas tanto para valores de σ mais elevados, como os praticados para a restauração de

(a) Original / SSIM = 1 (b) Degr. $\sigma = 3$ / SSIM = 0,72 (c) Rest. $\sigma = 3$ / SSIM = 0,82

(d) Original / SSIM = 1 (e) Degr. $\sigma = 4$ / SSIM = 0,88 (f) Rest. $\sigma = 4$ / SSIM = 0,94

(g) Original / SSIM = 1 (h) Degr. $\sigma = 5$ / SSIM = 0,63 (i) Rest. $\sigma = 5$ / SSIM = 0,87

Figura 23: Restauração para imagens degradadas por filtros com $\sigma = 3$, $\sigma = 4$ e $\sigma = 5$ pela rede Residual-SSIM. A rede foi treinada apenas para $\sigma = 3$ e $\sigma = 5$.

imagens coloridas, quanto para valores de σ menores, como os praticados para a restauração de imagens em nível de cinza.

Por fim, de maneira a compreender melhor como a rede Residual-SSIM realiza a restauração de imagens, foram colhidas as saídas da quinta camada da rede, composta por 32 matrizes de dimensão $(256, 256)$. A quinta camada da rede é a camada com mais filtros convolucionais, extraindo variados aspectos da imagem degradada para a tarefa de restauração. Após a quinta camada, a quantidade de filtros convolucionais por camada vai diminuindo gradativamente de forma com que, a partir dos aspectos extraídos pela quinta camada, a saída da última camada da rede corresponda à imagem restaurada. Na Figura 25, são expostas as matrizes de saída da quinta camada da rede, em forma de imagens em nível de cinza, após o treinamento para a

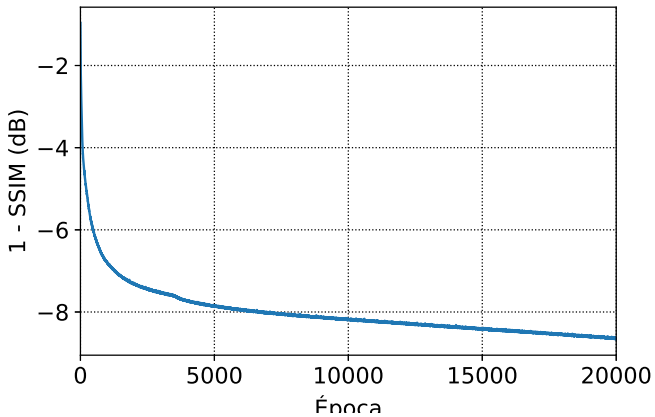


Figura 24: Gráfico da função custo da rede Residual-SSIM ao longo de 20000 épocas de treinamento.

restauração de imagens coloridas. No caso, a restauração apresentada é da imagem degradada com $\sigma = 4$ da Figura 23.

Como é possível observar, em todas as matrizes de saída da quinta camada da rede, ainda é possível reconhecer o rosto da imagem a ser restaurada, com nenhuma saída apresentando distorções que deixem a imagem irreconhecível. Apesar de se perceber modificações nas cores e texturas da imagem degradada, curiosamente não é possível observar sinais claros do processo de restauração em nenhuma das matrizes.

4.4 Restauração para Várias Dimensões de Filtros

Para a tarefa de restauração de imagens degradadas por filtros de dimensões variadas, foram considerados dois tipos de PSFs gaussianas, o primeiro tipo possuía $\sigma = 3$ e filtro com dimensão $(14, 2)$, e o segundo tipo possuía $\sigma = 3$ e filtro com dimensão $(2, 14)$. O primeiro filtro considerado gerava uma distorção principalmente na direção vertical, enquanto o segundo filtro gerava uma degradação na direção horizontal.

Para a resolução deste problema, foi escolhido o método da mistura de especialistas, a primeira rede especialista foi treinada para imagens degradadas pela PSF de degradação vertical (Especialista 1) e a segunda rede especialista foi treinada para imagens degradadas pela PSF de degradação horizontal (Especialista 2). Ambas as redes especialistas apresentavam uma arquitetura simplificada da rede Residual-SSIM, com apenas 8 camadas.

Como mencionado anteriormente, o conjunto de imagens considerado para treinamento da mistura de especialistas foi diferente do conjunto de imagens usado nas outras redes neurais apresentadas neste relatório. Para se obter melhores resultados, foi usado um novo conjunto

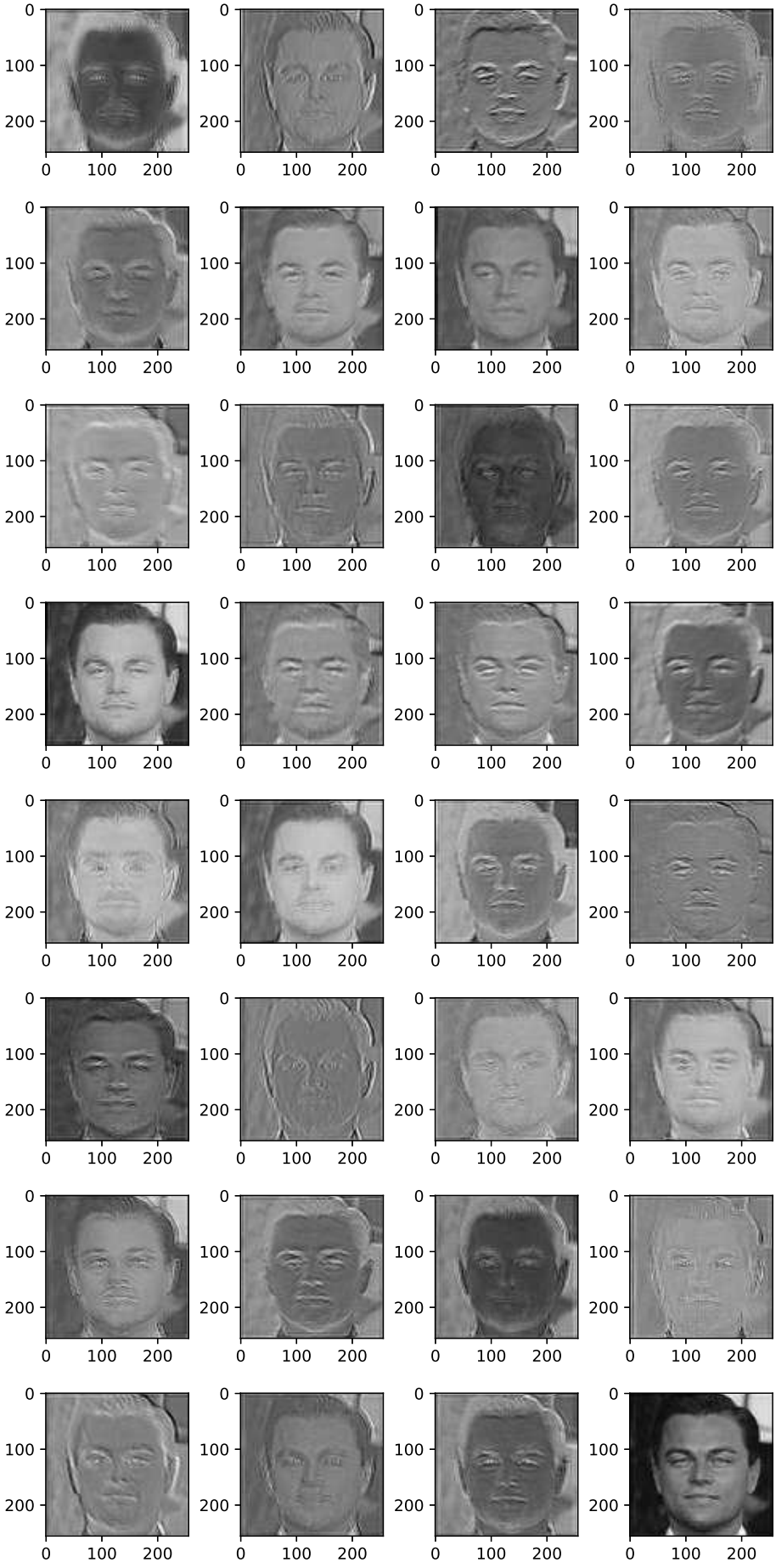


Figura 25: Saídas da quinta camada da rede Residual-SSIM para a restauração de imagens coloridas.

de treinamento com 196 imagens em 256 níveis de cinza, dando mais poder de generalização ao modelo. Como conjunto de teste da mistura de especialistas, foram consideradas as 21 imagens em cinza já usadas para treino e validação de outras redes neurais mostradas no relatório.

Na Figura 26, são mostrados os resultados obtidos com a mistura de especialistas após 10000 épocas de treinamento. É considerada uma imagem do conjunto de teste degradada verticalmente. São expostas tanto as restaurações por cada rede especialista, incluindo seus respectivos coeficientes a_1 e a_2, quanto a combinação linear entre essas restaurações, representando a saída final da mistura de especialistas.

(a) Original / SSIM = 1

(b) Degradação vertical / SSIM = 0,75

(c) Especialista 1 / a_1 = 9,9998E-1

(d) Especialista 2 / a_2 = 1,9474E-5

(e) Combinação linear / SSIM = 0,93

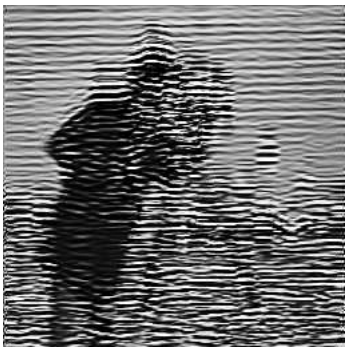
Figura 26: Restauração para imagem com degradação vertical (saída de cada especialista em (c) e (d); saída da mistura de especialistas, como combinação linear entre as especialistas, em (e)). Índices SSIM calculados em relação à imagem original.

Na Figura 27, são mostrados os resultados obtidos com a mistura de especialistas após 10000 épocas de treinamento. É considerada uma imagem do conjunto de teste degradada horizontalmente.

Enfim, na Figura 28, é apresentado o gráfico da função custo da mistura de especialistas ao longo das 10000 épocas de treinamento. Vale ressaltar que a função custo considerada também era baseada no índice SSIM e dada pela Equação (9).

(a) Original / SSIM = 1

(b) Degradação horizontal / SSIM = 0,74



(c) Especialista 1 / a_1 = 1,2390E-2

(d) Especialista 2 / a_2 = 9,8761E-1

(e) Combinação linear / SSIM = 0,93

Figura 27: Restauração para imagem com degradação horizontal (saída de cada especialista em (c) e (d); saída da mistura de especialistas, como combinação linear entre as especialista, em (e)). Índices SSIM calculados em relação à imagem original.

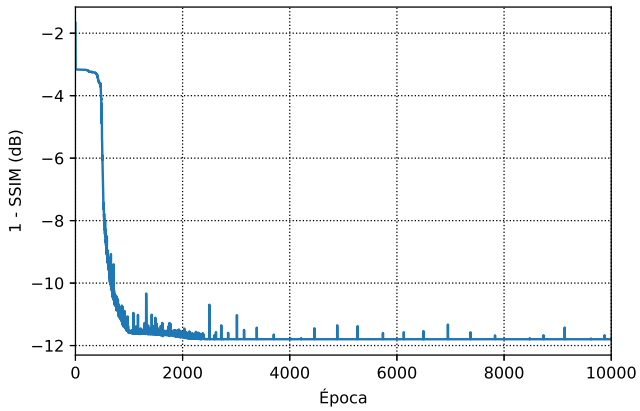


Figura 28: Gráfico da função custo da mistura de especialistas ao longo de 20000 épocas de treinamento.

Pelos valores obtidos para a_1 e a_2, sempre muito próximos de 1 ou 0, percebe-se que a mistura de especialistas considerada funciona na prática como um simples decisor entre as diferentes restaurações feitas por cada rede especialista. Supõe-se que tal fato se deve à grande diferença entre as degradações vertical e horizontal, permitindo que apenas uma rede especialista realize restaurações adequadas para cada tipo de degradação e, consequentemente, que a mistura de

especialistas se comporte apenas como um seletor binário.

Ademais, é possível observar que os valores da função custo da mistura de especialistas ficam estagnados a partir de 2000 épocas de treinamento, reforçando a impressão de que o comportamento da mistura de especialistas como um decisor é, de fato, o mínimo global para o modelo proposto.

5 Conclusões

Os problemas de restauração de imagens abordados neste trabalho foram resolvidos de forma satisfatória a partir dos modelos de redes neurais apresentados. A arquitetura de rede que obteve os melhores resultados foi a CNN residual, mais especificamente, a rede Residual-SSIM. No Problema dos 8 Níveis de Cinza, as CNNs residuais obtiveram maiores índices SSIM do que as redes MLP-Adam e CNN-Maior e, no Problema dos 256 Níveis de Cinza, as CNNs residuais também superaram os resultados apresentados pela arquitetura da GAN.

A rede Residual-SSIM realizou boas restaurações tanto para um conjunto de imagens degradadas por PSFs com filtros de diferentes valores de σ, a partir de um conjunto de treinamento mais abrangente, quanto para um conjunto de imagens degradadas por PSFs com filtros de diferentes dimensões, a partir da arquitetura da mistura de especialistas. Nenhum outro modelo proposto mostrou o mesmo poder de generalização da rede Residual-SSIM.

Contudo, a GAN utilizada ainda apresenta diversos pontos em que podem ser realizadas melhorias de forma a se obter um treinamento mais competitivo entre discriminador e gerador. Em [13], além de ser usado o modelo de gerador considerado para a GAN deste trabalho, também é usado um discriminador mais complexo e funções custo diferentes, que não são baseadas na entropia cruzada e que envolvem pesos de redes pré-treinadas usadas para a classificação de imagens. Sendo assim, estudos posteriores mais aprofundados sobre o funcionamento de GANs mostram grande potencial para a resolução dos problemas tratados no relatório.

Ademais, a avaliação do índice SSIM como medida de similaridade entre imagens é outro ponto a ser discutido. Apesar da CNN residual com função custo baseada no índice SSIM (Residual-SSIM) ter apresentado melhores resultados que a CNN residual com o MSE como função custo (Residual-MSE), o índice SSIM ainda apresentou algumas inconsistências para mensurar a nitidez de determinadas restaurações. Tanto o índice SSIM quanto o MSE não apontaram melhorias visuais para os resultados obtidos com a GAN, mesmo com a restauração sendo perceptivelmente mais nítida do que a imagem degradada. Enfim, ainda que seja mais

adequado que o MSE para tarefas de restauração de imagens, o índice SSIM ainda não capta todos os aspectos do sistema visual humano.

Quanto ao cronograma de atividades previsto, foi possível cumpri-lo dentro do limite de tempo programado mesmo com as dificuldades encontradas ao longo do projeto, tais como o ajuste e desenvolvimento da mistura de especialistas. Um estudo mais aprofundado sobre GANs para a restauração de imagens ficará como trabalho futuro.

Esta iniciação científica trouxe benefícios para algumas disciplinas de graduação, pois reforçou conceitos de Sistemas e Sinais e auxiliou na implementação de algoritmos da disciplina de Cálculo Numérico. Além disso, este trabalho aperfeiçoou o uso das bibliotecas *Tensorflow* e *Keras*, ferramentas muito importantes dentro da área de aprendizado de máquina.

A partir dos resultados obtidos neste trabalho, foi submetido ao *Simpósio Brasileiro de Telecomunicações e Processamento de Sinais* (SBrT 2020) um artigo sobre restauração de imagens coloridas a partir de CNNs residuais na categoria de artigos de Iniciação Científica. Este artigo se encontra anexo ao relatório.

A Fundamentos de Redes Neurais

Redes neurais artificiais, comumente chamadas apenas de "redes neurais", tiveram como primeiro motor de seu desenvolvimento a percepção de que o cérebro humano opera de maneira significativamente diferente de um computador digital. No caso, o cérebro humano pode ser considerado um sistema de processamento de informações altamente complexo e não linear. Ele tem a capacidade de organizar seus componentes estruturais – os neurônios – em configurações variadas de modo a realizar diferentes computações, como o reconhecimento de padrões ou o controle motor de outras partes do corpo [14].

A gênese das redes neurais se deu em meados da década de 1940 nos Estados Unidos, com os trabalhos do neurofisiologista Warren McCulloch e do matemático Walter Pitts [15]. Ambos buscavam esquematizar o funcionamento do sistema nervoso humano por meio de redes neurais descritas por uma série de sentenças lógicas e excitadas pelo meio externo. No modelo proposto, a saída de cada neurônio da rede era binária.

Na década de 1950, o psicólogo americano Frank Rosenblatt concebeu a ideia do *perceptron*, primeiro tipo de rede neural artificial, com apenas uma camada e um neurônio, usado até hoje como classificador linear [16]. O *perceptron* de Rosenblatt foi proposto seguindo o modelo de neurônio de McCulloch e Pitts, ou seja, com a saída do neurônio binária. O objetivo do *perceptron* era classificar os estímulos de entrada x_1, x_2, ..., x_N em duas classes, C_1 e C_2, previamente determinadas e linearmente separáveis. Na Figura 29, esquematiza-se o modelo do *perceptron* de Rosenblatt, com seus pesos w_i e seu nível de ajuste b. Ele implementa uma função não linear abrupta, basicamente um classificador binário, ou seja, sua saída y é dada por [14, 16]

$$y = \begin{cases} 1, & \text{se } \sum_{i=1}^{n} w_i x_i + b \geqslant 0 \\ 0, & \text{caso contrário} \end{cases} . \tag{11}$$

O pioneirismo do *perceptron* de Rosenblatt veio do uso de algoritmos de aprendizado para solucionar o problema da classificação, uma vez que as ligações entre os estímulos e o neurônio eram atualizadas de acordo com os erros entre as classes atribuídas aos estímulos pelo *perceptron* e as verdadeiras classes dos estímulos.

No restante do século XX, as redes neurais sofreram diversos questionamentos relacionados à sua praticidade no mundo real. A pouca sofisticação das arquiteturas de redes neurais à época, aliada a um menor poder computacional disponível e ao difícil acesso a grandes quantidades de dados, não correspondeu às expectativas impostas a essa tecnologia em seus anos iniciais.

Obstáculos técnicos tornavam o desempenho de redes neurais profundas, isto é, com mais

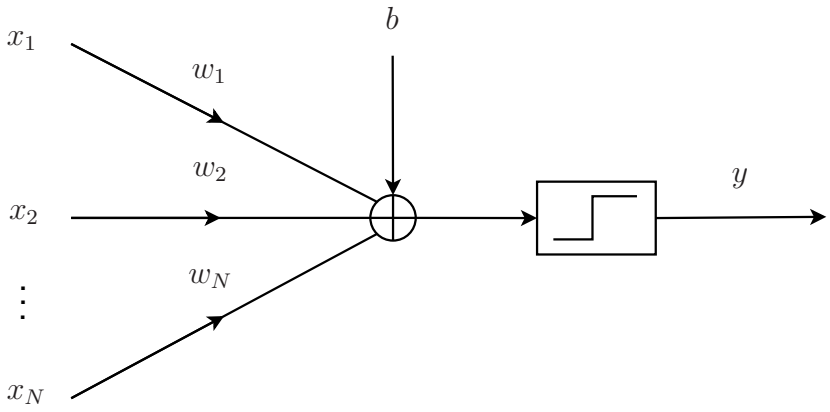


Figura 29: O *perceptron* de Rosenblatt, a primeira rede neural.

de uma camada de neurônios, inferior ao de redes neurais rasas, com apenas uma camada de neurônios. Sendo assim, apenas na década de 2000, com o desenvolvimento de novas arquiteturas e novos métodos de inicialização para os pesos das redes neurais, foi possível atingir resultados satisfatórios com o uso de redes profundas, emprestando novo fôlego ao estudo de redes neurais artificiais.

A.1 Modelo de Neurônio

No contexto de redes neurais artificiais, o neurônio é a unidade fundamental de processamento de informações [14]. Na Figura 30, elucida-se o modelo de um neurônio com seus três componentes principais:

1. Um conjunto de *sinapses*, cada qual associada a um peso sináptico. As sinapses conectam os N elementos do vetor coluna de entrada $\mathbf{x}$, definido por

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \tag{12}$$

   a um somador. Denota-se como w_{ki} o peso sináptico conectando o i-ésimo elemento do vetor de entrada ao neurônio k. Os pesos do neurônio k podem ser reunidos no vetor linha $\mathbf{w}_k$, definido por

$$\mathbf{w}_k = \begin{bmatrix} w_{k1} & w_{k2} & \dots & w_{kN} \end{bmatrix}. \tag{13}$$

2. Um *somador*, usado para somar todos os elementos do vetor de entrada do neurônio k após terem sido multiplicados pelos pesos de suas respectivas sinapses. O somador também inclui em sua soma um *bias* (nível de ajuste) b_k, independente do sinal de entrada.

3. Uma *função de ativação* $\varphi(\cdot)$, usada para introduzir uma não linearidade ao modelo de neurônio. O uso de uma função de ativação não linear permite que a rede neural não se limite apenas a resolver problemas de natureza estritamente linear.

Sendo assim, temos que a saída z_k do somador é dada por

$$z_k = \sum_{i=1}^{N} w_{ki} x_i + b_k = \mathbf{w}_k \cdot \mathbf{x} + b_k. \tag{14}$$

Por fim, a saída y_k do neurônio é dada por

$$y_k = \varphi(z_k). \tag{15}$$

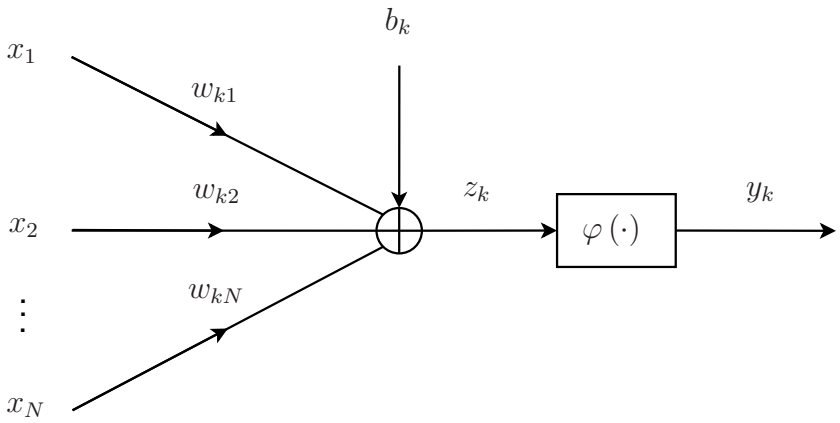
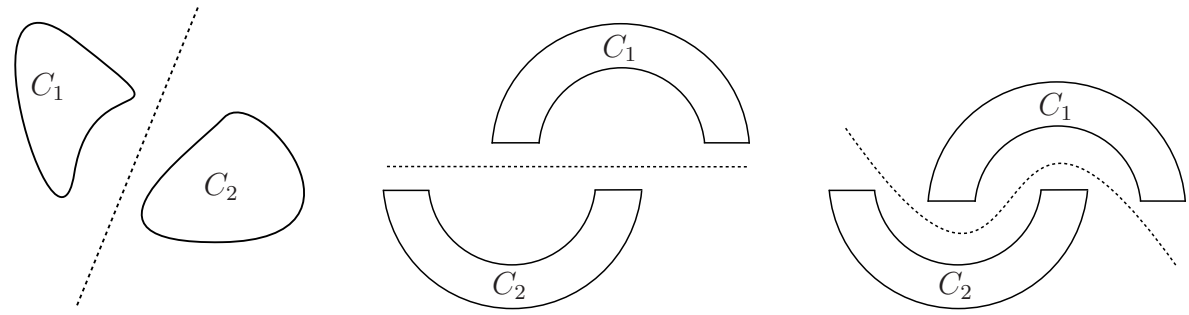


Figura 30: Modelo de um neurônio k, o bloco básico para a construção de uma rede neural.

A.2 Redes *Perceptron* Multicamada

O *perceptron* de Rosenblatt pode ser considerado uma rede neural de apenas uma camada. Por não possuir mais camadas, seu desempenho só é bom em problemas de classificação linear, ou seja, problemas em que as classes podem ser separadas por fronteiras de decisão lineares, como retas (caso bidimensional) ou planos (caso tridimensional) [14]. Na Figura 31, exemplificam-se alguns casos de classes linearmente e não linearmente separáveis para um espaço de entradas bidimensional.

De forma a superar a limitação do *perceptron* de Rosenblatt, as redes *perceptron* multicamada (*Multilayer Perceptron* – MLP) apresentam uma ou mais "camadas ocultas" de neurônios, isto é, camadas intermediárias entre a entrada e a saída da rede neural. Cada neurônio pertencente a uma camada oculta possui sinapses, associadas a pesos sinápticos, ligando-o a todos os neurônios das camadas anterior e posterior [14]. Na Figura 32, esquematiza-se como um grafo direcionado a representação de uma rede MLP com $L = 3$ camadas. Denotando N_j como o



(a) Linearmente separáveis (b) Linearmente separáveis (c) Não linearmente separáveis

Figura 31: Exemplos de classes linear e não linearmente separáveis.

número de neurônios da camada j, é conveniente introduzir uma notação para a configuração da rede. Assim, uma rede MLP de 3 camadas apresenta configuração N_1-N_2-N_3. Especificamente, a rede da Figura 32 tem configuração 3-2-1.

Para calcular a saída da rede MLP a partir do vetor de entrada, é conveniente definir os vetores $\mathbf{b}^{(j)}$, $\mathbf{z}^{(j)}$ e $\mathbf{y}^{(j)}$, que representam respectivamente os *bias*, as saídas dos somadores e as saídas dos N_j neurônios da camada $j = 1,2,\cdots,L$, em que L é o número de camadas da rede, ou seja,

$$\mathbf{b}^{(j)} = \begin{bmatrix} b_1^{(j)} \\ b_2^{(j)} \\ \vdots \\ b_{N_j}^{(j)} \end{bmatrix}, \mathbf{z}^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \vdots \\ z_{N_j}^{(j)} \end{bmatrix} \text{ e } \mathbf{y}^{(j)} = \begin{bmatrix} y_1^{(j)} \\ y_2^{(j)} \\ \vdots \\ y_{N_j}^{(j)} \end{bmatrix}. \tag{16}$$

Definindo agora a matriz de pesos sinápticos associados à camada j como

$$\mathbf{W}^{(j)} = \begin{bmatrix} w_{11}^{(j)} & w_{12}^{(j)} & \cdots & w_{1N_{j-1}}^{(j)} \\ w_{21}^{(j)} & w_{22}^{(j)} & \cdots & w_{2N_{j-1}}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N_j1}^{(j)} & w_{N_j2}^{(j)} & \cdots & w_{N_jN_{j-1}}^{(j)} \end{bmatrix}_{N_j \times N_{j-1}} = \begin{bmatrix} \mathbf{w}_1^{(j)} \\ \mathbf{w}_2^{(j)} \\ \vdots \\ \mathbf{w}_{N_j}^{(j)} \end{bmatrix}, \tag{17}$$

o vetor de saída $\mathbf{z}^{(j)}$ dos somadores da camada j é dado por

$$\mathbf{z}^{(j)} = \mathbf{W}^{(j)} \cdot \mathbf{y}^{(j-1)} + \mathbf{b}^{(j)}, \tag{18}$$

em que $\mathbf{y}^{(0)}$ representa o vetor de entrada da rede. Por fim, o vetor de saída dessa camada é dado por

$$\mathbf{y}^{(j)} = \varphi\left(\mathbf{z}^{(j)}\right), \tag{19}$$

em que a função de ativação $\varphi(\cdot)$ é aplicada de elemento em elemento no vetor $\mathbf{z}^{(j)}$ [17].

A utilização de neurônios ocultos, entretanto, traz alguns obstáculos para o entendimento completo do comportamento das redes neurais. A presença de funções de ativação não lineares no modelo de todos os neurônios e a alta conectividade entre as diferentes camadas tornam

muito difícil uma análise teórica do processo de aprendizado da rede [17]. Cada camada oculta representará um aspecto diferente do sinal de entrada da rede a ser aprendido, sendo o próprio processo de aprendizado responsável pela atribuição desses aspectos às camadas da rede [17,18]. O método mais popular de treinamento de redes neurais é com o algoritmo de retropropagação (*backpropagation*), que será explorado na Subseção A.5.

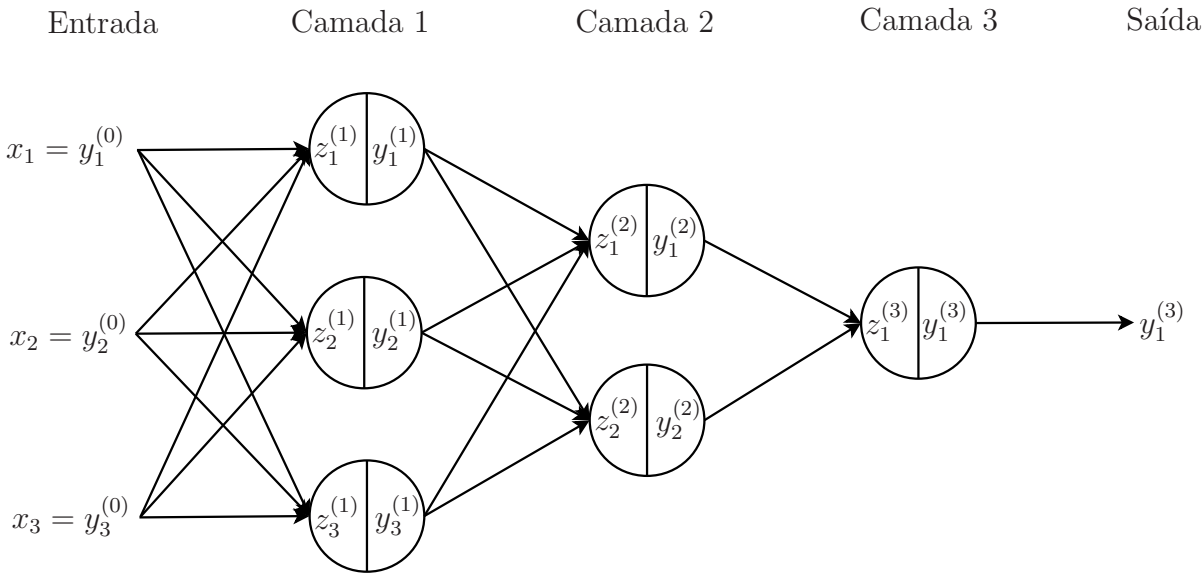


Figura 32: Representação de uma rede MLP com $L = 3$ camadas ($L-1 = 2$ camadas ocultas), cada círculo representa um neurônio.

A.3 Funções de Ativação

As funções de ativação de uma rede neural são essenciais para o aprendizado de tarefas mais complexas e não lineares. Sem as funções de ativação, todas as redes neurais profundas poderiam ser reduzidas a redes de apenas uma camada, capazes apenas de trabalhar com problemas lineares [17].

No caso de uma rede neural sem funções de ativação e com $L = 2$ camadas de neurônios, representadas pelas matrizes de pesos sinápticos $\mathbf{W}^{(1)}$ e $\mathbf{W}^{(2)}$ e também pelos vetores de *bias* $\mathbf{b}^{(1)}$ e $\mathbf{b}^{(2)}$, temos um vetor saída $\mathbf{y}^{(2)}$ igual a

$$\mathbf{y}^{(2)} = \mathbf{W}^{(2)} \cdot (\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} = \mathbf{W}^{(2)}\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}. \tag{20}$$

Dessa forma, essa mesma rede neural poderia ser reduzida a uma rede de apenas uma camada representada pela matriz de pesos sinápticos $\mathbf{W} = \mathbf{W}^{(2)}\mathbf{W}^{(1)}$ e pelo vetor de *bias* $\mathbf{b} = \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)}$.

As funções de ativação também podem ser usadas para limitar a saída de um neurônio entre dois valores reais, algo útil quando, por exemplo, a saída da rede é uma probabilidade

(confinada entre 0 e 1). Ademais, vale ressaltar que, para ser possível aplicar o algoritmo de retropropagação à rede, a função de ativação escolhida precisa ser diferenciável e, consequentemente, contínua. Algumas das principais funções de ativação usadas em redes neurais são [7, 14]:

- Função sigmoidal (ou logística), limitada entre 0 e 1, e definida por

$$\varphi(z) = \frac{1}{1 + e^{-z}}. \tag{21}$$

- Função da tangente hiperbólica, limitada entre -1 e 1, e definida por

$$\varphi(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}. \tag{22}$$

- Função *Rectified Linear Unit* (ReLU), definida por

$$\varphi(z) = \max(0, z). \tag{23}$$

Na Figura 33, são mostrados os gráficos dessas três funções de ativação no intervalo $z \in [-5, 5]$.

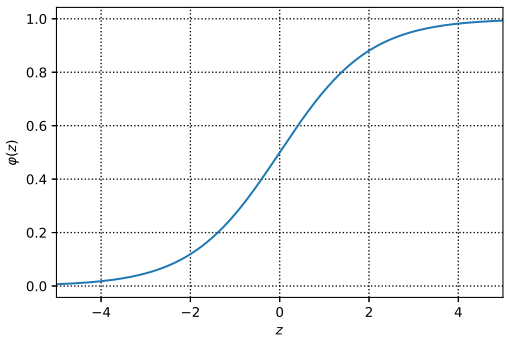
No caso de uma rede com mais de um neurônio em sua última camada, em que a saída esperada é a ativação de apenas um dos neurônios e a inativação dos neurônios restantes, é comum o uso da função de ativação *softmax* na camada de saída da rede [7].

Assim como a função sigmoidal, a função *softmax* limita a saída do neurônio entre 0 e 1, porém, diferentemente daquela, a função *softmax* também leva em consideração os outros neurônios da camada. Dessa forma, considera-se uma normalização fazendo com que a soma de todas as saídas dos neurônios da camada seja unitária, o que faz com que o vetor saída da camada seja um vetor de probabilidades [7]. A função *softmax* para o k-ésimo neurônio da última camada da rede é dada por

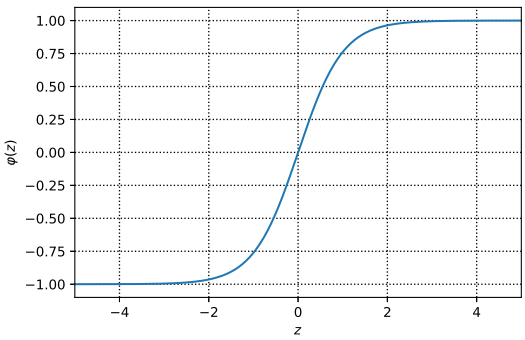
$$\varphi(z_k^{(L)}) = \frac{e^{z_k^{(L)}}}{\sum_{n=1}^{N_L} e^{z_n^{(L)}}}. \tag{24}$$

A.4 Funções Custo

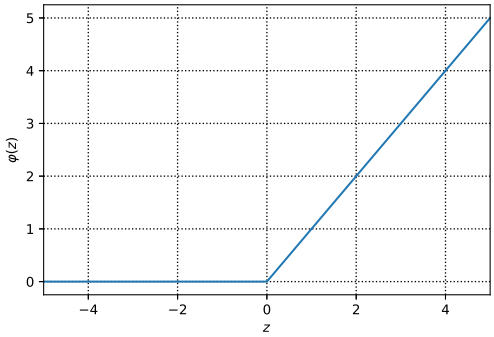
No contexto do processo de aprendizado de uma rede neural, o papel de uma função custo J é quantificar o quão próximos estão o vetor de saída $\mathbf{y}^{(L)}$ da rede a ser estimado e o vetor



(a) Sigmoidal



(b) Tangente hiperbólica



(c) ReLU

Figura 33: Gráficos das funções de ativação.

desejado $\mathbf{d}$, definido por

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{N_L} \end{bmatrix}. \tag{25}$$

As funções custo comparam os vetores $\mathbf{y}^{(L)}$ e $\mathbf{d}$ termo a termo por meio de uma função $\mathcal{L}(y_n^{(L)}, d_n)$, ou seja,

$$J = \frac{1}{N_L} \sum_{n=1}^{N_L} \mathcal{L}(y_n^{(L)}, d_n). \tag{26}$$

Cabe observar que $\mathcal{L}(y_n^{(L)}, d_n)$ e J são escalares. Para a aplicação do algoritmo de retropropagação, as funções custo precisam ser diferenciáveis [17].

A escolha da função custo depende da finalidade da rede neural. Nesse sentido, é interessante fazer uma distinção entre os problemas de regressão e de classificação. Os problemas de regressão envolvem a estimação de determinados parâmetros, como é o caso de uma rede neural focada no cálculo de regressões lineares. Os problemas de classificação envolvem o enquadramento de diferentes sinais de entrada em diferentes categorias, como é o caso de uma rede neural que decide se uma determinada imagem representa um cachorro ou um gato [17].

Para problemas de regressão, um exemplo de função custo frequentemente escolhida é o Erro Quadrático Médio (*Mean Squared Error* – MSE) definido por [14]

$$J_{\text{MSE}} = \frac{1}{N_L}\sum_{n=1}^{N_L} e_n^2, \tag{27}$$

em que e_n é definido por

$$e_n = y_n^{(L)} - d_n. \tag{28}$$

Para problemas de classificação, um exemplo de função custo frequentemente escolhida é a entropia cruzada, uma vez que ela é especialmente rigorosa para erros de categorização. No caso de uma classificação de N_L elementos entre duas classes, $d_n = 0$ ou $d_n = 1$, a entropia cruzada como função custo é dada por [7]

$$J_{EC} = -\frac{1}{N_L}\sum_{n=1}^{N_L} d_n \log y_n^{(L)} + (1 - d_n)\log\left(1 - y_n^{(L)}\right). \tag{29}$$

Existem também outras funções custo cujas derivadas não são determinadas analiticamente, mas por diferenciação automática (*autodiff*), que é um conjunto de técnicas usadas para avaliar derivadas de funções numéricas expressas como programas de computador [10].

A.5 Algoritmo de Retropropagação

O algoritmo de retropropagação é geralmente o método escolhido para o processo de aprendizado supervisionado das redes neurais, isto é, quando tem-se um conjunto de amostras de treinamento contendo entradas e saídas esperadas. O algoritmo é dividido entre duas etapas [14]:

1. Passo progressivo:

   Nesta etapa, os pesos sinápticos se mantém inalterados e um determinado sinal é alimentado à rede. De maneira a produzir a saída estimada $\mathbf{y}^{(L)}$, esse sinal é propagado ao longo da rede, camada por camada, de acordo com o mecanismo descrito na Subseção A.1. A implementação vetorizada do passo progressivo é a preferida por ser computacionalmente mais eficiente.

2. Passo regressivo:

   Essa etapa basicamente se resume a atualizar os pesos sinápticos de acordo com uma função custo J e um passo de aprendizado η, previamente determinados. A matriz de pesos sinápticos é atualizada utilizando o método do gradiente descendente estocástico, ou seja,

$$\mathbf{W}^{(j)}(n+1) = \mathbf{W}^{(j)}(n) - \eta\frac{\partial J}{\partial \mathbf{W}^{(j)}(n)}, \tag{30}$$

em que na posição (x, y) da matriz $\frac{\partial J}{\partial \mathbf{W}^{(j)}}$, temos o elemento $\frac{\partial J}{\partial w_{xy}^{(j)}}$.

Para que seja possível calcular $\frac{\partial J}{\partial \mathbf{W}^{(j)}}$, faz-se necessária a aplicação da regra da cadeia. Por exemplo, tomando o MSE como função custo, para o vetor de pesos sinápticos ligados diretamente à saída $\mathbf{y}^{(L)}$, temos que a i-ésima linha da matriz $\frac{\partial J_{\text{MSE}}}{\partial \mathbf{W}^{(L)}}$, é dada por

$$\begin{aligned}\frac{\partial J_{\text{MSE}}}{\partial \mathbf{w}_i^{(j)}} &= \frac{\partial e_i^2}{\partial y_i^{(L)}} \frac{\partial y_i^{(L)}}{\partial z_i^{(L)}} \frac{\partial z_i^{(L)}}{\partial \mathbf{w}_i^{(L)}} \\ &= 2e_i \frac{\partial (y_i^{(L)} - d_i)}{\partial y_i^{(L)}} \frac{\partial \varphi\left(z_i^{(L)}\right)}{\partial z_i^{(L)}} \frac{\partial (\mathbf{w}_i^{(L)} \cdot \mathbf{y}^{(L-1)} + b_i^{(L)})}{\partial \mathbf{w}_i^{(L)}} \\ &= 2e_i \frac{\partial \varphi\left(z_i^{(L)}\right)}{\partial z_i^{(L)}} \mathbf{y}^{(L-1)}, \end{aligned} \tag{31}$$

ou seja, o gradiente $\frac{\partial J_{\text{MSE}}}{\partial \mathbf{w}_i^{(L)}}$ depende apenas da derivada parcial $\frac{\partial e_i^2}{\partial y_i^{(L)}}$ devida à função custo, da derivada parcial $\frac{\partial y_i^{(L)}}{\partial z_i^{(L)}}$ associada à função de ativação e do vetor $\mathbf{y}_i^{(L-1)}$, que é o vetor de saída da camada imediatamente anterior à saída da rede.

Durante o passo regressivo, o vetor de *bias* $\mathbf{b}^{(j)}$ também é atualizado de acordo com o raciocínio expresso na Equação (30). Assim, como $\frac{\partial z_i^{(L)}}{\partial b_i^{(L)}} = 1$, para o i-ésimo elemento do vetor $\frac{\partial J_{\text{MSE}}}{\partial \mathbf{b}^{(L)}}$, temos que

$$\frac{\partial J_{\text{MSE}}}{\partial b_i^{(L)}} = \frac{\partial e_i^2}{\partial y_i^{(L)}} \frac{\partial y_i^{(L)}}{\partial z_i^{(L)}} \frac{\partial z_i^{(L)}}{\partial b_i^{(L)}} = 2e_i \frac{\partial \varphi\left(z_i^{(L)}\right)}{\partial z_i^{(L)}}. \tag{32}$$

Para o k-ésimo neurônio da camada L de saída de uma rede, é possível generalizar o algoritmo de retropropagação a partir do gradiente local $\delta_k^{(L)}$, definido como

$$\delta_k^{(L)} = \frac{\partial \mathcal{L}(y_k^{(L)}, d_k)}{\partial y_k^{(L)}} \frac{\partial \varphi\left(z_k^{(L)}\right)}{\partial z_k^{(L)}}. \tag{33}$$

De forma similar, para uma camada j qualquer da rede, tem-se que

$$\delta_i^{(j)} = \delta_k^{(j+1)} \frac{\partial \varphi\left(z_i^{(j)}\right)}{\partial z_i^{(j)}}, \tag{34}$$

em que o neurônio i da camada j está ligado ao neurônio k da camada $(j+1)$. Sendo assim, o gradiente $\frac{\partial J}{\partial w_{ki}^{(j)}}$ é computado como

$$\frac{\partial J}{\partial w_{ki}^{(j)}} = \delta_k^{(j)} y_i^{(j-1)}. \tag{35}$$

Analogamente, como na equação (32), o gradiente $\frac{\partial J}{\partial b_i^{(j)}}$ é dado por

$$\frac{\partial J}{\partial b_i^{(j)}} = \delta_k^{(j)}. \tag{36}$$

Por fim, vale ressaltar o caráter de retropropagação do algoritmo, ou seja, para se computar os gradientes de camadas menos profundas da rede, se faz necessário o cálculo dos gradientes associados às camadas mais profundas, como é visto na Equação (34), percorrendo-se o sentido inverso do passo progressivo. Uma dedução mais detalhada do algoritmo de retropropagação encontra-se em [14].

A.6 Problema das Meia-luas

O Problema das Meia-luas é um exemplo clássico de aplicação prática dos conceitos desenvolvidos até aqui [14]. O problema consiste em classificar um vetor de entrada bidimensional $[x_1 \;\; x_2]^T$ entre duas classes C_1 e C_2. As classes apresentam um formato de meia-lua no plano e suas posições podem ser modificadas a partir da manipulação de determinados parâmetros (r_1, r_2 e r_3), de modo que as classes podem ser tanto linearmente quanto não linearmente separáveis. Na Figura 34, são expostos os parâmetros que descrevem as posições das classes C_1 e C_2. Note que para $r_1 \leq 0$, as classes são não linearmente separáveis [14].

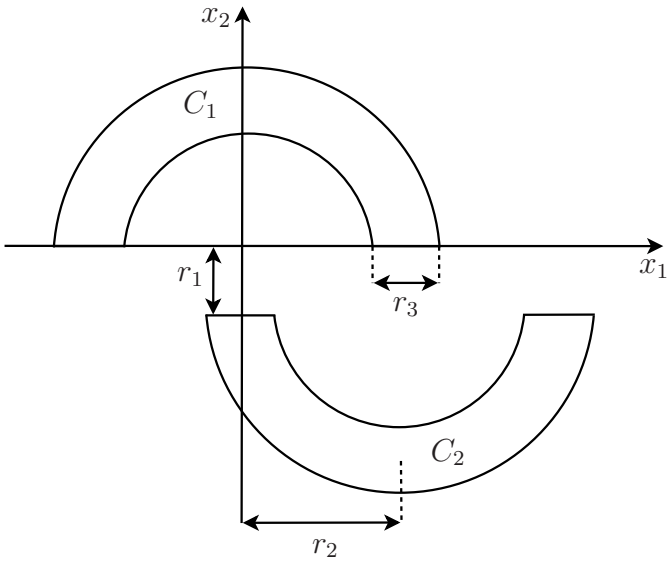


Figura 34: Parâmetros das meia-luas.

Para a resolução do problema, considerando classes não-linearmente separáveis ($r_1 = -5$, $r_2 = 10$ e $r_3 = 5$), foram aplicadas duas redes MLP distintas. A primeira não possuía nenhuma camada oculta de neurônios, configuração similar ao próprio *perceptron* de Rosenblatt e, consequentemente, capaz apenas de resolver problemas com classes linearmente separáveis. Sua entrada era composta por dois elementos (entrada bidimensional) e a camada de saída era composta por um neurônio (classificação binária), ou seja, a rede possuía uma configuração 1. A função de ativação escolhida foi a sigmoidal e a função custo aplicada foi a entropia cruzada.

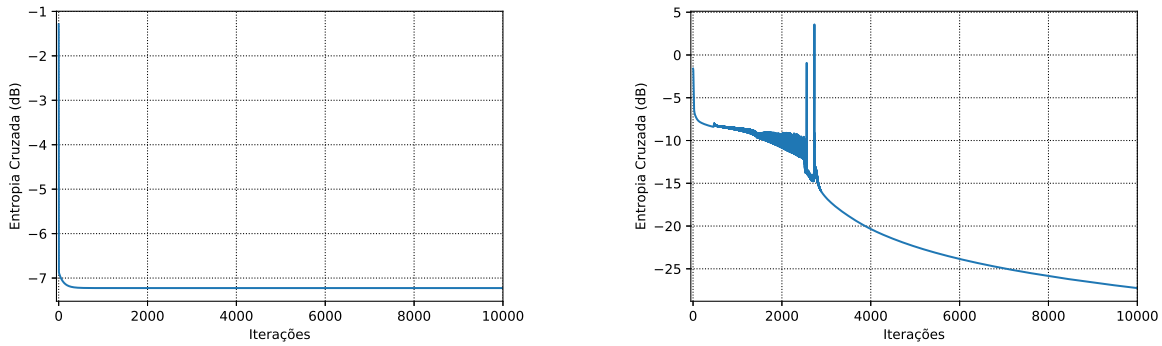
A segunda rede neural usada possuía uma entrada com dois elementos, uma camada oculta de 10 neurônios e uma camada de saída com um neurônio, ou seja, a rede possuía uma configuração 10-1. As funções de ativação escolhidas foram a ReLU para a camada oculta e a sigmoidal para a camada de saída, a função custo aplicada foi a entropia cruzada. Na Tabela 4, são sumarizadas as configurações das redes MLP utilizadas.

	Rede 1	**Rede 10-1**
Entrada	2 elementos	
Camada oculta	Nenhum neurônio	10 neurônios
Camada de saída	1 neurônio	
Função de ativação	Sigmoidal	ReLU (oculta); Sigmoidal (saída)
Função custo	Entropia cruzada	
Passo de aprendizado	0,15	

Tabela 4: Sumário das redes MLP utilizadas.

As implementações de ambas as redes foram feitas em *Python*, utilizando-se as bibliotecas *Numpy*, para a manipulação de matrizes e tensores, e *Matplotlib*, para a representação gráfica dos resultados.

Na Figura 35, mostram-se os gráficos das funções custo de ambas as redes ao longo de 10000 iterações do algoritmo de retropropagação. Na Figura 36, mostram-se as fronteiras de

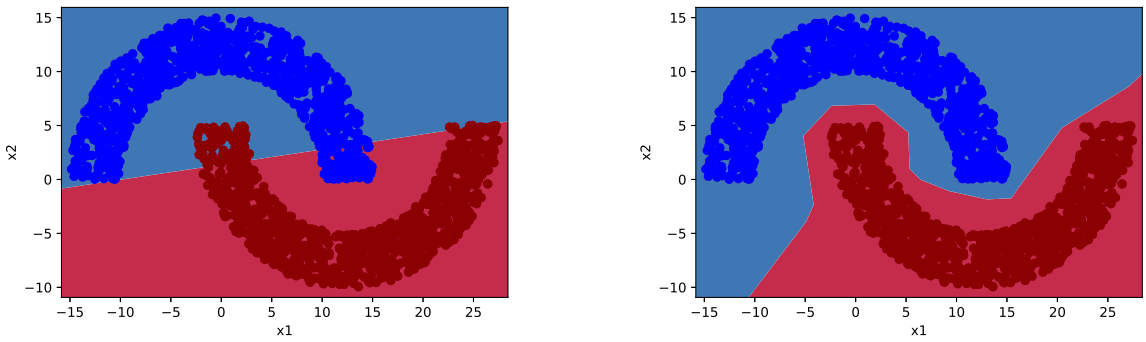


(a) Rede de configuração 1

(b) Rede de configuração 10-1

Figura 35: Gráficos das funções custo das redes neurais ao longo das iterações.

decisão traçadas por cada rede para a resolução do problema. Sobre as fronteiras, também foram colocadas as amostras de treinamento usadas no processo de aprendizado das redes.



(a) Rede de configuração 1

(b) Rede de configuração 10-1

Figura 36: Fronteiras de decisão traçadas por cada rede.

É interessante notar como a função de entropia cruzada da rede de configuração 2-1 não consegue diminuir mais a partir de certo ponto (-7 dB) e como a fronteira traçada por essa rede não é capaz de solucionar um problema com classes não linearmente separáveis.

B Redes Neurais Convolucionais

Redes Neurais Convolucionais (*Convolutional Neural Networks* – CNNs) são um tipo específico de redes neurais amplamente usadas para o processamento de dados organizados em matrizes, como é o caso do processamento de imagens, que são matrizes ou tensores de pixels [7].

As CNNs empregam em pelo menos uma de suas camadas uma operação matemática chamada de convolução, um tipo de operação linear, ao invés da simples multiplicação de matrizes. A maior vantagem de se usar a convolução é a menor quantidade de parâmetros a serem atualizados em comparação às camadas totalmente conectadas das redes MLP.

B.1 Camadas Convolucionais

A operação de convolução entre duas matrizes $\mathbf{I}$ e $\mathbf{K}$, denominadas respectivamente entrada e filtro, é dada por [7]

$$\mathbf{S}(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{I}(m,n)\mathbf{K}(i-m, j-n), \tag{37}$$

em que $\mathbf{S}(i,j)$ representa o elemento (i,j) do resultado dessa operação e $\mathbf{K}(i-m, j-n)$ representa o filtro reverso em ambas as direções.

Entretanto, no contexto do aprendizado de máquina, geralmente o filtro não é reverso, usando-se, na realidade, a operação de correlação cruzada entre as matrizes $\mathbf{I}$ e $\mathbf{K}$, dada por [7]

$$\mathbf{S}(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{I}(i+m, j+n)\mathbf{K}(m,j). \tag{38}$$

Por convenção, essa operação também é chamada em aprendizado de máquina de convolução, ou convolução sem reversão de filtro [7]. Na Figura 37, é elucidada de forma gráfica a operação de convolução sem reversão de filtro.

Vale notar que a convolução de uma matriz $\mathbf{I}$ de dimensão (i_1, i_2) com uma matriz $\mathbf{K}$ de dimensão (f_1, f_2) gera uma matriz $\mathbf{S}$ de dimensão $(i_1 - f_1 + 1, i_2 - f_2 + 1)$. Dessa forma, é comum que a matriz $\mathbf{I}$ seja aumentada por bordas adicionais preenchidas por elementos nulos, de modo que $\mathbf{S}$ fique com a mesma dimensão original de $\mathbf{I}$. Essa prática é conhecida por *zero padding* e a quantidade de colunas ou linhas preenchidas por elementos nulos adicionadas à matriz é dada por p. Outra prática aplicada em CNNs é o *stride*, isto é, ao invés de se deslocar linha por linha e coluna por coluna da entrada, é comum fazer com que o filtro se desloque mais de uma linha ou coluna por vez. A quantidade de linhas ou colunas percorridas de uma só vez pelo filtro é dada por s. Ao considerar *zero padding* (p) e *stride* (s), a dimensão da matriz $\mathbf{S}$ é dada por $(\lfloor \frac{i_1+2p-f_1}{s} \rfloor + 1, \lfloor \frac{i_2+2p-f_2}{s} \rfloor + 1)$ [17].

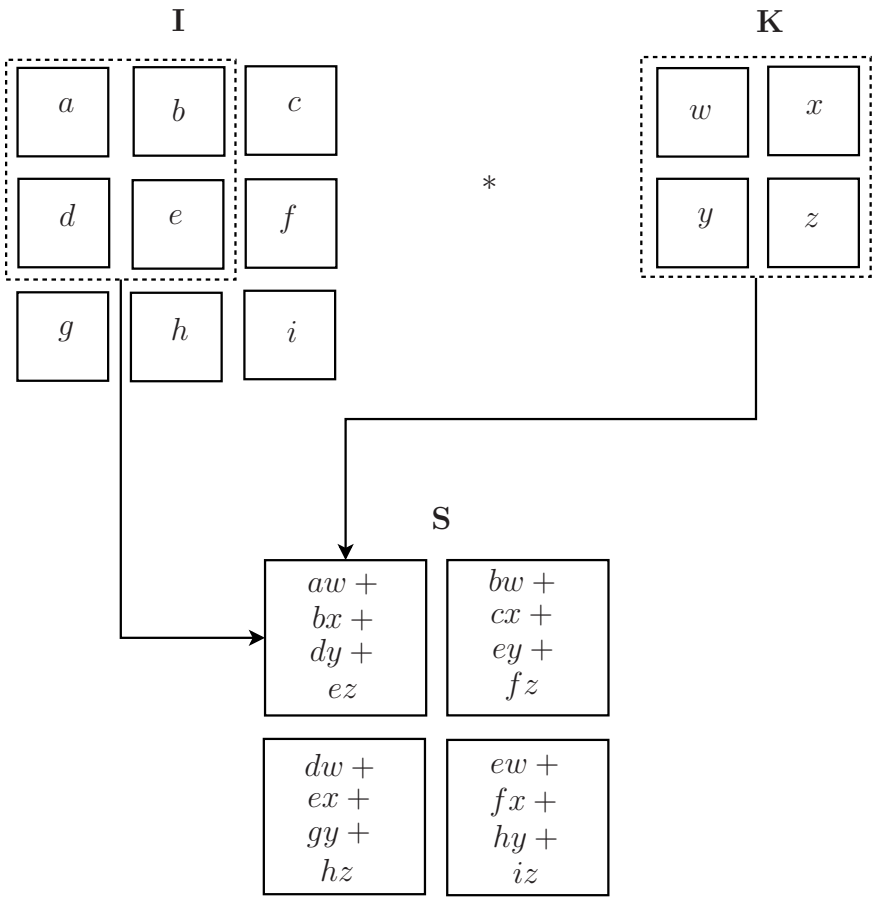


Figura 37: Exemplo de convolução sem reversão de filtro entre uma matriz **I** com dimensão (3, 3) e uma matriz **K** com dimensão (2, 2). Adaptado de [7].

Uma camada convolucional apresenta vários filtros com a mesma dimensão, o mesmo *zero padding* e o mesmo *stride*, cada filtro produz uma matriz de saída e todas as matrizes de saída formam um tensor tridimensional [17]. Por exemplo, uma camada com 10 filtros de dimensão (5, 5), com $p = 0$ e $s = 1$, para uma matriz de entrada com dimensão (64, 64), gerará um tensor de saída com dimensão (60, 60, 10).

Nesse tipo de camada, os parâmetros a serem atualizados são os elementos dos filtros, menos numerosos do que os parâmetros das camadas totalmente conectadas típicas das redes MLP. Sendo assim, com menos parâmetros a serem aprendidos, o processo de treinamento de uma CNN se torna mais simples e objetivo. O treinamento também é feito através do algoritmo de *backpropagation*.

C Ajustes de Hiperparâmetros

No contexto de redes neurais artificiais, hiperparâmetros são todos os parâmetros de uma rede que permanecem inalterados ao longo do treinamento. O passo de aprendizado η pode ser considerado um hiperparâmetro, já os pesos sinápticos não, uma vez que eles se atualizam a cada iteração do algoritmo de retropropagação.

O ajuste prévio desses hiperparâmetros possui influência sobre o desempenho de uma rede neural. No restante desta Subseção, serão introduzidos outros hiperparâmetros além do passo de aprendizado e também serão discutidos os efeitos que esses hiperparâmetros exercem sobre o processo de treinamento de redes neurais.

C.1 Inicializações de Parâmetros

Como visto em A.5, o cálculo do gradiente local de uma camada j qualquer de uma rede neural depende dos gradientes locais das camadas posteriores, isto é, pela Equação (34), o gradiente local $\delta_k^{(j)}$ carrega consigo a multiplicação de todos os gradientes locais das camadas mais profundas da rede [17].

Se os gradiente locais forem menores do que um, para redes neurais profundas, temos que as atualizações dos pesos sinápticos das camadas mais rasas da rede acabam assumindo valores muito pequenos, tornando o processo de aprendizado da rede lento e ineficiente. Analogamente, para gradientes locais sempre maiores que um, as atualizações dos pesos sinápticos das camadas menos profundas acabam assumindo valores muito elevados, levando o processo de treinamento à instabilidade, i.e, os gradientes vão para infinito. Esse obstáculo técnico em relação ao aprendizado de redes neurais é o chamado problema do desvanecimento ou explosão de gradientes [7].

O objetivo das técnicas de inicialização de parâmetros é mitigar os efeitos do desvanecimento ou explosão de gradientes. Para tal, os pesos sinápticos precisam ser inicializados dentro de um intervalo específico. Uma boa referência a ser seguida é garantir que a variância de $z_k^{(j)}$, para cada camada, seja inicialmente unitária [19].

Considerando que o vetor de entrada da rede esteja normalizado (média nula e variância unitária) e que $\mathbf{b}^{(j)}$ seja inicializado como um vetor nulo, a variância de $\mathbf{z}^{(j)}$ é dada por

$$\sigma^2(\mathbf{z}^{(j)}) = \sigma^2(\mathbf{W}^{(j)}\mathbf{y}^{(j-1)} + \mathbf{b}^{(j)}) = N_{j-1} \cdot \sigma^2(\mathbf{W}^{(j)}), \tag{39}$$

em que N_{j-1} é a quantidade de neurônios na camada $j-1$. Sendo assim, para que $\sigma^2(\mathbf{z}^{(j)}) = 1$,

$$\sigma^2(\mathbf{W}^{(j)}) = \frac{1}{N_{j-1}}. \tag{40}$$

Isto é, a matriz $\mathbf{W}^{(j)}$ pode ser inicializada a partir de uma distribuição normal tal que $\mathbf{W}^{(j)} \sim N(0, \frac{1}{N_{j-1}})$, o que é conhecido como inicialização de Xavier [19]. Outras inicializações se baseiam no mesmo princípio expresso na Equação (40), a inicialização de He [20], por exemplo, faz com que $\mathbf{z}^{(j)}$ possua uma variância igual a 2, ou seja, inicializa $\mathbf{W}^{(j)}$ a partir de $\mathbf{W}^{(j)} \sim N(0, \frac{2}{N_{j-1}})$. A inicialização de Glorot [19], por sua vez, inicializa a matriz $\mathbf{W}^{(j)}$ a partir da distribuição $\mathbf{W}^{(j)} \sim N(0, \frac{2}{N_{j-1}+N_j})$, em que são consideradas tanto a quantidade de neurônios na camada $j-1$, quanto a quantidade de neurônios na camada j. A inicialização de parâmetros usada nas redes MLP apresentadas em A.6 foi a inicialização de Xavier.

Em todas as inicializações citadas acima, os vetores $\mathbf{b}^{(j)}$ são inicializados como vetores nulos. A principal motivação em garantir com que $\mathbf{z}^{(j)}$ possua média nula e uma variância predefinida vem do fato de que, para as funções de ativação expostas em A.3, a derivada parcial $\frac{\partial\varphi(\mathbf{z}^{(j)})}{\partial\mathbf{z}^{(j)}}$ alcança seus valores máximos próximo da origem, agilizando o processo de aprendizado da rede ao evitar que os gradientes locais se tornem muito pequenos já nas primeiras iterações [19].

Por fim, vale ressaltar que os hiperparâmetros associado às inicializações discutidas são justamente os valores da média e variância associados de $\mathbf{z}^{(j)}$.

C.2 Regularização *Dropout*

Outro obstáculo técnico ao treinamento de redes neurais é o sobreajuste (*overfitting*), que ocorre quando há uma diferença significativa entre o desempenho da rede sobre o seu conjunto de treinamento e sobre um outro conjunto de amostras distinto, denominado conjunto de teste. Em outras palavras, uma rede apresenta sobreajuste ao se adaptar muito justamente ao conjunto de treinamento, porém sem apresentar capacidade de generalização satisfatória para outras entradas.

As regularizações, portanto, buscam amenizar os efeitos do sobreajuste, sendo o *dropout* um tipo de regularização. A técnica de *dropout* se resume a inativar aleatoriamente, em cada iteração do algoritmo de retropropagação, diferentes neurônios de cada camada oculta da rede [12]. Cada neurônio é desativado com probabilidade p, sendo p o hiperparâmetro associado a essa regularização. Na Figura 38, exemplifica-se a aplicação do *dropout* com $p = 0{,}5$.

Ao inativar neurônios de todas as camadas da rede, será mais difícil para a rede se adaptar muito justamente às amostras de treinamento, evitando o sobreajuste e garantindo um maior

poder de generalização.

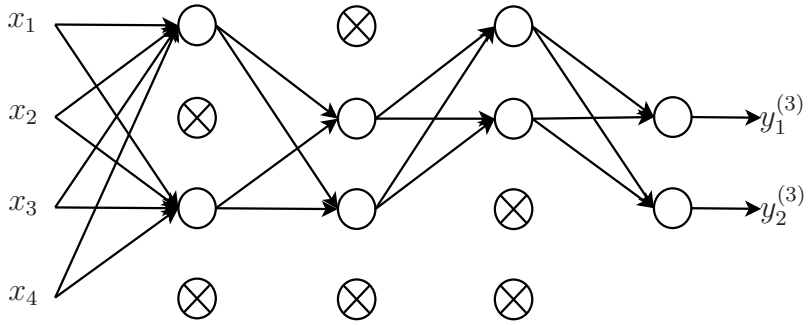


Figura 38: Exemplo de aplicação do *dropout* em uma rede MLP com $p = 0{,}5$.

C.3 Otimizador Adam

No contexto de redes neurais, otimizadores são algoritmos, com seus hiperparâmetros internos, que possuem a finalidade de minimizar determinada função custo. O algoritmo de otimização usado para a resolução do Problema das Meia-luas, explicado detalhadamente em A.5, é denominado gradiente descendente estocástico (*Stochastic Gradient Descent* – SGD). Aqui, é discutido outro otimizador: o otimizador Adam.

O otimizador Adam se resume a atualizar os pesos sinápticos e *bias* de uma rede neural a partir dos gradientes $\frac{\partial J}{\partial \mathbf{W}}$ e $\frac{\partial J}{\partial \mathbf{b}}$ calculados tanto na própria iteração do algoritmo quanto em iterações passadas, de forma a tornar mais estável o processo de aprendizado da rede, evitando variações excessivas em direções que não são a do mínimo da função custo [17].

Para esse otimizador, são introduzidos a matriz $\mathbf{V}_w^{(j)}$ e o vetor $\mathbf{v}_b^{(j)}$, definidos recursivamente na iteração $t+1$ por [17]

$$\begin{aligned} \mathbf{V}_w^{(j)}(t+1) &= \beta_1 \mathbf{V}_w^{(j)}(t) + (1-\beta_1)\frac{\partial J}{\partial \mathbf{W}^{(j)}}(t+1) \text{ e} \\ \mathbf{v}_b^{(j)}(t+1) &= \beta_1 \mathbf{v}_b^{(j)}(t) + (1-\beta_1)\frac{\partial J}{\partial \mathbf{b}^{(j)}}(t+1). \end{aligned} \tag{41}$$

Tanto $\mathbf{V}_w^{(j)}$ quanto $\mathbf{v}_b^{(j)}$ carregam informações sobre gradientes de iterações passadas. Em uma analogia mecânica, geralmente associa-se $\mathbf{V}_w^{(j)}$ e $\mathbf{v}_b^{(j)}$ ao conceito de velocidade e $\frac{\partial J}{\partial \mathbf{W}^{(j)}}$ e $\frac{\partial J}{\partial \mathbf{b}^{(j)}}$ ao conceito de aceleração.

Adicionalmente, são introduzidos a matriz $\mathbf{S}_w^{(j)}$ e o vetor $\mathbf{s}_b^{(j)}$, definidos recursivamente na iteração $t+1$ por [17]

$$\begin{aligned} \mathbf{S}_w^{(j)}(t+1) &= \beta_2 \mathbf{S}_w^{(j)}(t) + (1-\beta_2)\left[\frac{\partial J}{\partial \mathbf{W}^{(j)}}(t+1)\right]^2 \text{ e} \\ \mathbf{s}_b^{(j)}(t+1) &= \beta_2 \mathbf{s}_b^{(j)}(t) + (1-\beta_2)\left[\frac{\partial J}{\partial \mathbf{b}^{(j)}}(t+1)\right]^2, \end{aligned} \tag{42}$$

também considerando gradientes calculados em iterações prévias do algoritmo. Vale ressaltar que $\mathbf{V}_w^{(j)}$, $\mathbf{v}_b^{(j)}$, $\mathbf{S}_w^{(j)}$ e $\mathbf{s}_b^{(j)}$ são inicializadas como matrizes ou vetores nulos, apresentando valores próximos a zero nas primeiras iterações do otimizador. A operação de exponenciação representada em (42) se dá de elemento em elemento para as matrizes e vetores considerados.

Sendo assim, também são introduzidos as matrizes de correção $\mathbf{V}_w^{corrigido}$ e $\mathbf{S}_w^{corrigido}$ e os vetores de correção $\mathbf{v}_b^{corrigido}$ e $\mathbf{s}_b^{corrigido}$, definidos na iteração t por

$$\begin{aligned}
\mathbf{V}_w^{corrigido}(t) &= \frac{\mathbf{V}_w^{(j)}}{1-\beta_1^t}, \\
\mathbf{S}_w^{corrigido}(t) &= \frac{\mathbf{S}_w^{(j)}}{1-\beta_2^t}, \\
\mathbf{v}_b^{corrigido}(t) &= \frac{\mathbf{v}_b^{(j)}}{1-\beta_1^t} \text{ e} \\
\mathbf{s}_b^{corrigido}(t) &= \frac{\mathbf{s}_b^{(j)}}{1-\beta_2^t}.
\end{aligned} \tag{43}$$

Essas correções buscam amenizar as distorções apresentadas pelas matrizes e vetores durante as primeiras iterações do algoritmo.

Enfim, a atualização de parâmetros da rede é dada por

$$\begin{aligned}
\mathbf{W}^{(j)}(t+1) &= \mathbf{W}^{(j)}(t) + \eta \frac{\mathbf{V}_w^{corrigido}(t)}{\sqrt{\mathbf{S}_w^{corrigido}(t)} + \varepsilon} \text{ e} \\
\mathbf{b}^{(j)}(t+1) &= \mathbf{b}^{(j)}(t) + \eta \frac{\mathbf{v}_b^{corrigido}(t)}{\sqrt{\mathbf{s}_b^{corrigido}(t)} + \varepsilon}.
\end{aligned} \tag{44}$$

Geralmente, atribui-se à variável ε um valor pequeno, da ordem de 10^{-8}, para evitar divisões por zero no algoritmo. Os hiperparâmetros desse algoritmo são η, β_1 e β_2, em que geralmente $0 \ll \beta_1, \beta_2 < 1$ [17].

C.4 *Mini Batches*

O uso de *mini batches* no processo de aprendizado de redes neurais consiste em dividir aleatoriamente o conjunto de treinamento da rede em blocos de tamanho previamente definido, embaralhando-se as amostras do conjunto [7]. A atualização dos pesos sinápticos e dos *bias* ocorre apenas depois que são calculados $\frac{\partial J}{\partial \mathbf{W}^{(j)}}$ e $\frac{\partial J}{\partial \mathbf{b}^{(j)}}$ de todos os elementos de um *mini batch*. Dessa forma, a atualização dos parâmetros da rede estaria associada à somatória de todos $\frac{\partial J}{\partial \mathbf{W}^{(j)}}$ e $\frac{\partial J}{\partial \mathbf{b}^{(j)}}$ de um *mini batch*.

Considera-se passada uma época quando todos os *mini batches* são percorridos. Após cada época do algoritmo de otimização, a divisão do conjunto de treinamento entre *mini batches*

é refeita de maneira aleatória, embaralhando-se novamente o conjunto de treinamento. O tamanho de cada *mini batch* é um hiperparâmetro e não muda no decorrer das épocas.

Quando se considera que cada *mini batch* é formado apenas por uma amostra do conjunto de treinamento, diz-se que o método de atualização de parâmetros é estocástico. O uso do método estocástico para atualização de parâmetros é pouco eficiente, pois os parâmetros muitas vezes são atualizados em direções distintas do mínimo da função custo, levando mais épocas para se aproximar da convergência. O método estocástico também anula as vantagens computacionais de uma implementação vetorizada de rede neural, uma vez que as atualizações são realizadas sobre cada amostra de treinamento.

Quando um *mini batch* possui todos os elementos do conjunto de treinamento, como no caso das redes MLP testadas em A.6, nomeia-se o método de atualização de parâmetros apenas como *batch*. Com o método *batch*, os parâmetros são sempre atualizados na direção do mínimo da função custo, porém, como é necessário esperar até que todo o conjunto de treinamento seja percorrido para se realizar a atualização dos parâmetros, geralmente é um método mais demorado se comparado com outros métodos que usam *mini batches* menores [7].

Referências

[1] Deepa Kundur and Dimitrios Hatzinakos. Blind image deconvolution. *IEEE Signal Processing Magazine*, 13(3):43–64, 1996.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[3] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[4] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

[5] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, Geoffrey E Hinton, et al. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[6] Ajit V Rao, David Miller, Kenneth Rose, and Allen Gersho. Mixture of experts regression modeling by deterministic annealing. *IEEE Transactions on Signal Processing*, 45(11):2811–2820, 1997.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[8] L. A. Rodrigues, R. Candido, and M. T. M. Silva. Restauração de imagens com redes neuronais. In *Anais do Simpósio Brasileiro de telecomunicações e Processamento de Sinais (SBrT)*, pages 856–857, Campina Grande, PB, 2018. Sociedade Brasileira de Telecomunicações.

[9] Zhang X. Ren S. He, K. and J. Sun. Deep residual learning for image recognition. *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Pearlmutter B. A. Radul A. A. Baydin, A. G. and J. M. Siskind. Automatic differentiation in machine learning: a survey. *J. Machine Learning Research*, 18(153):1–43, 2018.

[11] Ravid Cohen. *Mixture of convolutional neural networks for image classification*. PhD thesis, Hebrew University of Jerusalem, 2018.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[13] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiri Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[14] Simon Haykin. *Neural Networks and Learning Machines, 3/E*. Pearson Education, 2010.

[15] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[16] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychologial Review*, 65(6):386, 1958.

[17] Andrew Ng. Deeplearning.ai. https://pt.coursera.org/specializations/deep-learning, 2017. Acessado em 04/07/19.

[18] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

Anexo 1 - Artigo submetido ao SBrT 2020

XXXVIII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES E PROCESSAMENTO DE SINAIS - SBrT 2020, 22–25 DE NOVEMBRO DE 2020, FLORIANÓPOLIS, SC

Restauração de imagens coloridas usando uma rede neural convolucional residual

Guilherme A. Lizarzaburu, Renato Candido e Magno T. M. Silva

***Resumo*— A restauração de imagens tem por objetivo atenuar distorções causadas no processo de aquisição. Neste artigo, utiliza-se uma rede neural convolucional residual para restauração de imagens coloridas, degradadas por uma função de espalhamento de ponto gaussiana. Como função custo, foram considerados o erro quadrático médio e uma função do índice de similaridade estrutural (*structural similarity* – SSIM). Por meio de simulações, verifica-se que os melhores resultados de restauração foram obtidos com a função custo baseada no SSIM.**

***Palavras-Chave*— Rede neural convolucional residual, restauração de imagens, função de espalhamento de ponto gaussiana, índice de similaridade estrutural.**

***Abstract*— Image restoration aims to attenuate distortions caused in the acquisition process. In this paper, we use a residual convolutional neural network to restore colored images which were degraded by a Gaussian point spread function. As cost function, we consider the mean square error, and a function based on the structural similarity (SSIM) index. By means of simulations, we verify that the best restoration results were achieved with the function based on SSIM.**

***Keywords*— Convolutional neural network with residual skip connections, image restoration, gaussian point spread function, structural similarity index.**

I. Introdução e formulação do problema

No processo de aquisição, as imagens digitais podem sofrer degradações, que em geral são modeladas por uma função de espalhamento de ponto (*point spread function* - PSF) [5]. Um dos modelos de distorção mais usados na literatura é a PSF gaussiana de dimensão $(2N_1+1)\times(2N_2+1)$ em que

$$\mathbf{H}(n_1,n_2) = (2\pi\sigma^2)^{-1}\exp\left(-(n_1^2+n_2^2)/(2\sigma^2)\right),$$

sendo σ o desvio padrão, $n_1 = -N_1,\cdots,-1,0,1,\cdots,N_1$ e $n_2 = -N_2,\cdots,-1,0,1,\cdots,N_2$. A imagem degradada é obtida a partir da convolução em duas dimensões entre a imagem original e a PSF. No caso de imagens coloridas, compostas por canais de cores vermelho, verde e azul, cada canal da imagem degradada é fruto da convolução entre a PSF e o respectivo canal da imagem original.

Para melhorar a qualidade de uma imagem degradada, o efeito da PSF deve ser atenuado, o que pode ser feito com técnicas de aprendizado de máquina [2], [3], [6], [8]. Neste contexto, a rede neural convolucional (*convolutional neural network* - CNN) tem ganhado destaque na literatura [3]. No entanto, a maior parte dos trabalhos que usam a CNN para restauração de imagens, considera o erro quadrático médio (*mean square error* – MSE) como função custo no treinamento. O MSE não é uma medida adequada quando se trata de imagens já que ele mede erros absolutos, não levando em conta características do sistema visual humano [7], [8]. No lugar do MSE, é mais adequado usar o índice de similaridade estrutural (*structural similarity* – SSIM), proposto em [7]. O SSIM mede a similaridade entre duas imagens a partir da combinação de três medidas distintas referentes à luminância, ao contraste e à estrutura das imagens [7]. Esse índice assume valores no intervalo $[-1,\ 1]$, sendo igual a um quando as duas imagens são iguais.

Neste trabalho, considera-se uma CNN residual [3], [4] para restauração de imagens coloridas degradadas por uma PSF gaussiana. Como função custo, considera-se uma função do índice SSIM. Os resultados obtidos são então comparados com a mesma rede treinada para minimizar o MSE.

Guilherme A. Lizarzaburu, Renato Candido, Magno T. M. Silva, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, emails: guilherme.lizarzaburu@usp.br; renatocan@lps.usp.br; magno.silva@usp.br. Este trabalho foi financiado pela FAPESP (2019/08636-8, 2017/20378-9), CNPq (304715/2017-4) e CAPES (código de financiamento 001).

II. Arquitetura da CNN proposta

A CNN residual proposta é esquematizada na Figura 1. Ela é composta por 10 camadas convolucionais, tendo a tangente hiperbólica como função de ativação. As dimensões e quantidades dos filtros por camada estão indicadas na figura. Cabe lembrar que cada camada convolucional realiza a operação de convolução entre seus filtros e o tensor de entrada, que por sua vez, passa pelo processo de *zero padding* para adequação das dimensões [3]. A quantidade de filtros por camada na CNN proposta aumenta gradativamente de 3 para 32 da primeira à quinta camada, respectivamente. Isso faz com que o tensor de saída da quinta camada tenha dimensão $256\times256\times32$, o que possibilita extrair diferentes características da imagem degradada, facilitando o processo de restauração. Da quinta à décima camada, a quantidade de filtros por camada diminui de 32 para 3, respectivamente. Assim, o tensor de saída da última camada é a imagem restaurada com dimensão $256\times256\times3$, exatamente igual à dimensão da imagem degradada. Além disso, considerou-se uma rede do tipo residual, em que existem atalhos entre camadas não consecutivas a fim de otimizar o processo de aprendizado [4]. No caso da CNN proposta, como esquematizado na Figura 1, a entrada da décima camada é dada pela soma das saídas da nona com a da primeira camada, a entrada da nona camada é dada pela soma das saídas da oitava com a da segunda camada e assim sucessivamente.

Os coeficientes dos filtros de cada camada são adaptados no treinamento utilizando o algoritmo de retropropagação (*backpropagation*) de modo a minimizar uma função custo [3]. Além do MSE, a rede proposta foi treinada para minimizar a função custo $J = 1 - \text{SSIM}(\mathbf{Y},\mathbf{D})$, em que $\mathbf{Y}$ é a imagem restaurada e $\mathbf{D}$ é a imagem original [8]. Para isso, utilizou-se

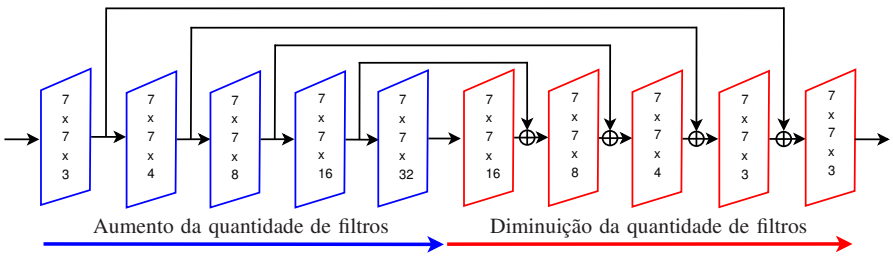


Fig. 1: Esquema da CNN residual utilizada.

a diferenciação automática (*autodiff*), que é um conjunto de técnicas usadas para avaliar derivadas de funções numéricas expressas como programas de computador [1].

III. Resultados de simulação

No treinamento da rede, foram utilizadas 29 imagens coloridas degradadas pela PSF gaussiana com $N_1 = N_2 = 3$ e três valores diferentes para σ: 1, 3 e 5. Os pixels de cada canal foram truncados para valores inteiros no intervalo [0, 255].

Depois de treinada por 2×10^4 épocas, a rede foi testada com duas imagens não pertencentes ao conjunto de treinamento e degradadas por PSFs gaussianas com $N_1 = N_2 = 3$ e dois valores diferentes de σ: $\sigma = 5$ considerado nas degradações do conjunto de treinamento e $\sigma = 4$ não utilizado no treinamento. Na Figura 2, são mostrados os resultados de restauração considerando duas versões da CNN residual proposta: uma treinada para minimizar o MSE e outra para minimizar a função do SSIM. É possível notar que a rede que utiliza o SSIM na função custo proporciona uma restauração melhor em termos do SSIM medido. Esse resultado pode ser observado visualmente, com as cores mais precisas no rosto mostrado na Fig. 2-(d) e a menor quantidade de manchas em torno da palmeira mostrada na Fig. 2-(h).

IV. Conclusões

A CNN residual proposta foi capaz de atenuar as distorções das duas imagens degradadas do teste, independente da função custo considerada no treinamento. No entanto, cabe ressaltar o desempenho superior da rede treinada com a função custo baseada no SSIM quando comparada com a rede treinada com o MSE. Em um trabalho futuro, pretende-se comparar os resultados obtidos com os de uma rede gerativa adversária.

Referências

[1] A. G. Baydin *et al.*, "Automatic differentiation in machine learning: a survey," *J. Machine Learning Research*, v. 18, p. 1–43, 2018.

[2] S. A. Bigdeli and M. Zwicker, "Image restoration using autoencoding priors." in *13th Int. Joint Conf. Comp. Vision, Imaging and Comp. Graphics Theory and Appl. (VISIGRAPP)*, p. 33–44, 2018.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

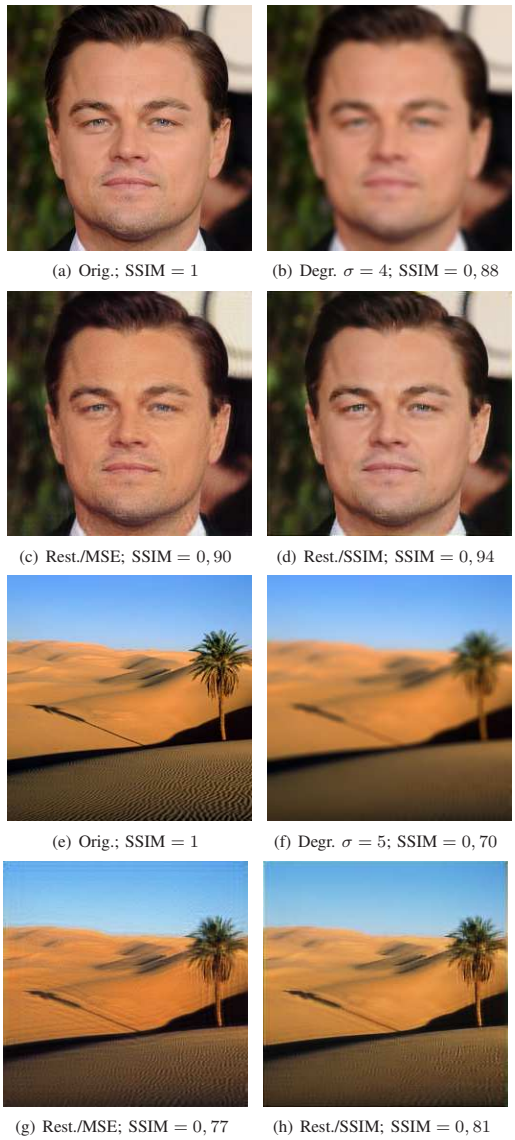
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition." in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] D. Kundur and D. Hatzinakos, "Blind image deconvolution," *IEEE Signal Process. Mag.*, v. 13, p. 43–46, 1996.

[6] X. Mao, S. Chunhua, and Y. Yu-Bin, "Image restoration using very deep convolutional enc.-dec. networks w/ symmetric skip connections," *Adv. Neural Inf. Process. Syst.*, p. 2802–2810, 2016.

[7] Z. Wang, A. C. Bovik,H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, v. 13, p. 600–612, 2004.

[8] H. Zhao *et al.*, "Loss functions for image restoration with neural networks," *IEEE Trans. Computational Imaging*, v. 3, p. 47–57, 2016.



(a) Orig.; SSIM $= 1$ (b) Degr. $\sigma = 4$; SSIM $= 0,88$

(c) Rest./MSE; SSIM $= 0,90$ (d) Rest./SSIM; SSIM $= 0,94$

(e) Orig.; SSIM $= 1$ (f) Degr. $\sigma = 5$; SSIM $= 0,70$

(g) Rest./MSE; SSIM $= 0,77$ (h) Rest./SSIM; SSIM $= 0,81$

Fig. 2: Exemplos de duas imagens restauradas pela CNN residual proposta usando MSE ou função do SSIM como função custo; SSIM calculado em relação à imagem original.