Relatório Final - Iniciação Científica

Recontrução de Imagens em Super-Resolução com Redes Neurais

Bolsista: Eduardo Paes Leme Jaqueira
Orientador: Magno Teófilo Madeira da Silva
Coorientador: Renato Candido

Processo Nº: 121036/2022-7

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

Setembro de 2023

Sumário

Resumo

Este relatório final tem como principal propósito apresentar os resultados obtidos para a super-resolução de imagens com três tipos de redes neurais: rede perceptron multicamada, rede neural convolucional e rede adversária gerativa. Primeiramente são apresentados as motivações e os objetivos dessa pesquisa. Posteriormente são apresentados os resultados alcançados com as três redes consideradas. Em todas as abordagens, um padrão de apresentação é utilizado. Em um primeiro momento, é apresentado o banco de dados utilizado para o treinamento e teste da rede, assim como os procedimentos adotados para formá-lo. Em seguida, a arquitetura da rede é apresentada, descrevendo o processo pelo qual as imagens são submetidas no treinamento. Por fim, os resultados dos testes realizados são apresentados. Cada rede tem seus resultados comparados com a interpolação bicúbica. Ao final, é apresentada uma conclusão sobre o projeto, comparando os resultados obtidos e com os objetivos propostos para essa pesquisa. O relatório termina com um apêndice contendo os fundamentos teóricos de redes neurais.

1 Introdução

Imagens em alta resolução são necessárias em diversas áreas como medicina, astronomia, segurança, monitoramento sísmico, entre outras. Aumentar a resolução de imagens se faz necessário porque as câmeras nem sempre possuem a resolução desejada [1]. Além disso, há situações em que algum fator externo pode reduzir a qualidade esperada de uma imagem, como acontece, por exemplo, quando o paciente se mexe durante a aquisição de uma imagem médica [2, 3].

As técnicas de super-resolução são utilizadas para recuperar a qualidade da imagem após sofrerem um processo de degradação desconhecido. Como exemplos de degradação, podem-se citar a difração óptica, a subamostragem, movimentos relativos, ruídos, entre outros [3].

A fim de apresentar resultados consolidados em relação a melhora na qualidade das imagens tratadas, será uitilizada a métrica *Structural Similarity*(SSIM) [4]. O SSIM mede a similaridade entre duas imagens e funciona como uma medida de qualidade de uma imagem em relação à outra, considerando características do sistema visual humano. O SSIM assume um valor no intervalo $[-1,1]$, sendo igual a um quando as duas imagens são iguais e zero quando são muito distintas [4]. A Figura 1 apresenta diversas imagens, submetidas a diferentes tratamentos de restauração, com o seu valor SSIM associado. É possível observar a imagem em baixa resolução (LR – *low resolution*) e a imagem original em alta resolução. A imagem que mais se aproxima da imagem original é a que possui SSIM de 0,823.

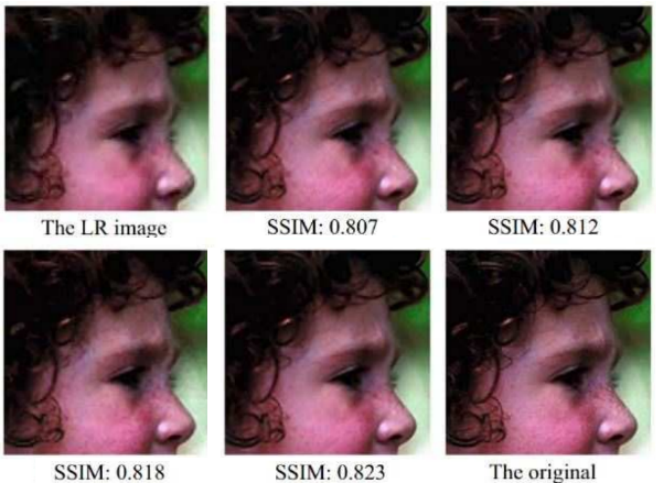


Figura 1: Exemplo de reconstrução de imagem em super-resolução. Fonte: Figura 9 de [8].

Para reconstruir a imagem em alta resolução, é possível utilizar uma ou mais imagens em baixa resolução. O foco deste trabalho está na reconstrução de imagens a partir de uma única imagem de baixa resolução. A seguir estão os objetivos do plano de pesquisa inicial.

1.1 Objetivos

Os objetivos principais estabelecidos nesse projeto de pesquisa são:

1. Realizar um estudo de redes neurais aplicadas a reconstrução de imagens em super-resolução a partir de uma imagem de baixa resolução;
2. Gerar imagens de baixa resolução a partir de conjuntos de imagens de alta resolução considerando bancos de imagens diversos;
3. Estudar diferentes soluções baseadas em redes neuronais, focando principalmente nas redes MLP (perceptron multicamada, do inglês *multilayer perceptron*), CNN (rede neural convolucional, do inglês *convolutional neural network)* e modelos gerativos, incluindo a GAN (rede adversária gerativa, do inglês *generative adversarial network*);
4. Verificar a possibilidade de treinar as redes para maximizar o SSIM;
5. Redigir um relatório final que exponha a metodologia, as técnicas estudadas e os resultados das análises comparativas que possa ser útil a futuros pesquisadores do grupo.

2 Resultados Alcançados

Com o objetivo de alcançar a super-resolução de imagens, três modelos distintos de redes neurais foram considerados. Partiu-se primeiramente de uma rede do tipo perceptron multicamada (*Multilayer Perceptron*–MLP) [5], seguiu-se então para a rede neural convolucional (*Convolutional Neural Network*–CNN) [5] e terminou-se com a rede adversária gerativa (*Generative Adversarial Network*–GAN) [6].

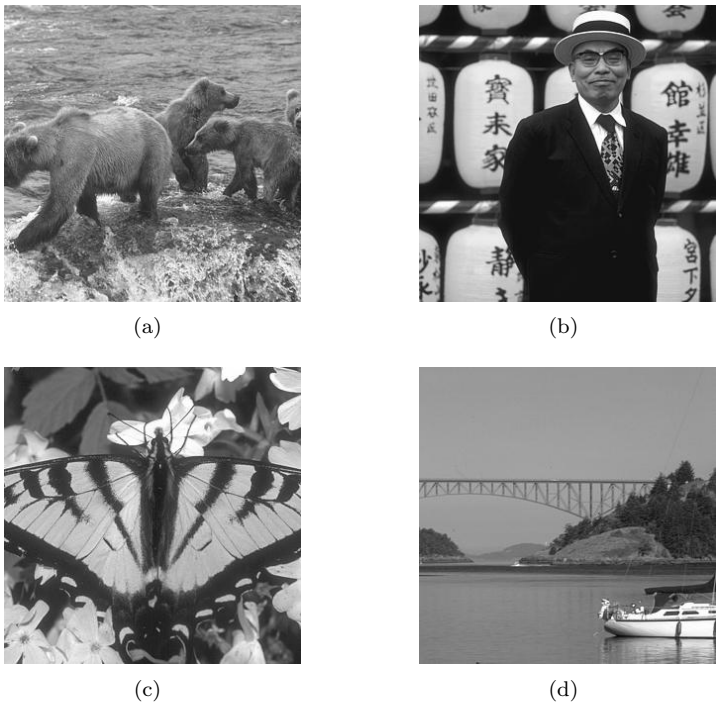
Para construir as redes e realizar os testes necessários foi utilizada a linguagem de programação *Python*. Com o auxílio da biblioteca *PyTorch*, além do uso das bibliotecas *open-cv* e *pillow* para tratamento de imagens especificamente. Bibliotecas auxiliares como *numpy* e *scikit* também foram empregadas para tratamento de tensores e obtenção de métricas comparativas, respectivamente. As subseções a seguir descrevem essas redes, assim como os resultados obtidos no uso de cada uma delas.

2.1 Aplicação da Rede MLP para Super-Resolução

Em uma primeira abordagem, o processo de aumento de resolução de uma imagem foi realizado com uma rede neural do tipo MLP, como mostrado a seguir.

2.1.1 Organização do Banco de Dados

Na abordagem com a MLP, foram utilizadas 10 imagens de treino retiradas de [7]. As imagens foram transformadas para *grayscale* e recortadas, utilizando a biblioteca *open-cv* em *python*, para o tamanho 300×300 pixels. A Figura 2 apresenta algumas das imagens utilizadas.



(a) (b)

(c) (d)

Figura 2: Imagens utilizadas para treino da rede neural MLP.

Em seguida, foi realizada a redução da resolução da imagem original, para que a rede desenvolvida possa ser treinada para aumentar a resolução dessa imagem reduzida, tendo a imagem original como saída ideal. Para esse objetivo, é utilizada uma operação de subamostragem com um filtro bicúbico, transformando a imagem de 300×300 pixels, para 150×150 pixels.

Em seguida, a biblioteca *open-cv* faz a leitura das imagens em 256 níveis

de cinza, variando de 0, preto, até 255, branco. Posteriormente, dividem-se esses valores por 255, para que os valores de cada pixel varie entre 0 e 1.

Com a matriz da imagem agora normalizada, as imagens de treino são divididas em blocos para permitir que os dados de cada imagem possam ser tratados em partes. Utilizando essa abordagem, é possível aumentar a quantidade de informações a serem utilizadas, além de reduzir o tamanho necessário para a rede.

No caso, foram selecionados quadrados de tamanho 15×15 pixels, fazendo com que existam no total 100 blocos por imagem, 1000 blocos considerando todas as 10 imagens. Em seguida, as matrizes de 15×15 são vetorizadas, fazendo com que efetivamente formem vetores de tamanho (225,1). Esses vetores são então organizados em um *Dataloader* para seguirem para a rede. A Figura 3, abaixo, representa o processo utilizado.

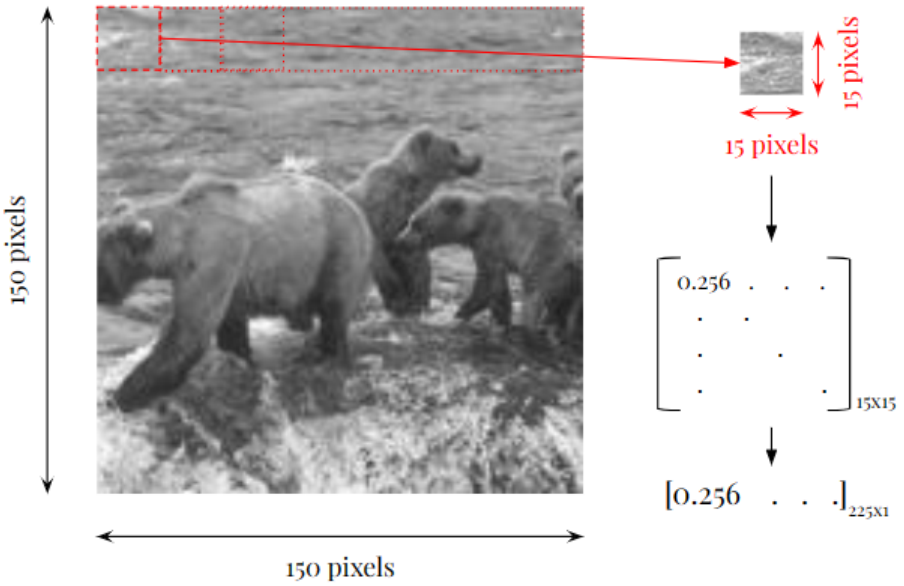


Figura 3: Processamento da imagem para rede MLP.

2.1.2 Metodologia do processo escolhido

A príncipio, um primeiro objetivo é retornar a imagem para a resolução original, ou seja, de 150×150 pixels para 300×300 pixels.

Fazendo uso de um filtro bicúbico, sem utilizar a rede MLP, a restauração da imagem é classificada com um valor de SSIM próximo de 0.846, e pode ser observada na Figura 4.

(a) Original, SSIM: 1.000 (b) Interpolação Bicúbica, SSIM: 0.846

Figura 4: Comparação imagem original e restauração por interpolação bicúbica.

Os vetores dos blocos 15×15 são alimentados a uma camada de entrada, de tamanho 225, condizente com a dimensão (225,1) de cada vetor de entrada, como visto na Figura 3. Por conseguinte, a saída deve ter 900 neurônios, uma vez que se deseja obter blocos de 30×30 pixels, obtidos com vetores de saída de tamanho (900,1). Como numa rede MLP clássica, foram implementadas camadas ocultas.

Após as 10 imagens terem sido usadas no treinamento da rede, uma décima primeira imagem foi utilizada para teste. O mesmo procedimento foi aplicado a essa imagem, ou seja, ela foi dividida em 100 vetores de dimensão (225,1), que foram submetidos a rede, retornando 100 vetores de dimensão (900,1). Por fim os blocos foram reconstruídos em matrizes, que então foram concatenadas para retornar a imagem restaurada obtida.

2.1.3 Configurações da Rede e Testes

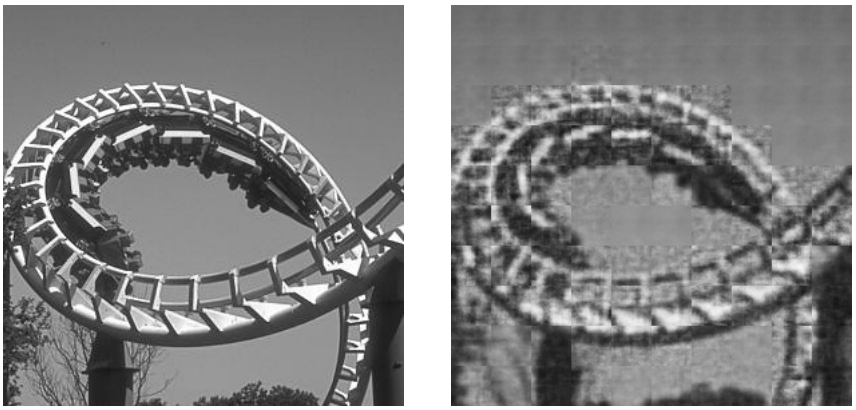
Para esse problema foi utilizada a função custo de aprendizado MSE. Além disso, foi utilizado o otimizador Adam na etapa de treinamento e tamanho de *mini-batch* $nb = 50$. A função de ativação considerada, para a camada de saída, foi a sigmoidal, uma vez que se deseja obter valores dos pixels no intervalo [0, 1].

Considerando duas camadas ocultas na rede MLP, variaram-se os números de neurônios dessas camadas, a função de ativação dos neurônios a menos dos de saída e o passo de adaptação. Os valores desses parâmetros e do SSIM obtido com 1000 épocas estão mostrados na Tabela 1. Como pode ser observado, o melhor resultado encontrado foi com a configuração $[225 - 80 - 40 - 900]$,

Tangente Hiperbólica como função de ativação nos neurônios da camada de entrada, assim como nos neurônios das camadas ocultas e passo de adaptação de 0,001.

Tabela 1: Comparação de resultados de restauração com MLP.

Dimensão da Rede	Função de ativação	Passo de adaptação	SSIM
[225-10-5-900]	ReLU	0,01	0,447
[225-10-5-900]	ReLU	0,001	0,461
[225-40-20-900]	ReLU	0,001	0,527
[225-40-20-900]	Tangente Hiperbólica	0,001	0,558
[225-80-40-900]	Tangente Hiperbólica	0,01	0,499
[225-80-40-900]	ReLU	0,001	0,577
[225-80-40-900]	Tangente Hiperbólica	0,001	0,602



(a) Original (b) Restauração MLP

Figura 5: Comparação imagem original e restauração com MLP.

Na Figura 5, são mostradas as imagens original e a de teste obtida com a melhor configuração da Tabela 1. É possível observar nessa figura que, apesar da rede ser capaz de restaurar a imagem para a resolução original, há uma perda significativa na qualidade da imagem. Além disso, como observado pela tabela apresentada, esse método chega a um máximo de 0,602 de SSIM, enquanto o filtro bicúbico chega a 0,846. Desse modo, o uso da MLP não apresenta vantagem efetiva considerando a configuração proposta, uma vez que um filtro de interpolação simples leva a um resultado de restauração superior, além de ser mais rápido por não necessitar do tempo de treino para realizar o processo e exigir menos recursos computacionais.

Vale ressaltar que talvez seja possível uma configuração que leve a um valor de SSIM superior aos 0,602 obtidos. Porém, dadas as vantagens de se trabalhar com uma rede CNN em processamento de imagens, considerou-se uma abordagem utilizando uma rede desse tipo como mostrado na subseção seguinte.

2.2 Aplicação de CNN para Super-Resolução

Com o intuito de melhorar os resultados previamente obtidos com a rede MLP, foi explorado o uso da rede CNN para realizar o processo de super-resolução. Esse tipo de rede é mais recomendado para a atuação sobre tratamento de imagens [9] devido a sua maior capacidade de interação com os dados de forma matricial, sendo cada dado tratado tanto individualmente quanto com relação a suas proximidades [9]. Primeiramente foi explorado o uso de redes do tipo CNN sem a adição de conexões residuais, que serão aqui denominadas de "CNN Simples", e posteriormente com a adição de conexões residuais.

2.2.1 Organização do Banco de Dados

Para a aplicação nessa rede, foi necessário um número maior de imagens de treinamento do que utilizado anteriormente na aplicação com a rede MLP da Seção 2.1.1. O banco de dados adotado foi o mesmo [7], porém, dessa vez, sendo utilizado em sua totalidade. Esse banco de dados possui um total de 200 imagens, que novamente foram tratadas com as bibliotecas *open-cv* e *pillow* em *python*. Essas imagens foram transformadas para o tamanho de 150×150 pixels e reduzidas para apenas 1 canal em escala de cinza.

Em seguida, essas mesmas imagens foram submetidas a um processo de subamostragem utilizando um filtro bicúbico, sendo então transformadas para o tamanho de 75×75 pixels. Além disso, a leitura da imagem foi feita em 256 níveis de cinza, variando de 0 a 255, e normalizada, dividindo seus valores por 255, fazendo com que então a imagem tivesse os valores de seus pixels variando de 0 a 1.

Após o pré-processamento, as imagens foram dividas em dois grupos: (i) um possuindo as imagens consideradas em alta resolução, 150×150 pixels, que foram determinadas como saída ideal da rede e (ii) outro grupo possuindo as imagens em baixa resolução, 75×75 pixels, que foram determinadas como entrada da rede, ou seja, as imagens a serem melhoradas com a intervenção da rede.

As 200 imagens do banco de dados foram dividas ainda em dois grupos: 150 foram utilizadas para treinamento e 50 no teste dos modelos.

2.2.2 Rede CNN Simples

A seguir serão apresentados os modelos utilizados e resultados obtidos considerando a rede CNN sem o uso de conexões residuais.

2.2.2.1 Arquitetura e Configurações

Afim de obter um melhor conhecimento da estrutura e funcionamento da rede CNN, ela foi considerada inicialmente sem conexões residuais. Nesse processo as imagens fornecidas em baixa resolução foram consideradas numa dimensão ligeiramente aumentada do banco de dados descrito, sendo da dimensão de 100×100 pixels, devido a simplicidade da rede. Essas imagens foram passadas por uma ou mais camadas convolucionais com um número de filtros F_l, que realizavam a convolução em 2 dimensões da imagem. A convolução transforma a imagem para a dimensão de $F_l \times 100 \times 100$, utilizando $stride = 1$ e *zero-padding* para manutenção do tamanho da imagem nas duas últimas dimensões. Em seguida aplica-se a função de ativação ReLU [10]. A imagem resultante entra na próxima camada que leva a uma saída com dimensões $4 \times 100 \times 100$. Neste ponto a função *pixel-shuffle* [11] é utilizada para modificar as dimensões da imagem através da justaposição ordenada de pixels de diferentes camadas da imagem original. A Figura 6 ilustra esse processo, em que se transforma uma imagem de dimensão $c \times h \times w$, selecionando um valor t, para a dimensão $\frac{c}{t^2} \times h.t \times w.t$.

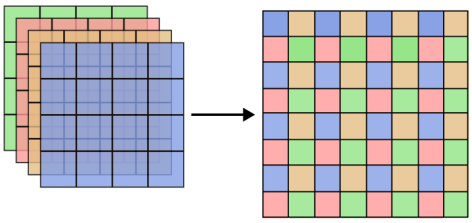


Figura 6: Esquema de funcionamento da função *pixel-shuffle*, com parâmetro $t = 2$.

Desse modo, se chega à imagem final de tamanho 200×200. A Figura 7 representa um esquema de funcionamento dessa rede.

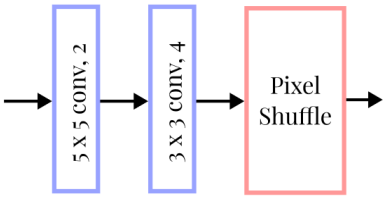


Figura 7: Esquema de arquitetura de rede CNN sem conexão residual.

Nas etapas de teste dessa rede foram consideradas duas abordagens com diferentes números de camadas. Em uma delas foi considerada apenas uma camada com 4 filtros e em outra duas camadas, uma primeira de 2 filtros, seguida por uma de 4 filtros. Ambas possuindo 4 filtros na última camada com o intuito de, no final do processamento da rede, obter uma imagem com 4 canais para ser aplicada a função de *pixel-shuffle*. A função custo considerada em todos os casos foi o MSE e o otimizador escolhido foi o Adam, com parâmetros $\eta = 10^{-3}$, $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$ e $\epsilon = 10^{-8}$.

2.2.2.2 Testes e Resultados

Os resultados de SSIM encontrados para esses testes foram menores ou iguais ao valor de SSIM de uma dada imagem restaurada por meio do método de interpolação bicúbica. Os valores de SSIM obtidos com as diferentes configurações propostas, para uma mesma imagem, podem ser obervado na Tabela 2 e as imagens comparadas estão apresentadas na Figura 8. O valor de SSIM obtido com o método de interpolação bicúbica para essa imagem foi de 0,84.

(a) Original

(b) Interpolação Bicúbica

(c)

(d)

(e)

(f)

(g)

(h)

Figura 8: Comparação de imagem sob tratamentos diferentes em CNN Simples. Imagens (c) a (h) foram obtidas com as configurações da Tabela 2.

Tabela 2: Comparação de resultados de restauração com CNN não residual em diferentes configurações.

Legenda	Número de épocas	Nº de camadas	Dimensão do(s) *kernel(s)*	SSIM
(c)	1000	2	5×5, 3×3	0,81
(d)	1000	2	7×7, 5×5	0,81
(e)	1000	1	5×5	0,80
(f)	1000	1	3×3	0,80
(g)	3000	1	5×5	0,80
(h)	3000	1	3×3	0,80

Desse modo, como o uso da CNN "simples", não apresentou vantagem significativa em comparação com o uso da interpolação bicúbica na restauração de imagens em alta resolução, passou-se a explorar o uso de conexões residuais como detalhado a seguir.

2.2.3 Rede CNN Residual

A seguir serão apresentados os métodos aplicados e resultados obtidos no caso da rede CNN com o uso de conexões residuais em duas arquiteturas distintas.

2.2.3.1 Arquitetura e Configurações

Na utilização da rede CNN residual foram propostas duas abordagens de arquiteturas diferentes para a super-resolução. A primeira abordagem, apresentada na Figura 9, propõe uma rede neural residual que recebe como entrada a imagem já nas dimensões desejadas. Nessa abordagem, como primeiro passo, as imagens do banco de dados de treinamento foram submetidas a um processo de sobreamostragem com um filtro bicúbico. Desse modo, as imagens foram tratadas dentro da rede com as dimensões já desejadas de 150×150 pixels.

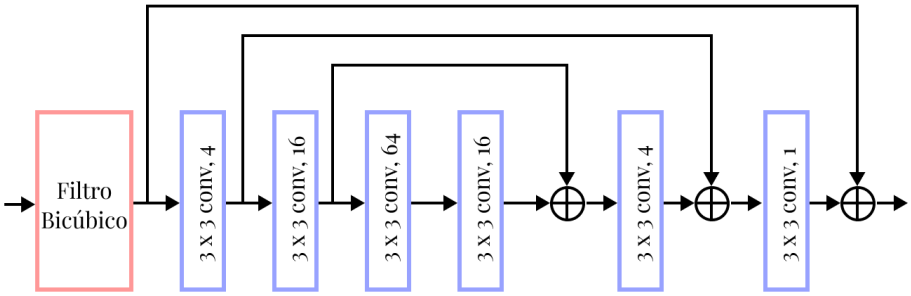


Figura 9: Primeira abordagem de arquitetura de rede CNN residual.

A saída do filtro bicúbico, denotada por O_0, entra na rede neural em si. Nessa primeira abordagem, a rede é composta de 6. Cada camadas convolucional, denotada por N_l, gera saída O_l com $l = 1,2 \ldots 6$. Além disso, considera-se a adição de resíduo nas camadas mais profundas e a função de ativação ReLU em todas as camadas. A princípio, uma imagem em escala de cinza de dimensão $1 \times 150 \times 150$ é fornecida à primeira camada convolucional, N_1. Nessa camada, 4 filtros de tamanho 3×3 realizam o processo de convolução em 2 dimensões com $stride = 1$. Para garantir a manutenção do tamanho da imagem, foi empregada a técnica de *zero-padding* com $z_p = 1$. Desse modo, foi gerada como saída O_1 uma imagem de dimensão $4 \times 150 \times 150$. Posteriormente, essa imagem entra na camada N_2 e é submetida então a 16 filtros de tamanho 3×3. Considerando o mesmo procedimento da camada N_1 obtém-se como saída O_2 uma imagem de dimensão $16 \times 150 \times 150$. Esse processo é continuado pelas camadas, aumentando-se o número de filtros até 64, na camada N_3, quando então a rede passa a diminuir o número de filtros simetricamente. Assim, gera-se na saída da camada N_6 uma imagem de dimensão 150×150. Além disso, as entradas das 3 primeiras camadas (N_1, N_2, N_3) foram armazenadas para serem utilizadas como realimentação residual nas saídas das 3 camadas finais (N_4, N_5, N_6) respectivamente. Essas entradas são somadas aos tensores de saída obtidos em cada uma das camadas mais profundas. O procedimento é descrito como

$$O_l = \begin{cases} Conv(O_{l-1}), & l = 1,2,3 \\ Conv(O_{l-1}) + O_{6-l}, & l = 4,5,6, \end{cases}$$

onde $Conv$ representa o processo convolucional realizado em cada camada, acompanhado pela aplicação da função ReLU.

A seguda abordagem, por sua vez, utilizou na entrada a imagem nas dimensões em baixa resolução, realizando o processo de *upscale* em uma camada final com o uso da função *pixel-shuffle*, como representado na Figura 10. Nessa abordagem, as imagens foram tratadas na rede com dimensão reduzida de 75×75 pixels, submetidas a 4 camadas convolucionais e foi mantida a função ReLU como função de ativação em todas as camadas.

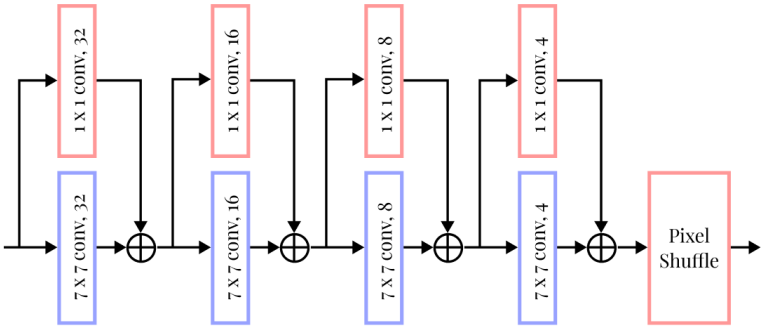


Figura 10: Segunda abordagem de arquitetura de rede CNN residual.

A imagem em escala de cinza de dimensão $1 \times 75 \times 75$ é submetida a uma camada convolucional composta de 32 filtros de dimensão 7×7, com *stride=1* e $z_p = 3$, gerando uma saída naquela camada de dimensão $32 \times 75 \times 75$. Paralelamente nessa mesma camada, uma outra convolução é realizada com 32 filtros de dimensão 1×1, levando a um tensor de mesma dimensão. Desse modo, essas duas saídas, dimensionalmente coerentes, são somadas e fornecidas como entrada para a próxima camada. Na camada seguinte, o procedimento é repetido, porém com um número de filtros reduzido igual a 16. Nessa camada, são geradas duas saídas paralelas de dimensão $16 \times 75 \times 75$, que são somadas e passadas para a camada seguinte. Esse processo é realizado de maneira análoga em cada camada, reduzindo o número de filtros para 8 na camada seguinte, e para 4 na última camada. Ao chegar na última camada, as saídas são somadas novamente e é gerado um tensor de dimensão $4 \times 75 \times 75$. Esse tensor é então submetido à função de *pixel-shuffle*, resultando numa imagem final de dimensão 150×150.

Para essa abordagem utilizando redes neurais residuais, também foi explorado o uso de uma função custo diferente da utlizada previamente. Para isso, adaptou-se uma função custo baseada no SSIM [12]. Como a métrica do SSIM representa o grau de similaridade entre duas imagens, sendo retornado o valor 1 como representativo de igualdade entre elas, a função custo J proposta é definida por

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

onde $\boldsymbol{Y}$ representa a matriz da imagem obtida pela rede e $\boldsymbol{D}$ representa a matriz da imagem ideal em alta resolução. Além disso, para a otimização do treinamento foi utilizado o algoritmo Adam com parâmetros $\eta = 10^{-3}$, $\beta_1 = 0{,}9$,

$\beta_2 = 0{,}999$ e $\epsilon = 10^{-8}$ e foi considerado um número de épocas de treinamento igua a 10^3.

2.2.3.2 Testes e Resultados

Os resultados obtidos com o uso de conexões residuais, em conjunto com a CNN, foram consideravelmente superiores em comparação com os resultados obtidos pela CNN Simples. A Figura 11 apresenta os resultados comparativos, com a imagem original e com a imagem obtida com o método de interpolação bicúbica, das imagens obtidas pela reconstrução em super-resolução com a CNN Residual em ambas as arquiteturas propostas.

(a) Original

(b) Interpolação Bicúbica

(c) 1ª abordagem

(d) 2ª abordagem

Figura 11: Comparação imagem original, interpolação bicúbica e super-resolução com CNN Residual.

No caso apresentado, as imagens de super-resolução apresentam um resultado visivelmente mais nítido com o uso de redes CNN Residuais do que

apenas utilizando a interpolação bicúbica. Isso é comprovado pelo resultado obtido pelo SSIM das imagens, comparadas à imagem original. No caso da interpolação bicúbica foi obtido um SSIM de 0,675, enquanto com a 1ª abordagem de CNN Residual se obteve 0,712 e com a 2ª abordagem um valor de 0,708.

Para garantir que o resultado positivo não era exclusivo a certas imagens, uma análise dos valores de SSIM foi realizada em todo o conjunto de teste utilizado. Alguns exemplos dessas imagens estão representadas na Figura 12, podendo todas as imagens de validação comparadas serem encontradas em https://github.com/Adelkend/SRRNN. Foram então gerados 4 gráficos representativos da distribuição de SSIM nas imagens do conjunto de dados. A Figura 13 apresenta um histograma combinado dessas distribuições, assim como seus valores médios de SSIM.

Nesse gráfico, "Bicubic" se refere ao resultado obtido por interpolação bicúbica, "CNN1" ao resultado advindo da 1ª abordagem com CNN Residual e "CNN2" do resultado advindo da 2ª abordagem com CNN Residual. O gráfico também é acompanhado dos valores médios de SSIM obtidos em cada método, apresentando 0,696 para o método de interpolação bicúbica e um mesmo valor de 0,719 para ambos os métodos utilizando a CNN Residual.

Além desse gráfico de distribuição combinada, foram gerados gráficos de cada distribuição separadamente, como apresentado na Figura 14.

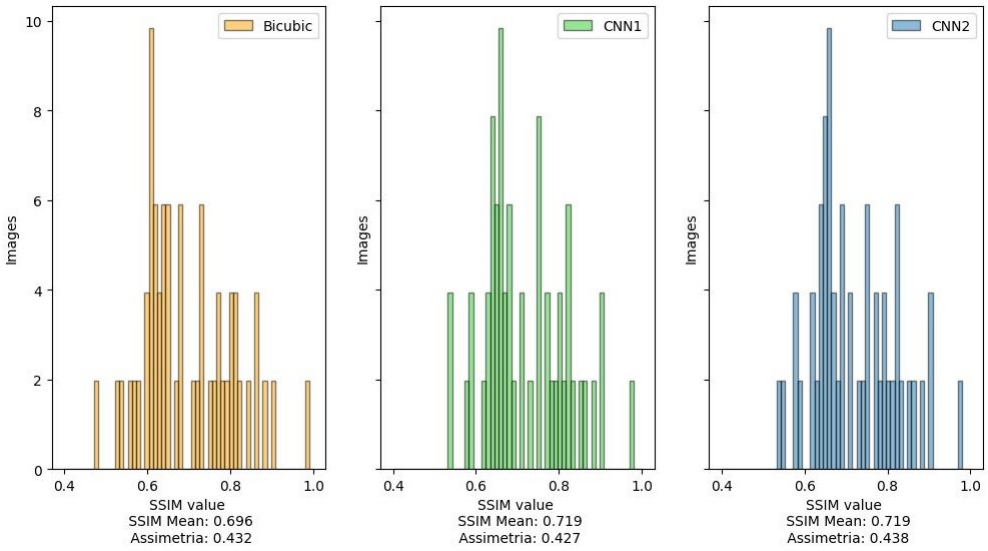
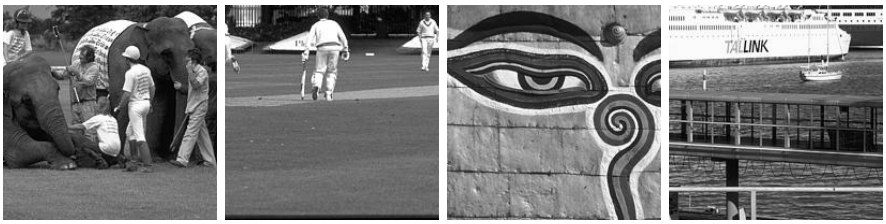


Figura 14: Histogramas separados das distribuições de SSIM com 3 métodos diferentes para super-resolução.



(a) Original

(b) Interpolação Bicúbica

(c) 1ª abordagem

(d) 2ª abordagem

Figura 12: Comparação de imagens do set de validação sob diferentes tratamentos.

Em adição à apresentar os valores observados no gráfico da Figura 13, os histogramas da Figura 14 acompanham o valor da assimetria [13] das distribuições. Esse valor aponta que, no caso da 2ª abordagem, há um maior valor de assimetria do que na 1ª abordagem. Desse modo, há a indicação de uma cauda direita, onde o valor de SSIM é maior, mais pesada na 2ª abordagem do que na

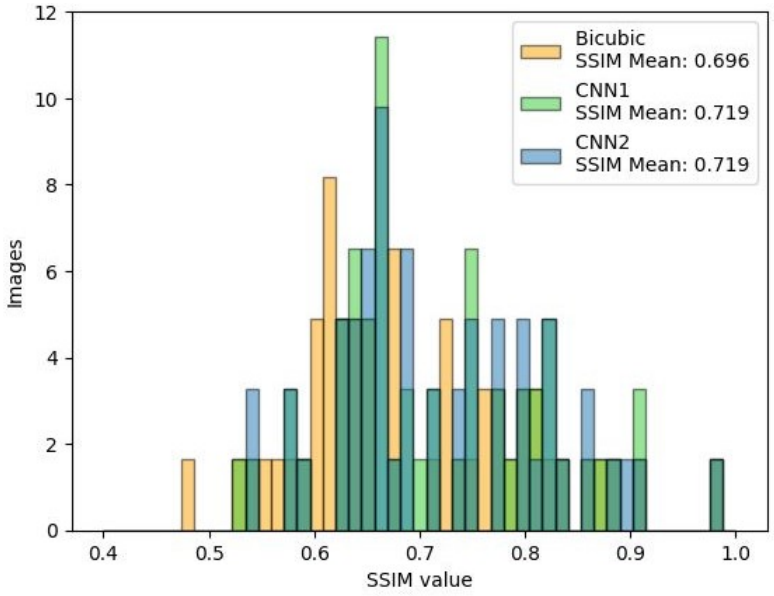
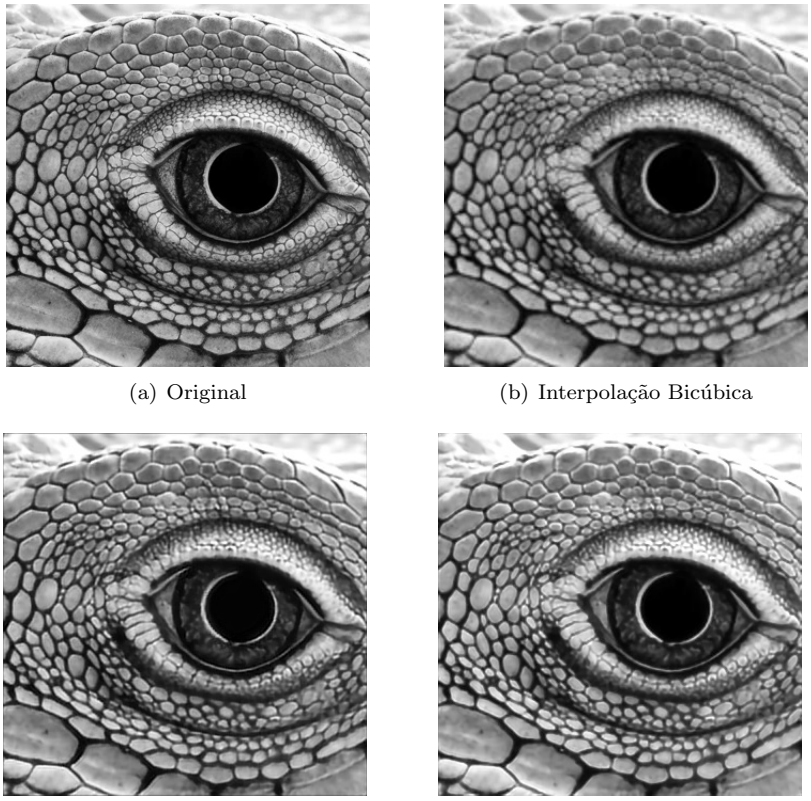


Figura 13: Histograma combinado das distribuições de SSIM com 3 métodos diferentes para super-resolução.

1ª, indicando uma têndencia que favorece a 2ª abordagem.

Uma outra análise comparativa também foi realizada com essas redes, porém atuando sobre uma imagem fora do conjunto de teste. O objetivo desse teste foi, além de comparar mais profundamente as eficácias das redes, garantir que a rede seria capaz de atuar sobre imagens diferentes da base de dados. Para isso, uma imagem de dimensão 340×340 pixels foi selecionada e submetida a um pré-processamento semelhante ao das imagens de treinamento, sendo transformada para escala de cinza e obtida uma versão em baixa resolução com o uso de um filtro bicúbico. Após o pré-processamento, foram realizados os testes de super-resolução, sendo seus resultados apresentados na Figura 15.

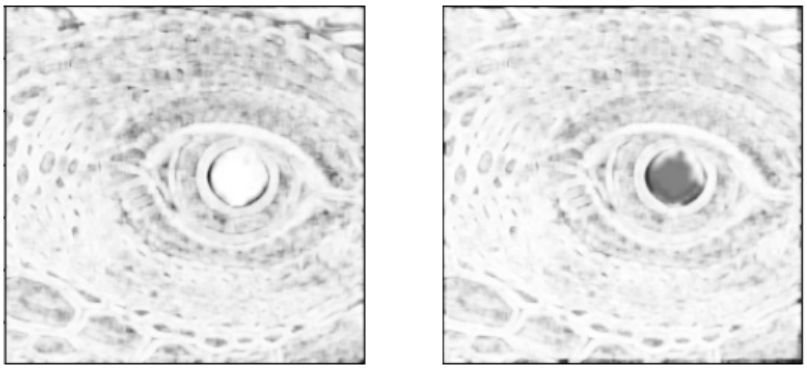


(a) Original (b) Interpolação Bicúbica

(c) 1ª abordagem (d) 2ª abordagem

Figura 15: Comparação imagem original, interpolação bicúbica e super-resolução com CNN Residual de imagem fora do banco de dados de treinamento.

Comparando-se com a imagem original o SSIM da imagem obtida pela interpolação bicúbica é igual a 0,877, enquando a 1ª abordagem obteve 0,897 e a 2ª abordagem um valor de 0,891. Esses valores, a princípio, discordavam da hipótese inicial de que a 2ª abordagem apresentava uma vantagem quando comparada com a 1ª abordagem. Desse modo, realizou-se uma averiguação dos mapas de SSIM, gerados pela representação gráfica da matriz de SSIM obtida na comparação das imagens, apresentados na Figura 16.



(a) 1ª abordagem (b) 2ª abordagem

(c)

Figura 16: Comparação de mapas de SSIM gerados pelas abordagens de CNN Residual de imagem fora do banco de dados de treinamento.

A partir dessas imagens foi possível verificar onde cada cada método desmonstrou maior eficácia. Nessas representações, com o auxílio da escala fornecida na Figura 16(c), os pontos pretos representam um SSIM igual a 0, ou seja, um local em que a imagem restaurada se distância da imagem original, enquanto os pontos brancos representam um SSIM igual a 1, indicando uma igualdade com os valores dos pixels da imagem original.

Comparando a 2ª abordagem com a 1ª, é possível observar que esta possui regiões mais claras nas partes correspondentes a pele do lagarto. Entretanto, na região do olho do lagarto, a 2ª abordagem apresenta um SSIM muito menor que o da 1ª abordagem, indicando uma alta taxa de erro nessa região da imagem. Desse modo, apesar da 2ª abordagem apresentar um SSIM local maior na imagem como um todo, esta acabou apresentando um valor menor de SSIM médio quando comparada com a 1ª abordagem. Acredita-se que esse erro possa ser causado pela tentativa de ação dos filtros das camadas da rede neural, no ato de adição de resíduo, quando na verdade a ação deveria ser de preservação apenas do resultado anterior. Além disso, como na 1ª abordagem, o processo se inicia com a interpolação bicúbica, é de se esperar que o resultado seja melhor na parte do olho do lagarto. Vale-se ressaltar, também, que a elevada taxa de erro é pouco perceptível na região do olho do lagarto na imagem gerada pela 2ª abordagem, principalmente por ser tratar de uma região escura. O SSIM,

apesar de apresentar uma medida de extrema utilidade na análise de qualidade das imagens, possui certas limitações, principalmente na comparação de regiões muito escuras [14].

Os resultados obtidos com o uso de redes do tipo CNN, principalmente com a implementação de conexões residuais, foi consideravelmente superior aos obtidos previamente com o uso de redes do tipo MLP. Além disso, os resultados com esse tipo de rede já são superiores também aos resultados obtidos pelo método clássico de interpolação bicúbica. Assim, o uso de redes CNN Residuais apresenta um passo positivo referente ao alcance da reconstrução de imagens em super-resolução. Com ainda espaço para aprimoramento desses resultados, foram exploradas em seguida o uso de redes do tipo GAN.

2.3 Aplicação de GAN para Super-Resolução

Com o objetivo de obter resultados superiores aos obtidos com a CNN, passou-se então a utilizar a rede do tipo GAN. As redes do tipo GAN têm se tornado ponto de grande relevância no estudo de super-resolução de imagens devido ao seu treinamento aperfeiçoado, gerado pela competição de duas redes neurais distintas. No caso da aplicação para super-resolução, a rede chamada de "Gerador" se encarrega de gerar imagens em alta resolução, enquanto a rede denominada de "Discriminador" tenta discernir as imagens reais das imagens em alta resolução geradas pelo Gerador.

Essa competição entre as redes gera vantagens no treinamento para a configuração da rede do tipo GAN [15] e para a obtenção da super-resolução de imagens. A seguir serão descritos os banco de dadoss adotados para o treinamento da rede, assim como as arquiteturas adotadas para as redes do tipo Gerador e Discriminador, seguido pelos resultados obtidos.

2.3.1 Organização do Banco de Dados

Para a rede GAN proposta, foi utilizado, a princípio, o mesmo banco de dados utilizado anteriormente para a rede do tipo CNN [7], com imagens em baixa resolução representadas com 75×75 pixels e imagens em alta resolução representadas com 150×150 pixels. Os procedimentos adotados para tratamento das imagens, transformando-as em escala de cinza, assim como a normalização dos valores dos pixels para [0,1], seguem a descrição observada na Subseção 2.2.1.

Posteriormente, porém, se fez necessário o uso de um banco de dados maior que o utlizado anteriormente [6]. Desse modo, um novo banco de dados foi organizado com 5000 imagens do banco de dados "ImageNet" [16], enquanto

o banco de dados utilizado anteriormente, "Berkeley" [7], possuia apenas 200. Nesse novo banco de dados, foram consideradas imagens em escala de cinza. O objetivo era transformar imagens de baixa resolução com dimensão 100×100 em imagens de alta resolução com dimensão 200×200.

2.3.2 Arquitetura e Configurações das Redes

Foram exploradas diferentes configurações para as redes do Gerador e Discriminador. Em todas as configurações abordadas, considerou-se a rede do tipo Adversária Gerativa Condicional (*Conditional Generative Adversarial Networks* – CGAN) [17]. Nesse tipo de rede, além do espaço latente fornecido em algumas abordagens, é adicionada uma informação principal. No problema de super-resolução, foi fornecida a imagem em baixa resolução, para auxiliar a rede a obter a saída desejada, correspondendo a imagem em alta resolução.

Para a arquitetura da rede do Gerador, foi utilizada uma CNN Residual correspondente a 2ª abordagem explorada na Subseção 2.2.3.1 e presentada na Figura 10. No caso do Discriminador, foi utilizada uma rede que combina camadas convolucionais, como na rede CNN e camadas de perceptrons, como na rede MLP.

Nesse tipo de rede, a imagem é submetida inicialmente ao processo descrito pela CNN de 2ª abordagem quando passando pelo Gerador. Assim como na CNN utilizada anteriormente, para esse estudo inicial da GAN, foi considerado o banco de dados Berkeley. Nesse caso, uma imagem de dimensão reduzia, correspondente a 75×75 pixels, por exemplo, é fornecida a rede e repassada através de uma sequência de camadas convolucionais. O número de canais dessas camadas aumentam e em seguida são reduzidos de modo a obter-se a dimensão de $4 \times 75 \times 75$ na saída da última camada convolucional. Nessa fase, é aplicada a função de *pixel-shuffle* produzindo a imagem na dimensão desejada de 150×150 pixels.

Durante o treinamento dessa rede, imagens reconstruídas em alta resolução pelo Gerador são fornecidas em conjunto com as imagens originais para o Discriminador. O Discriminador, por sua vez, é treinado para diferenciar as imagens reais das imagens geradas.

Na rede do Discriminador, representada esquematicamente na Figura 17, as imagens são submetidas a 3 blocos convolucionais. O número de canais dessas imagens é aumentado em cada camada convolucional. Em um primeiro momento, uma imagem tem sua dimensão expandida de 150×150 pixels para $2 \times 150 \times 150$ com o auxílio de uma camada convolucional composta de 2 filtros de dimensão 3×3, acompanhada da aplicação da função ReLU na saída. Em

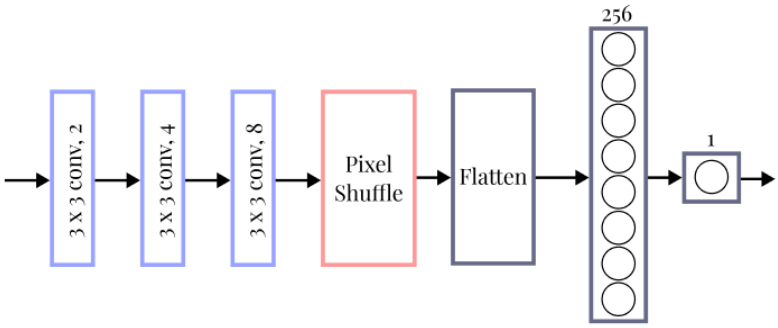


Figura 17: Arquitetura proposta para rede do Discriminador da GAN.

seguida, o mesmo processo é realizado para expandir a imagem para o tamanho de $4 \times 150 \times 150$ com o uso de 4 filtros de dimensão 3×3. Por fim, é realizado um processo análogo, no qual chega-se a uma imagem de tamanho $8 \times 150 \times 150$ pixels. Após obter a imagem nessas dimensões, o tensor representativo é então vetorizado com o uso da função *flatten*, gerando um vetor de dimensão 180000×1. Esse vetor é então alimentado a uma camada de rede MLP com 256 perceptrons, seguido pela aplicação de uma função LeakyReLU [28], com parâmetro *negative-slope=0,2*. As saídas dessa camada são então fornecidas a um perceptron, que gera uma saída única, a qual é submetida a uma função sigmoidal que indica qual o apontamento do Discriminador sobre a imagem fornecida. O valor 1 indicando que a imagem é real, enquanto o valor 0 indicando que a imagem é gerada.

2.3.2.1 Testes e Resultados

Para realizar os testes com a rede do tipo GAN, foram exploradas diversas abordagens de função custo. As funções custo exploradas foram a Entropia Cruzada (*Binary Cross Entropy*–BCE) [6], o MSE e a *Wasserstein Loss* [29]. Além disso, foi explorado o uso de uma função *Perceptual Loss* [30] customizada, definida como

$$l^{SR} = l_X^{SR} + 10^{-3}.l_{Gen}^{SR}, \tag{2}$$

onde l_X^{SR} é definida como *content-loss* e l_{Gen}^{SR} como *adversarial-loss*. No cálculo da chamada *content-loss* é utilizada uma rede pré-treinada chamada *Visual Geometry Group* (VGG) [30]. Essa rede é carregada com os pesos pré-treinados e é responsável pelo cálculo da distância euclidiana entre a imagem gerada e a imagem ideal [30]. O cálculo da *content-loss* é definido então como uma soma

do cálculo do MSE pixel a pixel nas imagens comparadas com o valor retornado pela saída da rede VGG. A chamada *adversarial-loss* é a função custo definida pela relação probabilística dada por

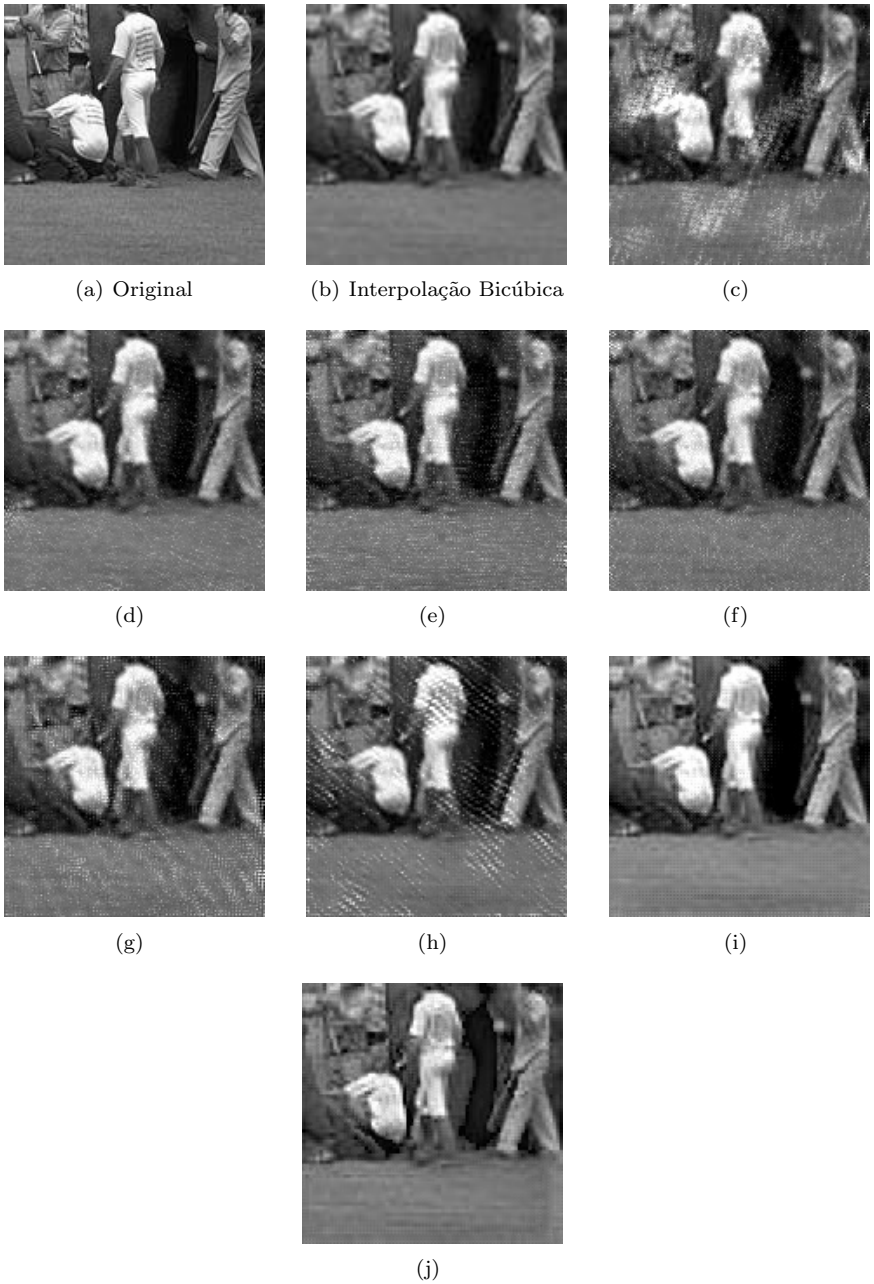
$$l_{Gen}^{SR} = \sum_{n-1}^{N} -\log D(G(I^{LR})), \tag{3}$$

onde $D(G(I^{LR}))$ indica a probabilidade de a imagem gerada, $G(I^{LR})$, ser uma imagem real. Para normalizar os valores obtidos dessas funções a serem somadas, pesos arbitrários são definidos e multiplicados em cada fator integrante como mostrado na Eq. (2).

Os resultados obtidos com essas configurações de redes, utilizando o banco de dados Berkeley, podem ser observados na Tabela 3 e na Figura 18, onde a imagem de interpolação bicúbica selecionada apresenta um valor de SSIM igual a 0,739.

Tabela 3: Comparação de resultados de restauração com GAN em diferentes configurações.

Legenda	Número de épocas	Função Custo	Otimizador	SSIM
(c)	30	BCE	Adam	0,496
(d)	30	MSE	Adam	0,585
(e)	30	Wasserstein	Adam	0,510
(f)	40	Wasserstein	Adam	0,544
(g)	40	Wasserstein	RMSprop	0,519
(h)	60	Wasserstein	RMSprop	0,525
(i)	25	Perceptual	Adam	0,641
(j)	50	Perceptual	Adam	0,670



(a) Original (b) Interpolação Bicúbica (c)

(d) (e) (f)

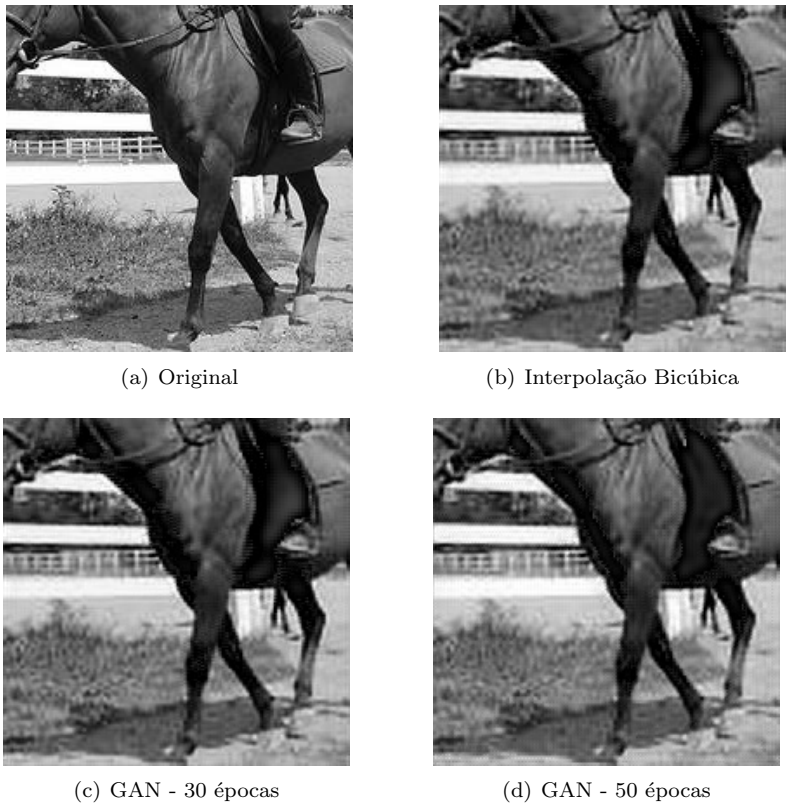
(g) (h) (i)

(j)

Figura 18: Comparação imagem original, interpolação bicúbica e super-resolução com GAN. As imagens (c) a (j) seguem as configurações da Tabela 3.

A partir dessas imagens geradas, verifica-se que a rede treinada com a

função *Perceptual Loss* leva a resultados superiores àqueles encontrados utilizando as demais funções apresentadas. Desse modo, a fim de melhorar esses resultados, que ainda se encontram inferiores aos obtidos com o uso da interpolação bicúbica, foi realizado o treinamento da rede com o banco de dados ImageNet, que possui um número de imagens selecionadas para treinamento 25 vezes maior. Os resultados obtidos utilizando esse banco de dados, para um número de épocas igual a 30 e 50 épocas, utilizando a função *Perceptual Loss* pode ser observado na Figura 19.



(a) Original

(b) Interpolação Bicúbica

(c) GAN - 30 épocas

(d) GAN - 50 épocas

Figura 19: Testes da rede do tipo GAN utilizando o banco de dados ImageNet e função custo *Perceptual Loss*.

Para esses testes, o valor obtido de SSIM foi de 0,634 para a restauração por Interpolação Bicúbica, 0,634 para a GAN com 30 épocas e 0,633 para a GAN com 50 épocas. Portanto, nas configurações propostas, a rede GAN não apresentou vantagens significativas para a super-resolução quando comparada com os resultados obtidos utilizando a interpolação bicúbica.

3 Conclusão

Durante o tempo de realização desse projeto de iniciação científica diversos tópicos relevantes ao estudo de aprendizado de máquina e modelos de redes neurais, assim como suas aplicações, foram abordados. A princípio, um estudo do funcionamento de elementos fundamentais de redes neurais como perceptrons, algoritmos de treinamento, funções custo etc. foi realizado para se estabelecer uma base de conhecimento sobre redes neurais. Além disso, foi realizado um estudo sobre processos clássicos para aumento de resolução de imagens utilizando filtros de interpolação bilinear e bicúbica.

Em seguida, foi explorada a aplicação de uma rede MLP para a super-resolução de imagens a partir dos conhecimentos adquiridos. O processo adotado para realizar essa aplicação impôs uma segmentação rigorosa à imagem, que foi divida em blocos menores para processamento. Desse modo, as imagens reconstruídas apresentaram resultados inferiores, utilizando a métrica do SSIM para comparação, em relação as imagens obtidas a partir dos métodos clássicos de interpolação.

Com os resultados inferiores obtidos com a rede do tipo MLP, passou-se para a aplicação de uma rede mais usual para tratamento de imagens, CNN. Utilizando essa rede, foram testadas diversas abordagen distintas. Primeiramente foi proposta a utilização de uma rede CNN denominada de CNN Simples, que não fazia uso de conexões residuais. Os resultados obtidos com esse tipo de rede ainda se encontravam inferiores aos resultados obtidos pelos métodos de interpolação, portanto passou-se a explorar a implementação desse tipo de rede com conexões residuais.

Fazendo uso das redes denominadas CNN Residuais, foram obtidos resultados superiores para aplicação da super-resolução. Duas arquiteturas distintas foram propostas para esse processo utilizando conexões residuais entre as camadas convolucionais. Complementarmente à essas conexões, foi definida um função custo customizada baseada na métrica de comparação do SSIM. Os resultados obtidos com essas configurações foram superiores aos obtidos com os métodos clássicos de interpolação e o estudo teve um artigo aceito para apresentação no Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT) a ser realizado em outubro de 2023.

Por fim, foi realizado o estudo da aplicação de redes do tipo GAN para super-resolução. Diferentes tópicos e estruturas de GAN foram estudados para essa aplicação, com ênfase na rede do tipo CGAN, e diversos testes foram realizados. O uso de diferentes funções custo foi um aspecto central dessa aplicação, sendo abordadas 4 funções custo diferentes nos testes. Dentre essas funções, a

função definida como *Perceptual Loss*, que utilizava em sua computação a saída de uma outra rede pré-treinada denominada VGG, obteve os resultados mais proeminentes. Então, foi realizado um estudo aprimorado sobre ela, aplicando-a sobre a mesma rede, porém utilizando um banco de dados mais extenso do que anteriormente. Os resultados obtidos com essa rede foram semelhantes aos resultados obtidos com o método de interpolação bicúbica. Desse modo, a rede, nas configurações testadas, não apresentou vantagem para a super-resolução, tendo em vista o elevado custo computacional do treinamento dessa rede.

Portanto, tendo em vista o cronograma proposto para o projeto de pesquisa, todos os tópicos previstos foram abordados. Os resultados obtidos com a rede do tipo CNN foram os mais relevantes levando-se em consideração a métrica do SSIM utilizada e a comparação feita com os métodos clássicos de aumento de super-resolução de imagens.

A Fundamentação Teórica

Primeiramente, nesta seção, serão explicados alguns conceitos gerais em termos de redes neurais. Um problema de classficação binária será abordado para demonstrar o uso de uma rede neural e trazer uma ilustração sobre sua relevância. Posteriormente, serão apresentados alguns filtros para processamento de imagens, junto com seu funcionamento, que serão relevantes para a pesquisa.

A.1 Modelo de Neurônio e a Rede Perceptron Multicamada

Em problemas de classificação, principalmente, é comum utilizar redes neurais. A forma de uso, assim como suas vantagens serão abordadas nas subseções seguintes.

A.1.1 Problemas de Classificação Binária

As redes neurais são comumente utilizadas em problemas de classficação. Nesses problemas, a rede é treinada para reconhecer o padrão de classficação, a fim de automatizar o processo de decisão.

Um problema clássico abordado em aprendizado de máquina é o "problema de classificação das meia-luas" [18]. Esse problema consiste em um conjunto de dados de teste, gerados pela seleção de alguns parâmetros geométricos que definem a forma das meia-luas, como ilustrado na Figura 20. O objetivo a ser alcançado é gerar uma curva de separação no plano xy, tal que cada metade da lua seja definida como pertencente a uma determinada região.

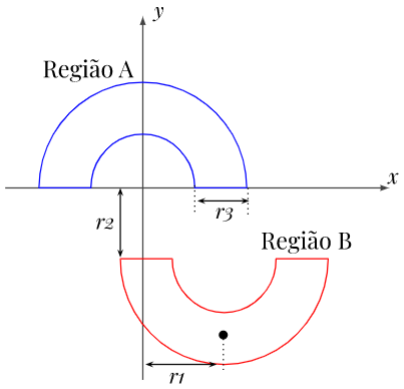
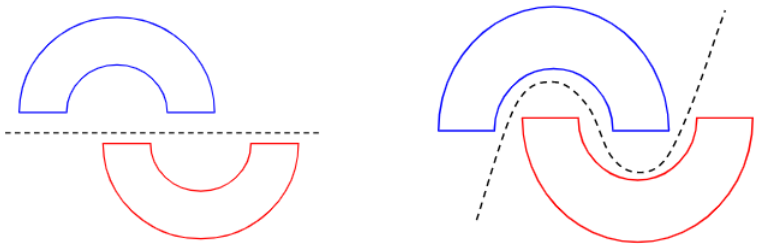


Figura 20: Esquema geométrico para problema de classificação de meia-luas.

No problema, é proposto que, dados r_1, r_2 e r_3, definem-se duas variáveis aleatórias θ, distribuída no intervalo [0, π] e ρ, distribuída no intervalo $[r_1 - r_3/2, r_1 + r_3/2]$. Essas variáveis determinam a distribuição dos pontos gerados no plano xy. Os pontos da Região A possuem sinal desejado $d = 1$, enquanto os prontos da Região B possuem sinal desejado $d = -1$.

O problema das meia-luas pode ser abordado de diferentes maneiras, dependendo dos valores dos parâmetros r_1, r_2 e r_3 [18]. Desse modo, o problema pode resultar em possíveis curvas de classificação lineares ou não, como apresentado na Figura 21. No caso (a) da Figura 21, uma função linear do tipo $f(x) = y$ com $y \in$]-r_2, 0] soluciona o problema, sem a necessidade da atuação de uma rede neural. Para o problema do caso (b), seria necessária uma solução polinomial [18], mais complexa do que a apresentada para (a). A resolução prática desse problema será abordada posteriormente.



(a) Linear (b) Não Linear

Figura 21: Linearidade das classificações das Meia-Luas.

Portanto, um algoritmo dotado de adaptabilidade, como uma rede neural, é ideal para a solução desse problema [19]. Desse modo, será apresentado a seguir, a unidade fundamental de uma rede neural.

A.1.2 Perceptron de Rosenblatt

O modelo do neurônio para aplicação computacional, denominado "perceptron", foi originalmente proposto pelo psicólogo americano Frank Rosenblatt [20] numa tentativa de simplificar a implementação de um "processo de decisão", baseado no funcionamento de um neurônio biológico. Nesse neurônio artificial, o sinal recebido faz o papel do estímulo do neurônio biológico. Ponderando-se esse sinal com pesos e somando-se o resultado a um *bias* obtém-se o sinal de entrada da função não linear, chamada de função de ativação. A saída dessa função faz o papel do estímulo transmitido pelo neurônio.

Matematicamente, considerando M pesos ω_k, k = 1,2,...M e o *bias* b_1, define-se o vetor

$$\boldsymbol{\omega} = [b_1\ \omega_1\ \omega_2\ \cdots\ \omega_M\ ]^T\ . \tag{4}$$

em que $(\cdot)^T$ representa a transposição. Definindo o vetor de entrada como

$$\mathbf{x} = [\,1\ x_1\ x_2\ \cdots\ x_M]^T\ , \tag{5}$$

e a função de ativação não linear por $\varphi(\cdot)$, obtém-se a saída do neurônio, dada por

$$y = \varphi(\mathbf{x}^T\boldsymbol{\omega}). \tag{6}$$

Um esquema ilustrativo desse processo pode ser observado na Figura 22. Cabe observar que como o *bias* b_1 foi incorporado ao vetor de pesos, a primeira posição do vetor de entrada deve ser igual a 1.

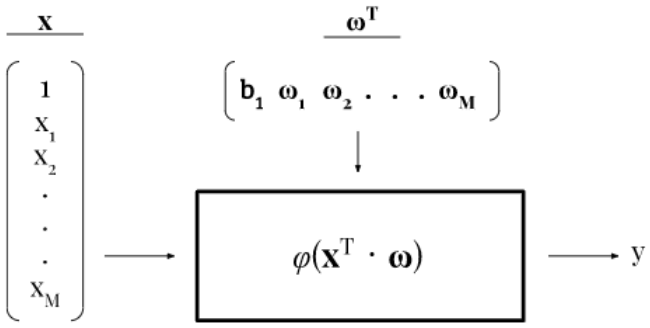


Figura 22: Esquema de funcionamento de um perceptron.

A função de ativação tem o objetivo de definir o domínio da saída do neurônio, restringindo essa saída e, idealmente, mantendo-a dentro do escopo do problema que se deseja abordar. No caso do perceptron de Rosenblatt, a função de ativação utilizada é a função sinal, o que faz com que a curva de separação seja linear como a da Figura 21 (a). Além disso, um único neurônio não é capaz de resolver problemas mais complexos e por isso foram organizados em camadas dando origem à rede perceptron multicamdada [19], como será explicado posteriormente. As funções de ativação mais utilizadas em redes neurais são descritas a seguir.

A.1.3 Funções de Ativação

As funções de ativação são componentes essenciais em redes neurais artificiais. Essas funções fazem parte de cada neurônio da rede para introduzir não-linearidade nas saídas [21]. A função de ativação determina se um neurônio deve ser ativado ou não, com base em uma combinação linear das entradas que recebe.

Existem diversas funções de ativação utilizadas em redes neurais [22]. Algumas das mais comuns são: *Rectified Linear Unit*, conhecida como ReLU, a função Sigmoidal e a Tangente Hiperbólica (TanH). Suas equações são dadas, respectivamente, por:

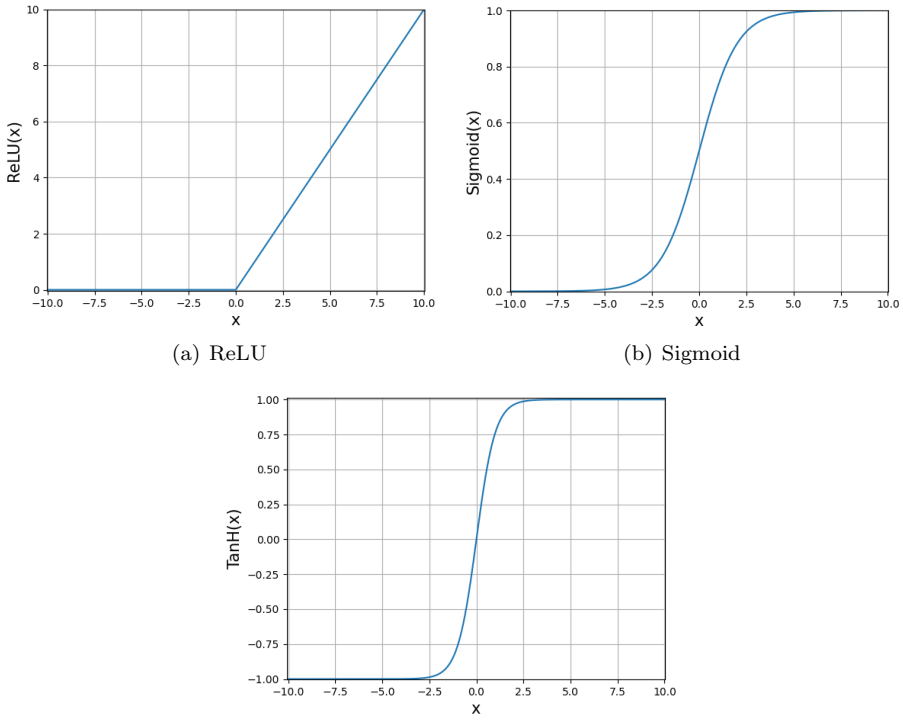
$$\text{ReLU(x)} = max(0,x), \tag{7}$$

$$\text{Sigmoidal(x)} = \frac{1}{1+e^{-x}}, \tag{8}$$

$$\text{TanH(x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \tag{9}$$

Os gráficos dessas funções de ativação podem ser observados na Figura 23.

Diferente da função sinal, utilizada no perceptron de Rosenblatt, as funções sigmoidal e TanH possuem derivada em todos os pontos. Por isso, ambas são utilizadas em redes neurais com frequência, uma vez que o algoritmo de atualização dos pesos depende da derivada dessa função. Já a função ReLU não possui derivada definida em $x = 0$. No entanto, ela tem sido frequentemente utilizada já que ajuda o algoritmo de atualização dos pesos a evitar mínimos locais. Cabe observar que existem outras funções de ativação, como a softmax, utilizada em problemas de classificação multiclasse e diferentes variantes da ReLU. Vamos abordar essas funções apenas se forem utilizadas neste trabalho.



(a) ReLU

(b) Sigmoid

(c) TanH

Figura 23: Gráficos de funções de ativação.

A.1.4 Rede Perceptron Multicamada

Para problemas mais complexos, o uso de um único neurônio não permite atingir bons resultados. Nesses problemas, se faz necessária a criação de uma rede que associa uma gama de neurônios artificiais em série e em paralelo, incorporando combinações mais complexas entre saídas dos neurônios e pesos associados. Essa rede é chamada de Perceptron Multicamada (*Multilayer Perceptron* – MLP).

Numa rede MLP, os neurônios da camada j são associados aos neurônios das camadas $j-1$ e $j+1$, sem associação dos neurônios da mesma camada. Essas camadas podem ser divididas em 3 tipos: camada de entrada, camadas ocultas e camada de saída. Destas, apenas as camadas de entrada e saída podem ser acessadas, como ilustrado na Figura 24. Para denotar as configurações de camadas e números de neurônios, em uma rede MLP, será utilizada a notação $[N_0 - N_1 - N_2 - \cdots - N_j - N_L]$, onde N_0 representa o número de neurônios de entrada, N_j o número de neurônios na camada j e N_L o número de neurônios

na camada de saída. Na Figura 24 é exemplificada uma rede MLP com configuração de camadas [2-3-3-1].

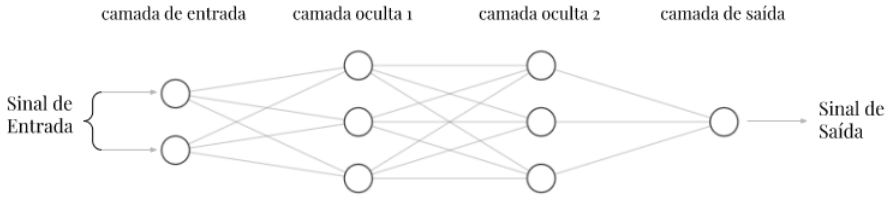


Figura 24: Ilustração de uma rede MLP com configuração [2-3-3-1].

Para realizar uma determinada tarefa, os pesos dos neurônios da rede precisam ser adaptados. Nessa adaptação, utiliza-se uma parte dos dados disponíveis. Essa etapa é chamada de treinamento. Uma outra parte dos dados precisa ser reservada para verificar se o treinamento foi adequado e se a rede consegue efetuar a tarefa com dados diferentes dos usados no treinamento. Nessa etapa, chamada de teste, os pesos não são mais atualizados. O teste proporciona as métricas de desempenho da rede e indica se a rede precisa de ajustes. A seguir, aborda-se o ajuste dos pesos no treinamento.

A.1.5 Algoritmo *Backpropagation* e o otimizador Adam

Uma ferramenta essencial no treinamento de redes neurais é o uso de épocas de treinamento [18]. Considere, por exemplo, que estejam disponíveis N_t dados de treinamento. Muitas vezes a convergência do algoritmo de atualização dos pesos não é atingida com a utilização desses N_t dados uma única vez. Além disso, há aplicações em que é muito difícil gerar mais dados para o treinamento. Diante disso, os N_t dados disponíveis são utilizados mais de uma vez no treinamento. Chama-se *época* cada vez que esses N_t dados são utilizados. Para inserir diversidade, esses dados devem ser embaralhados a cada época para que a sequência dos dados não influencie no treinamento. A cada vez que o algoritmo de aprendizado utiliza todos os N_t dados de entrada, uma época é contada, e a próxima época se inicia, continuamente pelo número ne de épocas definido *a priori*.

Para realizar o cálculo do algoritmo, convém definir os vetores $\boldsymbol{v}^{(j)}$ e $\boldsymbol{y}^{(j)}$, que representam, respectivamente, o vetor de saída dos N_j neurônios da camada j antes da intervenção da função de ativação e o vetor de saída dos N_j neurônios da camada j após a intervenção da função de ativação, com $j = 1,2,\ldots,L$, onde

L indica o número de camadas da rede. Desse modo, os vetores $\boldsymbol{v}^{(j)}$ e $\boldsymbol{y}^{(j)}$ são da forma

$$\boldsymbol{v}^{(j)} = \begin{bmatrix} v_1^{(j)} \\ v_2^{(j)} \\ \vdots \\ v_{N_j}^{(j)} \end{bmatrix}, \boldsymbol{y}^{(j)} = \begin{bmatrix} y_1^{(j)} \\ y_2^{(j)} \\ \vdots \\ y_{N_j}^{(j)} \end{bmatrix}.$$

Além dos vetores já descritos, é preciso definir a matriz de pesos $\boldsymbol{W}^{(j)}$. Essa matriz engloba não só os pesos sinápticos, mas também os *biases* associados, combinando os vetores de pesos definidos em (4) de cada neurônio da camada j, ou seja,

$$\boldsymbol{W}^{(j)} = \begin{bmatrix} \boldsymbol{\omega}_1^{(j)} \\ \boldsymbol{\omega}_2^{(j)} \\ \vdots \\ \boldsymbol{\omega}_{N_j}^{(j)} \end{bmatrix}.$$

Por fim, para associar os vetores de saída $\boldsymbol{y}^{(j)}$ da camada j, como vetores de entrada para a camada $j+1$, um vetor intermediário é criado. Esse vetor $\boldsymbol{x}^{(j)}$ é necessário pois ele insere o valor "1" na primeira posição do vetor, para permitir a utilização do *bias* de acordo com a formulação utilizada. Esse vetor é então definido por

$$\boldsymbol{x}^{(j)} = \begin{bmatrix} 1 \\ \boldsymbol{y}^{(j-1)} \end{bmatrix},$$

com o vetor de entrada da rede dado como

$$\boldsymbol{x}^{(0)} = \begin{bmatrix} 1 \\ x_1^{(0)} \\ x_2^{(0)} \\ \vdots \\ x_{N_0}^{(0)} \end{bmatrix}.$$

O algoritmo *Backpropagation* consiste em duas etapas na fase de treinamento. Primeiramente é realizado o chamado *feedforward*, onde o sinal de entrada é fornecido à rede e transmitido pelas camadas intermediárias até atingir a camada de saída, para gerar o sinal de saída. O sinal de saída de cada camada j é passado para a camada $j+1$ como sinal de entrada, sendo então submetido ao processo descrito pela Figura 22, em cada neurônio, na camada seguinte.

Para as saídas de cada camada, a fórmula

$$\boldsymbol{v}^{(j)}(n) = \boldsymbol{W}^{(j)}(n-1)\boldsymbol{x}^{(j)}(n), \tag{10}$$

explicita o vetor de saída $\boldsymbol{v}^{(j)}$, da camada j na iteração n do algoritmo, sendo a mesma notação utilizada para a matriz de pesos $\boldsymbol{W}^{(j)}$ de cada camada e para os vetores de saídas $\boldsymbol{y}^{(j)}$, com cada vetor $\boldsymbol{y}^{(j)}(n)$ dado por

$$\boldsymbol{y}^{(j)}(n) = \varphi(\boldsymbol{v}^{(j)}(n)). \tag{11}$$

A Figura 25 ilustra o processo descrito.

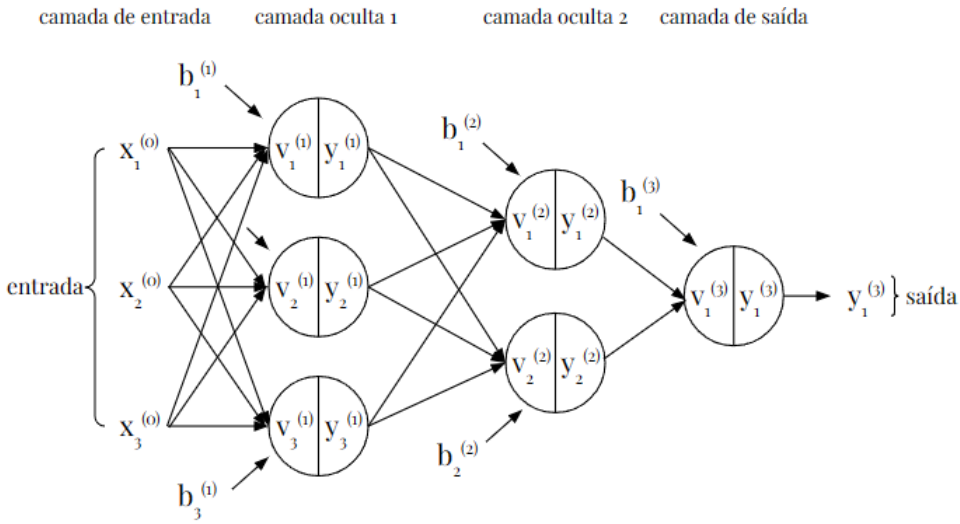


Figura 25: Representação do processo de *feedforward* de uma rede MLP com configuração [3-3-2-1].

Em posse agora do sinal de saída, a rede realiza a atualização dos pesos sinápticos e dos *biases* dos neurônios de cada camada, contidos nas matrizes $\boldsymbol{W}^{(j)}$, por meio da minimização de uma função custo. Para o problema apresentado previamente e estudado a seguir, utilizaremos a função custo do erro quadrático médio (*Mean Square Error* – MSE) [23] dada pela equação

$$\Theta(n) = \frac{1}{N_L}\sum_{i=1}^{N_L}|e_i(n)|^2, \tag{12}$$

onde N_L indica o número de neurônios na última camada e sendo o erro $e_i(n)$, de cada neurônio i, $i = 1,2,\cdots,N_L$ da última camada, dado por

$$e_i(n) = d_i(n) - y_i^{(L)}(n). \tag{13}$$

Na equação acima $d_i(n)$ representa o sinal esperado no neurônio i da camada de saída, fornecido à rede com os dados de treino e $y_i^{(L)}(n)$ representa a saída obtida no mesmo neurônio da camada de saída da rede.

Para realizar a atualização dos pesos, um processo de retropropagação é implementado. Nesse processo, primeiramente calcula-se o gradiente local na camada de saída, seguido pelo gradiente das camadas intermediárias. Então, calcula-se a derivada da função custo, em relação aos pesos da camada por

$$\boldsymbol{\nabla\Theta}^{(j)}(n) = \frac{\partial\Theta(n)}{\partial\boldsymbol{W}^{(j)}(n-1)}. \tag{14}$$

Por fim, é realizado o ajuste dos vetores de pesos de cada camada, levando em conta o gradiente calculado, pela equação

$$\boldsymbol{W}^{(j)}(n) = \boldsymbol{W}^{(j)}(n-1) - \eta\boldsymbol{\nabla\Theta}^{(j)}(n), \tag{15}$$

onde η representa o passo de adaptação.

O algoritmo se repete pelo número de iterações pré-determinado. Além de fazer uso da divisão em épocas, é vantajoso também determinar o padrão de vezes em que ocorre o ajuste de pesos. Em uma primeira observação, é possível suspeitar que o caso estocástico, em que se atualizam os pesos a cada dado de treinamento, seja ideal. Entretanto, raramente essa hipótese se verifica. Os métodos mais comuns utilizam os chamados *batch* ou *mini-batch*. Nesses casos, o vetor gradiente é estimado por um certo intervalo, e então é atualizado usando essa estimativa, ao invés de usar cada dado de treino. No caso do *batch* o número de dados utilizados na estimativa é igual ao número de dados do conjunto de treino, ou seja, há apenas uma iteração dos pesos por época. Enquanto no caso do *mini-batch* um valor menor nb é determinado, tendo então $\frac{N_t}{nb}$ iterações por época. Então, a eficiência da rede é verificada na fase de teste, determinando se os parâmetros escolhidos são adequados ou não, e se a rede é capaz de solucionar o problema de maneira satisfatória.

Devido à não convexidade da função custo, um dos maiores problemas do algoritmo *backpropagation* é que ele pode ficar parado em mínimos locais. Uma das formas de evitar isso é utilizar o otimizador Adam.

No otimizador Adam, são introduzidas quatro novas matrizes auxiliares, que atuarão diretamente na etapa de atualização de pesos. Essas matrizes são denominadas $\boldsymbol{S(n)}$, $\boldsymbol{S_c(n)}$, $\boldsymbol{V(n)}$ e $\boldsymbol{V_c(n)}$. A Matriz $\boldsymbol{S(n)}$ da camada j é dada por

$$\boldsymbol{S}^{(j)}(n) = \beta_2 \boldsymbol{S}^{(j)}(n-1) + (1-\beta_2)[\boldsymbol{\nabla\Theta}^{(j)}(n)]^{\odot 2}, \tag{16}$$

onde $\boldsymbol{\nabla\Theta}^{(j)}(n)$ é o gradiente calculado em (14), β_2 é um hiperparâmetro a ser fornecido à rede, tal que $0 << \beta_2 < 1$, e $\odot 2$ denomina a operação de elevar todos os elementos da matriz ao quadrado. Da mesma maneira, A matriz $\boldsymbol{V(n)}$ da camada j é dada por

$$\boldsymbol{V}^{(j)}(n) = (1-\beta_1)\sum_{k=1}^{n} \beta_1^{(n-k)} \boldsymbol{\nabla\Theta}^{(j)}(k), \tag{17}$$

onde β_1 é um hiperparâmetro, tal que $0 << \beta_1 < 1$. Em posse das matrizes $\boldsymbol{S(n)}$ e $\boldsymbol{V(n)}$, calculam-se então $\boldsymbol{S_c(n)}$ e $\boldsymbol{V_c(n)}$, na camada j, por meio das fórmulas

$$\boldsymbol{V_c}^{(j)}(n) = \frac{1}{1-\beta_1^n} \boldsymbol{V}^{(j)}(n) \tag{18}$$

e

$$\boldsymbol{S_c}^{(j)}(n) = \frac{1}{1-\beta_2^n} \boldsymbol{S}^{(j)}(n). \tag{19}$$

Desse modo, a atualização dos pesos, utilizando o otimizador Adam, passa a ser dada por

$$\boldsymbol{W}^{(j)}(n) = \boldsymbol{W}^{(j)}(n-1) - \eta \boldsymbol{V_c}^{(j)}(n) \oslash [[\boldsymbol{S_c}^{(j)}]^{\odot \frac{1}{2}} + \epsilon \mathbf{1}], \tag{20}$$

de modo que $\oslash$ identifica a divisão de Hadamard, onde cada elemento da matriz a esquerda é dividido pelo respectivo elemento da matriz a direita, ϵ identifica uma constante positiva pequena, utilizada para evitar as divisões por 0, e $\mathbf{1}$ identifica a matriz unitária com as dimensões necessárias para realizar a operação proposta.

Com os fundamentos da rede MLP estabelecidos, a seguir será resolvido o problema anteriormente proposto de divisão de meia-luas [18].

A.1.6 Rede MLP na resolução de problema de classificação de Meia-Luas

Para a abordagem do problema utilizaremos os valores de $r_1 = 10$, $r_2 = -4$ e $r_3 = 6$. Os pontos das Regiões A e B foram gerados de forma aleatória utilizando a biblioteca numpy como explicado na Seção 2.1.1. Foram gerados 1000 pontos para a fase de treino, 500 para a Região A e 500 para a Região B, e 2000 pontos para teste, 1000 para a Região A e 1000 para a Região B.

Primeiramente, destacamos que essa questão não pode ser resolvida dados quaisquer valores de r_1, r_2 e r_3, utilizando um único neurônio. Para os parâmetros abordados, por exemplo, utilizando 2 neurônios para captar os dados em x e y na camada de entrada e 1 único neurônio como camada de saída, sem utilizar camadas ocultas, obtém-se a curva de separação da Figura 26. Pode-se ver claramente que essa solução não resolve o problema.

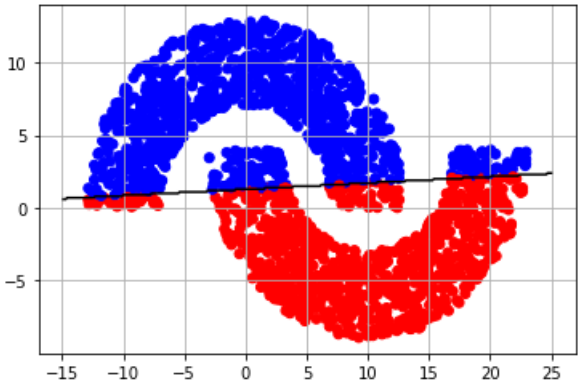


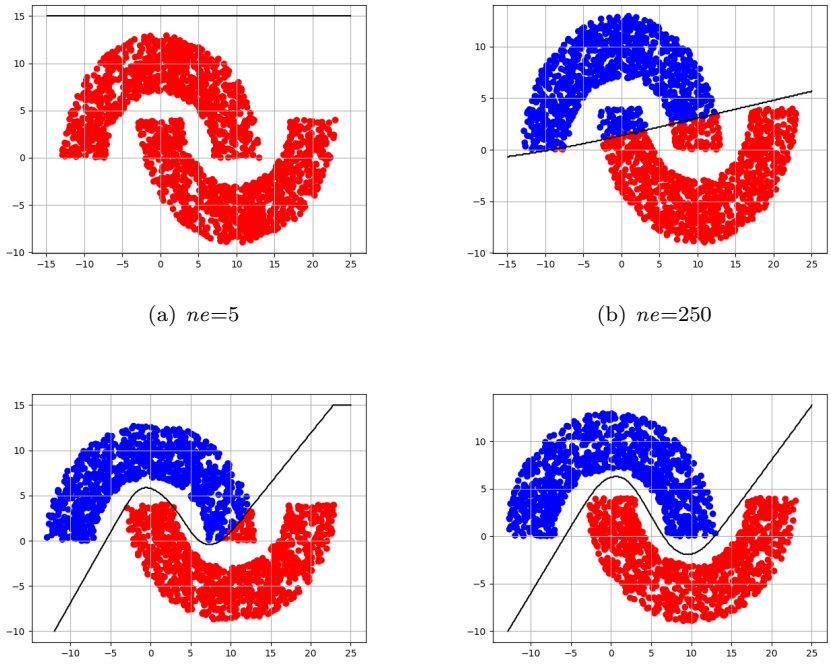
Figura 26: Divisão de meia-luas com rede de configuração [2-1].

A solução buscada então passa a ser uma rede MLP com camadas ocultas. O algoritmo utilizado para adaptação dos pesos é o *backpropagation* e a função custo é o MSE. Foram escolhidos para a rede: passo de adaptação η=0.1 e uma configuração de camadas [2-3-2-1].

A função ReLU é a mais comumente utilizada como função de ativação. Entretanto, para o problema em questão, não é a ideal, pois o contra-domínio dessa função se encontra no intervalo $[0,+\infty)$. Desse modo, as saídas do neurônio final acabariam sendo muito variadas, e essa função não atingiria o valor de -1, da Região B. Por outro lado, a função sigmoidal, que possui seu gráfico apresentado na Figura 23 (b), possui um contra-domínio limitado, que condiz melhor com as condições do problema. Entretanto, essa função apresenta um problema similar ao da ReLU em termos de atingir o valor de -1. Diante disso, a função considerada em todos os neurônios foi a tangente hiperbólica, indicada na Figura 23 (c), que possui como contra-domínio o intervalo $[-1, 1]$. O uso de outras funções de ativação nas camadas intermediárias não foi considerado no experimento proposto, porém é possível realizá-lo, mantendo a função tangente hiperbólica no neurônio de saída, utilizando outras funções nos neurônios das camadas ocultas.

Por fim, para a solução deste problema, basta determinar o número de

épocas, ne, a qual a rede será submetida. Para a atualização do gradiente, será utilizado o modelo de treinamento *mini-batch* com tamanho nb=50. Logo os pesos são atualizados 20 vezes a cada época, com o uso do gradiente calculado. A Figura 27 ilustra as curvas de separação com os dados de teste, sem utilizar o otimizador Adam, considerando uma variação do número de épocas, ou seja, $ne = 5$, $ne = 250$, $ne = 500$ e $ne = 600$.



(a) ne=5 (b) ne=250

(c) ne=500 (d) ne=600

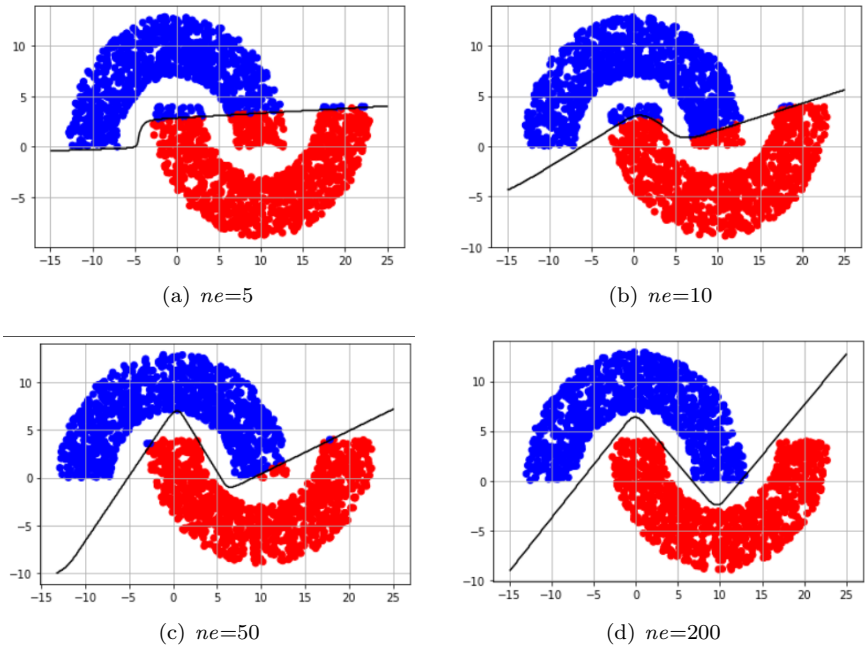
Figura 27: Curvas de separação geradas por variação no número de épocas de treinamento, sem otimizador Adam.

As taxas de acerto percentual, para os números de épocas considerados na Figura 27 são mostradas na Tabela 4. Como apresentado, sem utilizar o Adam, o algoritmo demora um número elevado de épocas para convergir, sendo a taxa de acerto nas primeiras épocas consideravelmente baixa.

Tabela 4: Relação de Número de épocas e Taxa de acerto para problema das Meia-Luas, sem otimizador Adam.

Número de épocas (ne)	Taxa de acerto (%)
5	55
250	92
500	98
600	100

A Figura 28, em comparação com os resultados apresentados na Figura 27, apresenta o mesmo algoritmo, acrescido do otimizador Adam. Os números de épocas considerados foram diferentes para melhor representar o problema. Nesse caso, foram considerados: $ne = 5$, $ne = 10$, $ne = 50$ e $ne = 200$.



(a) ne=5 (b) ne=10

(c) ne=50 (d) ne=200

Figura 28: Curvas de separação geradas por variação no número de épocas de treinamento, com otimizador Adam.

As taxas de acerto percentual, para os números de épocas considerados na Figura 28 são mostradas na Tabela 5. Como pode ser observado, mesmo com um número baixo de épocas, a taxa de acerto ultrapassa 90%, além de o processo de convergência da rede ser mais rápido. Isso se dá devido à utilização do otimizador Adam.

Tabela 5: Relação de Número de épocas e Taxa de acerto para problema das Meia-Luas, com otimizador Adam.

Número de épocas (ne)	Taxa de acerto (%)
5	92
10	95
50	99
500	100

Explorados os conceitos da MLP e resolvido o problema anteriormente proposto, a seguir, será apresentada a segunda parte da fundamentação teórica, na qual são explorados os filtros para processamento de imagens.

A.2 Filtros para processamento de Imagens

Antes de aplicar soluções com redes neurais, é comum fazer uso de algumas funções de filtragem, seja para pré-processamento ou para obter um *upscale* da imagem de maneira rápida. Em geral, essas funções descrevem filtros que levam em conta o valor do pixel analisado e de sua vizinhança para estimar o valor de outros pixels na imagem final. A seguir, descrevem-se os filtros mais comuns utilizados em super-resolução de imagens.

A.2.1 Filtro Bilinear

O filtro bilinear interpola a matriz que representa a imagem nas coordenadas x e y. Considere o pixel $P(x,y)$ cujo valor deseja-se estimar com a interpolação. Para isso, utilizam-se as informações dos pixels do entorno de $P(x,y)$ na forma de um retângulo [24], como mostrado na Figura 29. Os vértices do retângulo são dados por V_{11}, V_{12}, V_{22} e V_{21}.

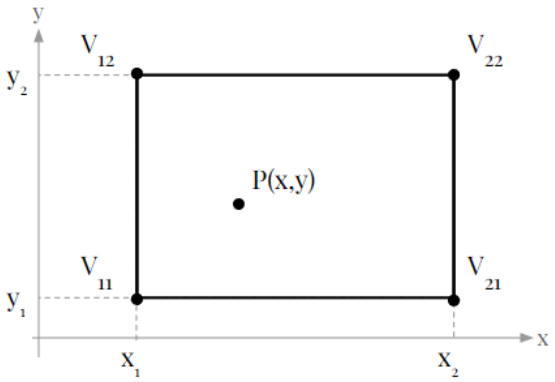


Figura 29: Esquema de seleção para filtro bilinear.

Em seguida, são necessárias duas interpolações ao longo do eixo x, gerando variáveis intermediárias R_1 e R_2, e uma interpolação ao longo do eixo y [24].

As interpolações na direção x são dadas por

$$R_1(x) = \frac{x_2 - x}{x_2 - x_1} V_{11} + \frac{x - x_1}{x_2 - x_1} V_{21}, \tag{21}$$

e

$$R_2(x) = \frac{x_2 - x}{x_2 - x_1} V_{12} + \frac{x - x_1}{x_2 - x_1} V_{22}, \tag{22}$$

onde V_{mn} indica o valor do pixel na posição (m,n) do retângulo de seleção e x_k indica o valor x em cada um dos 4 pixel do retângulo com k variando entre 1 e 2. Em posse das variáveis intermediárias, realizamos a interpolação em y, obtendo o valor no ponto P de acordo com a fórmula

$$P(x,y) = \frac{y_2 - y}{y_2 - y_1} R_1 + \frac{y - y_1}{y_2 - y_1} R_2. \tag{23}$$

Juntando as equações (21)–(23), obtém-se

$$P = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} x_2 - x & x - x_1 \end{bmatrix} \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix}. \tag{24}$$

Apesar do resultado dessa interpolação ser razoável, esse filtro costuma levar a resultados ligeiramente inferiores ao da interpolação bicúbica [25], apresentado a seguir. Por isso, a interpolação bilinear não é muito utlizada na prática.

A.2.2 Filtro Bicúbico

Similarmente à interpolação bilinear, a interpolação bicúbica utiliza informações obtidas de pixels próximos do pixel que se deseja estimar [26]. A diferença é que nesse caso são utilizados 16 pixels para calcular o novo pixel ao invés dos 4 utilizados anteriormente. Para isso é considerada uma grade de tamanho 4×4, como representada na Figura 30, onde $f(x,y)$ indica o valor do pixel em (x,y) com $x,y = -1,0,1,2$.

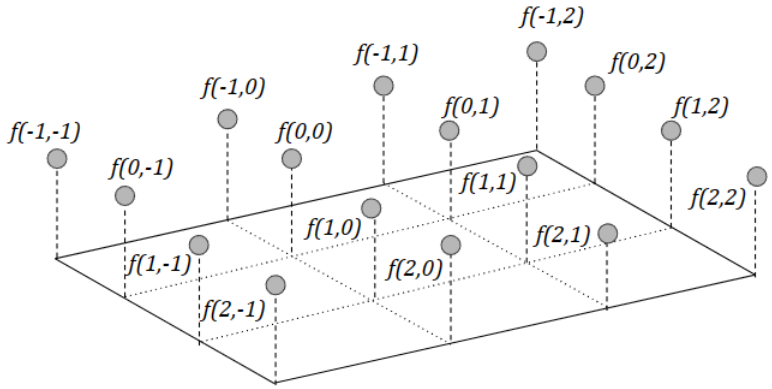


Figura 30: Esquema de pixel para filtro bicúbico.

Para determinar o valor do pixel $P(x,y)$ [27] é utlizada a fórmula

$$P(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j, \tag{25}$$

que pode ser lida na forma matricial como

$$P(x,y) = f(x,y) = \begin{bmatrix} x^3 & x^2 & x & 1 \end{bmatrix} \boldsymbol{A} \begin{bmatrix} y^3 \\ y^2 \\ y^1 \\ 1 \end{bmatrix}. \tag{26}$$

onde $\boldsymbol{A}$ identifica a matriz dos coeficientes a_{ij}, da forma

$$\boldsymbol{A} = \begin{bmatrix} a_{33} & a_{32} & a_{31} & a_{30} \\ a_{23} & a_{22} & a_{21} & a_{20} \\ a_{13} & a_{12} & a_{11} & a_{10} \\ a_{03} & a_{02} & a_{01} & a_{00} \end{bmatrix}.$$

Nessa formulação, os coeficientes a_{ij}, com $i,j = 0,1,2,3$, devem ser determinados. Para isso, utilizam-se então as informações dos 16 pixel da imagem original,

como mostrado na Figura 30. Assim, a Equação (26), em um sistema 4×4, pode ser lida como

$$\boldsymbol{F} = \boldsymbol{B}\boldsymbol{A}\boldsymbol{B}^T, \tag{27}$$

Nesse caso, $\boldsymbol{F}$ representa a matriz que contém valores $f(x,y)$, dada por

$$\boldsymbol{F} = \begin{bmatrix} f(-1,-1) & f(-1,0) & f(-1,1) & f(-1,2) \\ f(0,-1) & f(0,0) & f(0,1) & f(0,2) \\ f(1,-1) & f(1,0) & f(1,1) & f(1,2) \\ f(2,-1) & f(2,0) & f(2,1) & f(2,2) \end{bmatrix}.$$

Além disso, a matriz $\boldsymbol{B}$ representa todos os valores de x,y do retângulo pré-estabelecido, sendo dada por

$$\boldsymbol{B} = \begin{bmatrix} (-1)^3 & (-1)^2 & -1 & 1 \\ 0^3 & 0^2 & 0 & 1 \\ 1^3 & 1^2 & 1 & 1 \\ 2^3 & 2^2 & 2 & 1 \end{bmatrix}.$$

Com a equação (27), obtém-se

$$\boldsymbol{A} = \boldsymbol{B}^{-1}\boldsymbol{F}(\boldsymbol{B}^{-1})^T, \tag{28}$$

onde $(\cdot)^{-1}$ indica a operação inversa. Desse modo, encontram-se os valores para os coeficientes a_{ij} de $\boldsymbol{A}$ [27]. Em posse desses coeficientes, aplica-se a equação (25) para encontrar os novos pixels desejados.

Aplicando ambos os filtros de interpolação à mesma imagem, verifica-se a diferença na eficiência desses filtros no processo de *upscale*. Utilizando a métrica SSIM descrita anteriormente e comparando as imagens filtradas com a imagem original, é possível observar os resultados do processo de aumento na resolução de uma imagem de 150×150 pixels para 300×300 pixels, apresentados na Figura 31. Logo, é possível perceber que o filtro bicúbico apresenta um resultado ligeiramente superior ao filtro bilinear levando em consideração o SSIM.

(a) Bilinear, SSIM:0.831 (b) Bicúbico, SSIM: 0.846

Figura 31: Comparação entre *upscale* de filtros Bilinear e Bicúbico.

Referências

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Prentice Hall, NJ, 3rd edition, 2008.

[2] L. Yue et al., "Image super-resolution: The techniques, applications, and future," *Signal Processing*, vol. 128, pp. 389–408, 2016.

[3] Z. Wang, J. Chen, and S. H. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 43, no. 10, pp. 3365–3387, Oct. 2021.

[4] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

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

[6] 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*, pp. 2672–2680, 2014.

[7] D. Martin et al., "A database of human segmented natural images and its application to evaluating segmentation valgorithms and measuring ecological statistics", 8th Int. Conf. Computer Vision, p. 416–423, 2001.

[8] L. Yue *et al.*, "Image super-resolution: The techniques, applications, and future," *Signal Processing*, vol. 128, pp. 389–408, 2016.

[9] O'Mahony, Niall, et al. "Deep learning vs. traditional computer vision."Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1. Springer International Publishing, 2020.

[10] Banerjee, Chaity, Tathagata Mukherjee, and Eduardo Pasiliao Jr. "An empirical study on generalizations of the ReLU activation function."Proceedings of the 2019 ACM Southeast Conference. 2019.

[11] Shi, Wenzhe, et al. "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network."Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[12] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[13] Neto, Pedro Luiz de Oliveira Costa. Estatística. Editora Blucher, 2002.

[14] Nilsson, Jim, and Tomas Akenine-Möller. "Understanding ssim."arXiv preprint arXiv:2006.13846 (2020).

[15] Foster, David. Generative deep learning. "O'Reilly Media, Inc.", 2022.

[16] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255).

[17] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets."arXiv preprint arXiv:1411.1784 (2014).

[18] S. Haykin, "Neural Networks and Learning Machines", 3/E. Pearson Education, 2009.

[19] C. M. Bishop, *Pattern recognition and machine learning*, Springer, 2006.

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

[21] A. Y. Ng, K. Katanforoosh, and Y. B. Mourri, "Neural networks and deep learning," deeplearning.ai, disponível em: https://www.coursera.org/learn/neural-networks-deep-learning. Acesso em: 21 jan. 2020.

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

[23] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[24] Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, Brian P., *Numerical recipes in C: the art of scientific computing*. New York, NY, USA: Cambridge University Press, 2nd ed., 1992

[25] Han, Dianyuan. "Comparison of commonly used image interpolation methods,"in Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013). Atlantis Press, 2013.

[26] R. Keys, "Cubic convolution interpolation for digital image processing", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 29, no. 6, pp. 1153-1160, December 1981, doi: 10.1109/TASSP.1981.1163711.

[27] Xiao Shu, "Bicubic Interpolation", McMaster University, Canada, https://www.ece.mcmaster.ca/ xwu/3sk3/interpolation.pdf, 2013.

[28] Maas, Andrew L, Hannun, Awni Y, and Ng, Andrew Y. Rectifier nonlinearities improve neural network acoustic models. In ICML, volume 30, 2013

[29] Arjovsky, M. (2017, January 26). Wasserstein GAN.

[30] Johnson, J. (2016, March 27). Perceptual Losses for Real-Time Style Transfer and Super-Resolution.