Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica
Projeto de Engenharia Elétrica

Trabalho de Conclusão de Curso
Colorização Automática de Imagens em Escala de Cinza Utilizando Deep Learning

Leo de Lima Araújo

Campina Grande - PB
Julho de 2019

Leo de Lima Araújo

Trabalho de Conclusão de Curso
Colorização Automática de Imagens em Escala de Cinza Utilizando Deep Learning

Trabalho de Conclusão de Curso submetido à Coordenação de Curso de Graduação de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Área de Concentração: Processamento de Imagens.

Orientadora: Prof. Luciana Ribeiro Veloso, Dr. Sc

Campina Grande - PB

Julho de 2019

Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Engenharia Elétrica - DEE
Projeto de Engenharia Elétrica

Leo de Lima Araújo

Trabalho de Conclusão de Curso
Colorização Automática de Imagens em Escala de Cinza Utilizando Deep Learning

Trabalho aprovado em: Campina Grande - PB, / /

Leo de Lima Araújo
Aluno

Luciana Ribeiro Veloso, Dr. Sc
Professora Orientadora

Campina Grande - PB

Julho de 2019

Resumo

Redes Neurais Convolucionais e outros métodos de *Deep Learning* vêm se tornando cada vez mais relevantes nos últimos anos e têm superado as técnicas tradicionais em diversos aspectos de domínios como Visão Computacional e Processamento de Sinais. Nesse contexto, propõe-se a realização de um estudo sobre tais métodos de modo a determinar a viabilidade de sua implementação para realizar a colorização automática de imagens em escala de cinza, algo que vem sendo alvo de pesquisas de Visão Computacional desde a década de 1980, determinando, dentre as execuções bem sucedidas realizadas em outros trabalhos, a melhor forma de fazê-lo. Foram comparados três modelos de Redes Neurais Convolucionais treinados para a realização da colorização automática e eles foram avaliados segundo três índices qualitativos, a raíz do erro médio quadrático (RMSE), a relação sinal ruído de pico (PSNR) e o índice de similaridade estrutural (SSIM), além de um teste de usabilidade, permitindo que se determinasse os diferenciais de cada modelo e seus problemas. Concluiu-se que, além de viável, é possível fazê-lo com resultados bastante realistas, capazes de convencer outras pessoas da autenticidade das colorizações produzidas.

Palavras-chave: *Deep Learning*, Redes Neurais Convolucionais, Colorização Automáticos.

Abstract

Convolutional Neural Networks and other Deep Learning techniques have become more relevant outperforming numerous classic applications in fields like Computer Vision and Signal Processing. Therefore, this project proposes a study covering those techniques in order to ascertain the viability of implementing them in order to colorize grayscale images automatically, something that has been studied by Computer Vision experts since the decade of 1980, and determine the most efficient way of doing it. Three Convolutional Neural Network models, trained to perform automatic colorization, were evaluated according utilizing the root mean square error (RMSE), the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) together with an user study, thus allowing the strengths and weaknesses of each model to be determined. It was concluded that it is not only possible to implement automatic colorization utilizing Deep Learning, but results that are good enough to convince other people of the colorization's authenticity can be achieved.

Keywords: Deep Learning, Convolutional Neural Networks, Automatic Colorization.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AE	Absolut Error
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GPU	Graphics Processing Unit
LReLU	Leaky Rectified Linear Unit
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RMS	Root Mean Square
RMSE	Root Mean Square Error
RNA	Rede Neural Artificial
SE	Squared Error
SSIM	Structural Similarity
VGG	Visual Geometry Group

Sumário

1 Introdução

Colorir imagens automaticamente tem sido uma meta das pesquisas de Processamento de Imagem e de Visão computacional desde a década de 1980, quando estúdios cinematográficos começaram a lançar versões coloridas dos filmes em preto e branco. Como colorir cada *frame* de um filme é um trabalho caro e tedioso, sistemas semi-automatizados rapidamente surgiram e, com o passar do tempo, esses sistemas passaram a requisitar cada vez menos interação humana, (ROYER; KOLESNIKOV; LAMPERT, 2017).

O problema apresenta certa complexidade devido à perda de informação observada ao converter uma imagem colorida em uma imagem em escala de cinza. Dois terços da informação é perdida no processo, (ZHANG; ISOLA; EFROS, 2016), e, por conta disso, não é possível determinar algebricamente a informação perdida a partir do canal restante. Todavia, ao observar uma imagem em preto e branco, é possível, para um humano, inferir qual seria a provável cor de alguns de seus elementos, como um morango, a grama, ou o céu.

Estes fatores indicam a existência, dentro da própria imagem, de informações capazes de revelar sua matriz de cor. Para poder explorar essas informações de modo a produzir colorizações realistas, um sistema deve ser capaz de interpretar a imagem e identificar os objetos que compõem a cena e sua localização, (LARSSON; MAIRE; SHAKHNAROVICH, 2016). Neste contexto, técnicas tradicionais de Processamento de Imagem e Visão Computacional não são adequadas devido à abstração do problema.

Figura 1 – Exemplos de imagens com colorizações bem sucedidas.

Fonte: Zhang, Isola e Efros (2016).

Técnicas de *Deep Learning* têm mostrado desempenho formidável na realização de tarefas com alto nível de abstração em áreas como Visão Computacional (Sun, Wu e Hoi (2018), Voulodimos et al. (2018)), Reconhecimento e Síntese de Voz (Wang, Takaki e Yamagishi (2018), Hannun et al. (2014)) e Tradução Automática (Bahdanau, Cho e Bengio (2014), Junczys-Dowmunt et al. (2018)). Modelos como Redes Neurais Artificiais (RNAs) e Redes Neurais Convolucionais (CNNs) mapeiam entradas brutas nas saídas desejadas e, diferentemente das técnicas tradicionais de aprendizado de máquina, que requerem a implementação de um extrator de características a partir dos dados de entrada, métodos de Deep learning podem aprender a realizar essa extração diretamente a partir dos dados, (CHARTRAND et al., 2017). Sua capacidade de abstrair padrões a partir de seu treinamento, sem a necessidade de explicitamente descrever como ela deve implementar a função é precisamente o que faz das redes neurais uma ótima ferramenta para este problema.

Nesse contexto, é de se esperar que, dispondo de um grande banco de imagens para realizar o treinamento da rede, ela seja capaz de aprender a extrair informações relevantes das imagens de entrada e as utilize para realizar a colorização. De fato, diversos modelos de colorizadores automáticos já foram implementados utilizando *Deep Learning*, a exemplo de Cheng, Yang e Sheng (2015), Iizuka, Simo-Serra e Ishikawa (2016), Deshpande et al. (2017), Cao et al. (2017), e outros. Os exemplos de colorizações bem sucedidas da Figura 1 mostram que os modelos existentes já conseguem realizar colorizações bastante verossímeis.

Todavia, nem todos os desafios nessa área foram transpostos, uma vez que os colorizadores já implementados apresentam limitações, como pode ser visto na Figura 2. Sendo assim, torna-se necessário estudar as arquiteturas existentes e promover formas de avaliar seu sucesso, de modo que os fatores limitantes possam ser identificados e novas arquiteturas mais robustas possam ser projetadas.

Figura 2 – Exemplos de imagens com erros de colorização.

Fonte: Larsson, Maire e Shakhnarovich (2016).

Nesse contexto, este trabalho tem por objetivo realizar um estudo sobre Deep Learning e realizar a comparação entre modelos de Redes Neurais Convolucionais utilizados para colorir imagens em escala de cinza de forma automática. Observa-se que, devido à complexidade do problema, não se espera que os resultados desses modelos correspondam exatamente às cores reais da imagem, mas apenas que o padrão de cores determinado pelas redes seja capaz de convencer seres humanos da autenticidade das imagens produzidas.

Uma aplicação interessante para esse tipo de rede neural é mostrado na Figura 3, onde são mostradas colorizações realizadas por Iizuka, Simo-Serra e Ishikawa (2016) de imagens captadas utilizando câmeras antigas. Colorizações semelhantes são feitas por Zhang, Isola e Efros (2016), que realizam, inclusive, a colorização de uma foto de um lobo da tasmânia, animal extinto em 1936.

O restante deste trabalho será organizado conforme o modelo descrito a seguir. O Capítulo 2 realizará uma fundamentação teórica abordando desde os conhecimentos básicos de Deep Learning até conceitos mais complexos como Redes Neurais Profundas e Redes Neurais Convolucionais. Em seguida, no Capítulo 3, será realizado um levantamento dos trabalhos acadêmicos mais relevantes na área de colorização automática com ênfase nos modelos a serem comparados. O Capítulo 4 tratará da metodologia adotada e será seguido pelos resultados obtidos, no Capítulo 5, e as considerações finais, no Capítulo 6.

Figura 3 – Exemplos de colorizações de imagens históricas com mais de 100 anos.

Fonte: Iizuka, Simo-Serra e Ishikawa (2016).

2 Fundamentação Teórica

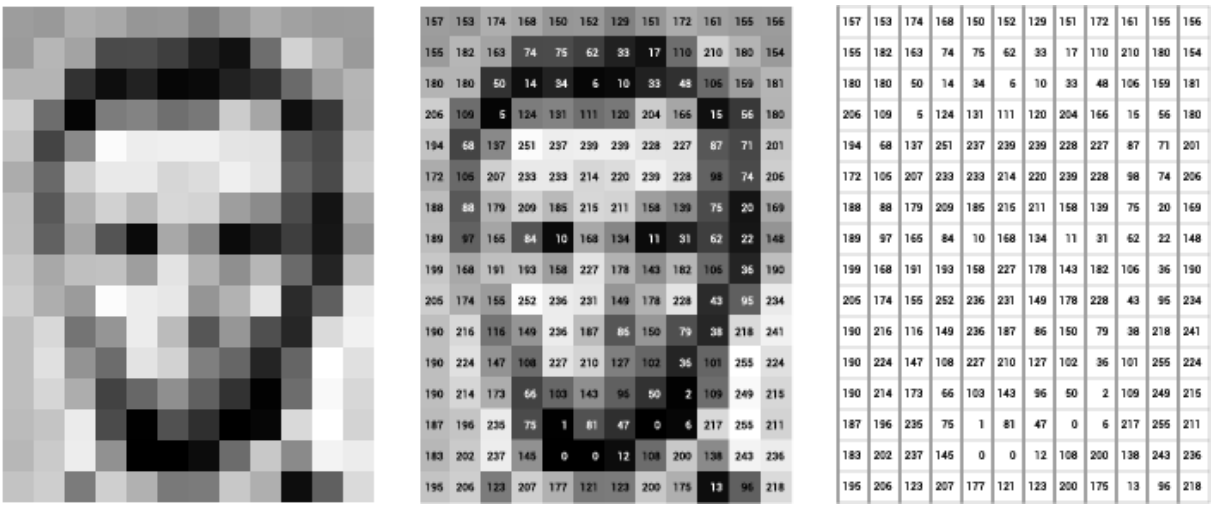
2.1 Espaços de Cores

Um espaço de cores é um modelo matemático capaz de descrever cores a partir de conjuntos de, em geral três ou quatro, números, (STOKES et al., 1996), permitindo que as cores sejam mensuradas e especificadas quantitativamente, (TOET, 1992). Enquanto que para imagens coloridas mais canais podem ser necessários, imagens em escala de cinza só precisam de um único canal, que indica a intensidade de cada *pixel*.

Como pode ser visto na Figura 4, em uma imagem em escala de cinza de dimensões $H \times W$ cada um de seus *pixels* é representado por um único valor, usualmente um inteiro entre 0 e 255. Nessas imagens, a cor preta é comumente representada pelo valor 0 e a intensidade vai gradativamente se tornando mais clara à medida que o valor aumenta, se tornando branca ao atingir o valor máximo, 255. Em fotografias em escala de cinza a intensidade dos pixels corresponde à intensidade da luz mensurada pela câmera nos pixels em questão, mas em aplicações distintas a sua interpretação física pode variar.

Os diferentes espaços de cores existentes servem para descrever esses conjuntos de valores, fornecendo uma interpretação para cada um deles. Alguns exemplos são os espaços RGB, empregado na maioria dos monitores, e L*a*b*, que apresenta propriedades interessante para a realização de colorização automática.

Figura 4 – Imagem em escala de cinza, à esquerda, representada como matriz, à direita.



Fonte: `https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html`.

2.1.1 RGB

O sistema RGB utiliza misturas de diferentes quantidades de vermelho, verde e azul para produzir diferentes tonalidades de cores, sendo a sigla derivada das iniciais das cores utilizadas, em inglês. Em uma imagem digital colorida no sistema RGB, um pixel pode ser entendido como um vetor cujas componentes representam as intensidades de vermelho, verde e azul de sua cor.

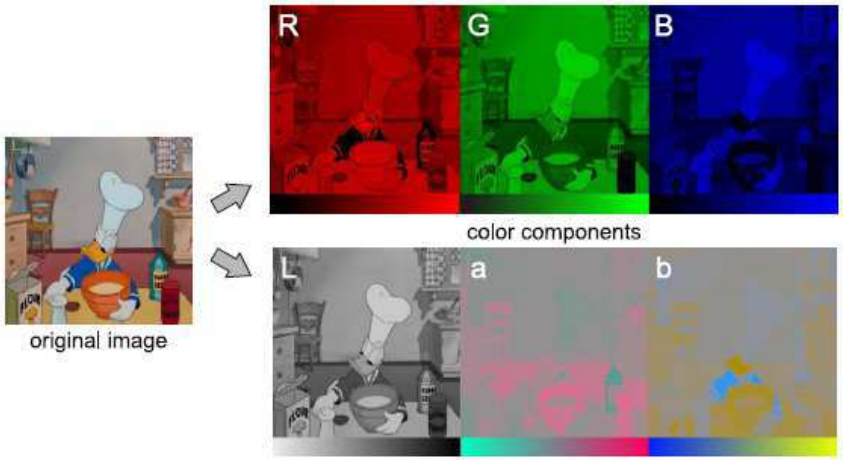
A mistura de cores no espaço RGB pode ser melhor compreendida ao pensar na interação entre luzes monocromáticas das cores referidas, uma vez que as luzes se misturam de maneira similar, o que torna o espaço RGB apropriado para o uso em dispositivos emissores de luz, como monitores e televisores. A Figura 5 mostra, isoladamente, os canais RGB de uma imagem colorida.

2.1.2 Espaço L*a*b*

O espaço de cores L*a*b* é um dos espaços mais utilizados em aplicações de colorização automática porque o canal L se refere à própria intensidade luminosa, que varia de 0 (preto) a 100 (branco), e corresponde a uma versão escalonada da imagem em escala de cinza. Consequentemente, o dado de entrada já corresponde a um terço do resultado final, sendo necessário apenas descobrir os dois outros canais.

Os canais a e b formam um plano que descreve as possíveis cores da imagem e ambos variam de -128 a 127, sendo o ponto (0,0) referente à cor cinza. Variando a na direção negativa, obtém-se uma coloração mais verde e, na direção positiva, vermelha. Para b a coloração se torna mais azul na direção negativa, e amarela, na positiva. A Figura 5 mostra individualmente os canais L*a*b* formando uma imagem colorida, enquanto a

Figura 5 – Representação dos espaços RGB e L*a*b*.



Fonte: (CHYBICKI et al., 2019).

Figura 6 ilustra as cores formadas para diferentes valores de a, b e L.

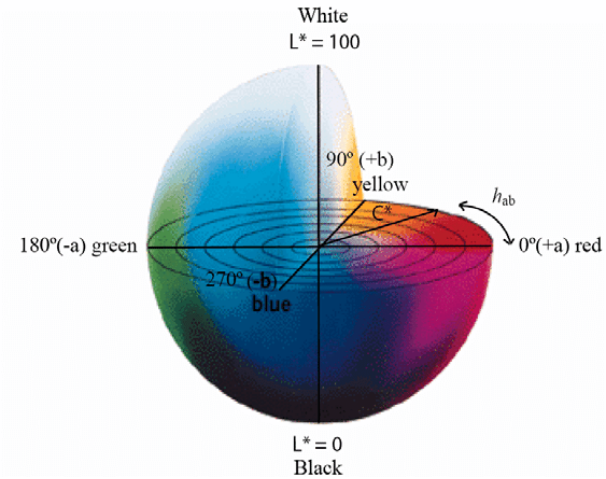
2.1.3 Espaço HCL

Outro espaço de interesse para aplicações de colorização automática é o HCL (tonalidade (*hue*), saturação (*chroma*), intensidade luminosa (*lightness*)). A saturação fornece uma indicação da pureza da cor, de modo que cores com baixa saturação têm tonalidades mais acinzentadas, e a tonalidade indica a matiz da cor, ou seja, sua coloração em seu estado puro.

Nesse espaço, descrito em coordenadas cilíndricas, a tonalidade é representada por um ângulo entre 0 e 360 graus e a saturação representa a distância entre o ponto em questão e o eixo de intensidade luminosa, que é a mesma do espaço L*a*b*. Como pode ser visto na Figura 6, o espaço HCL é simplesmente uma versão em coordenadas cilíndridas do espaço L*a*b*.

Embora tanto o espaço HCL quanto o espaço L*a*b* representarem as mesmas cores a partir de diferentes sistemas de coordenadas, as distinções no significado de seus parâmetros têm importância. Pode ser que, usando um determinado conjunto de treino, seja mais fácil para a rede neural identificar padrões relacionados a um sistema de coordenadas do que outro, permitindo, assim, que melhores resultados sejam alcançados.

Figura 6 – Representação do Espaço de Cores CIE L*a*b* e HCL.



Fonte: https://www.researchgate.net/figure/The-three-dimensional-CIELab-colour-space-showing-h-ab-hue-angel-C-chroma-and-L_fig4_267180218.

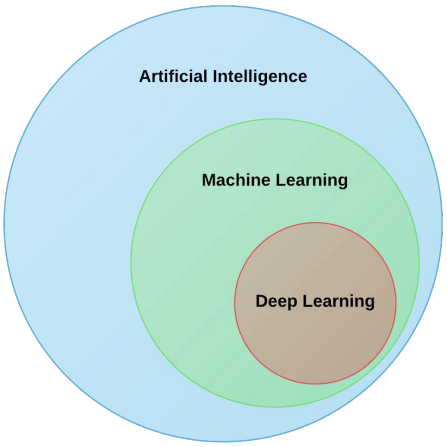
2.2 Deep Learning

Aprendizado de máquina (*Machine Learning*) é o campo de estudo que proporciona às máquinas a habilidade de aprender sem que elas sejam explicitamente programadas, (SAMUEL, 1959). Alguns dos métodos inclusos neste conjunto são *K Nearest Neighbors Classifier*, *Suport Vector Machine*, *Decision Tree Classifier* e *Random Forest Classifier*. Como pode ser visto na Figura 7, as técnicas de aprendizado de máquina são um dos métodos de Inteligência Artificial.

As técnicas de aprendizado profundo (*Deep Learning*) são um subcampo de aprendizado de máquina, onde o aprendizado se dá a partir de dados por meio de sucessivas camadas de níveis crescentes de abstração, (FRANCOIS, 2017). O uso de múltiplas camadas permite o aprendizado de funções que muito dificilmente poderiam ser descritas por programação explícita. Neste conjunto estão inclusas redes neurais artificiais, redes neurais convolucionais, redes neurais recorrentes, dentre outras arquiteturas.

Comumente, modelos de *Deep Learning* fazem uso de um tipo específico de aprendizado de máquina denominado aprendizado supervisionado. Neste tipo de aprendizado os dados de treinamento que são repassados ao algoritmo incluem as saídas desejadas, denominadas rótulos, (GÉRON, 2017). Utilizando esses dados, redes neurais e redes convolucionais conseguem regular seus parâmetros de modo a produzir as saídas desejadas a partir das entradas fornecidas.

Figura 7 – Hierarquia entre Deep Learning, Machine Learning e Inteligência artificial.



Fonte: `https://steemit.com/ai/@quantum-cyborg/what-is-deep-learning-how-does-it-differ-from-most-of-ai`

2.3 Redes Neurais Artificiais

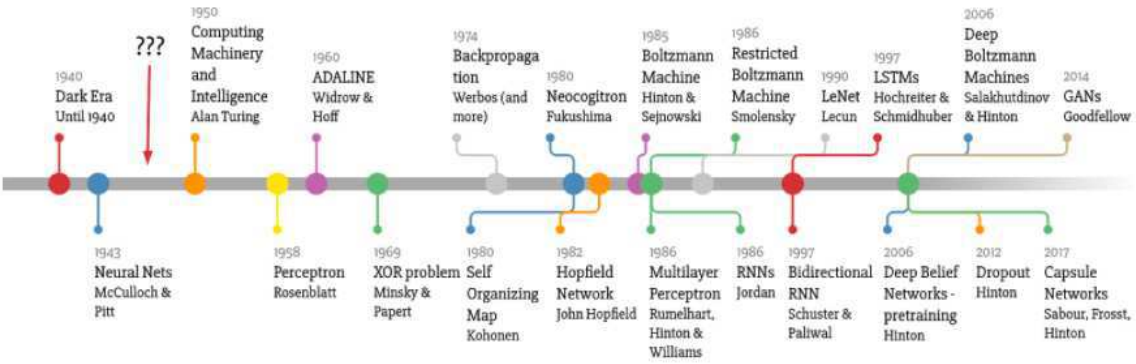
Redes Neurais Artificiais (RNA) podem ser definidas como um conjunto de unidades básicas de processamento, altamente conectadas, que processam a informação contida nas entradas fornecidas e produzem as saídas desejadas, (KHAN et al., 2018). As redes devem ser treinados a partir de grandes bancos de dados de modo a aprimorar a qualidade das saídas, podendo mapear de forma coerente funções não lineares de alta complexidade.

Como pode ser visto na linha do tempo da Figura 8, os primeiros modelos matemáticos de redes neurais datam da década de 50, com McCulloch e Pitts (1943). Naturalmente eles eram bastante limitados, mas serviram de inspiração para que, quase duas décadas depois, modelos mais complexos surgissem, como os de Widrow e Hoff (1960) e Rosenblatt (1961). Em 1969 foi publicado um estudo, (MINSKY; PAPERT, 1969), que mostrava que os modelos da época eram incapazes de resolver problemas que não fossem linearmente separáveis, como a função lógica **XOR**, levando a comunidade científica a perder interesse nas redes neurais durante anos.

Na década de 80 foi proposto o método de *backpropagation*, que permitiu que os modelos utilizassem mais camadas e, assim, superassem limitações como o problema **XOR**. Novas pesquisas continuaram abordando RNAs, porém a falta de recursos computacionais e dados disponíveis para a realização dos treinamentos comprometia a performance das redes mais profundas. Isso fez com que nos anos 90 a comunidade científica novamente perdesse a atenção nas redes neurais e se voltasse para outros métodos de aprendizado de máquina mais populares, como *Suport Vector Machine*.

Foi apenas nos anos 2000 que as Redes Neurais voltaram a receber atenção, algo que se deve aos avanços observados no poder computacional dos computadores modernos e da implementação de GPUs, além do grande acúmulo de dados observado na atualidade,

Figura 8 – Linha do Tempo do Deep Learning.



Fonte: `https://www.bbvadata.com/a-weird-introduction-to-deep-learning/`

que proporciona grandes *datasets* para a realização do treinamento das redes. A união desses fatores proporcionou o ambiente perfeito para a criação de redes cada vez mais profundas e bancos de dados cada vez maiores.

2.3.1 Modelo

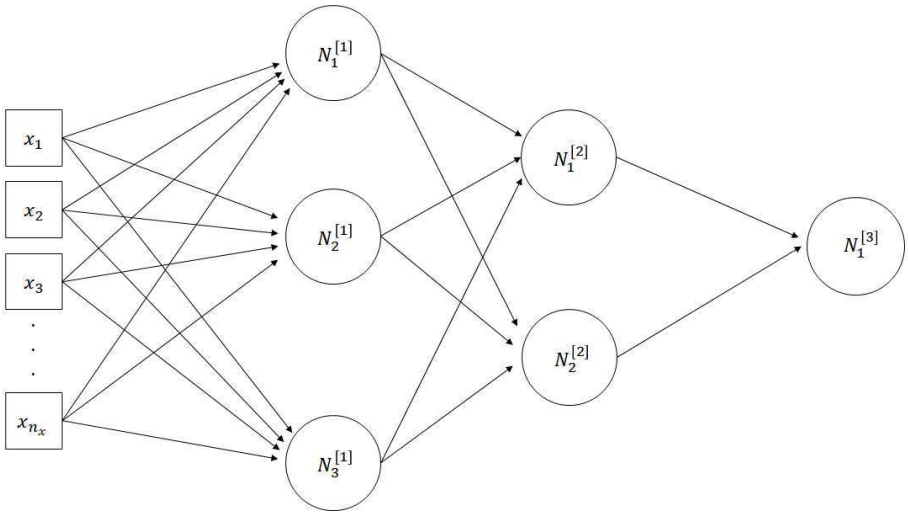
Um modelo de uma rede neural pode ser visto na Figura 9. Olhando da esquerda para a direita, a primeira camada é a camada de entrada, a última é a camada de saída e as demais são denominadas camadas ocultas. Comumente não se considera a camada de entrada quando se conta o número de camadas da rede, de modo que esta seria uma RNA de três camadas e, o que caracteriza uma rede neural "rasa", visto que atualmente já são concebidos modelos com mais de cem camadas.

As camadas ocultas têm esse nome porque não se tem controle sobre como elas processam a informação recebida. Seu conteúdo raramente é de interesse de quem utiliza a rede neural, elas representam apenas passos intermediários entre a entrada e a saída do sistema. Desta forma, na maioria das aplicações a rede neural é vista apenas como uma caixa preta que recebe um vetor de entradas X e produz um vetor de saídas Y.

Cada camada é composta de um determinado número de neurônios, representados pelos círculos na Figura 9, que são as unidades básicas do processamento da rede. Os neurônios de cada camada se conectam a todos os elementos da camada anterior de forma que a rede se torna densamente conectada. O número da camada de um determinado neurônio é determinado pelo expoente entre colchetes.

Todo neurônio interage com os elementos da camada anterior a partir de um conjunto de pesos, w, e um viés, b. A partir deles, se calcula um número real z, pela

Figura 9 – Modelo de rede neural utilizando duas camadas ocultas.



Fonte: Próprio autor.

Equação 2.1, onde x representa as entradas do neurônio em questão, sendo a própria entrada da rede para as unidades da primeira camada e, para as demais camadas, é o conjunto de saídas, a, da camada anterior.

$$z = w^T.x + b \tag{2.1}$$

$$a = g(z) \tag{2.2}$$

Em seguida é aplicada uma função de ativação $g(z)$, não linear, que produz a saída do neurônio. Essa função insere não linearidades no sistema, fazendo com que a RNA não fique restrita ao mapeamento de apenas funções lineares. Nota-se que funções de ativação distintas podem ser escolhidas para cada camada.

2.3.2 Atualização dos Parâmetros

Os pesos e o viés de cada neurônio são atualizados segundo um algoritmo denominado descida do gradiente. Para tal, é necessário definir duas funções importantes:

1. $L(y, \hat{y})$ - Função de Perdas: Mensura para o i-ésimo exemplo do conjunto o quão diferente do rótulo $y^{(i)}$ está a previsão $\hat{y}^{(i)}$ fornecida pela rede. A escolha dessa função é fundamental para que a rede seja capaz de efetivamente generalizar a função a ser mapeada e atualizar seus parâmetros. Mais informações sobre essa função podem ser encontrados em (FRANCOIS, 2017).

2. $J(w, b)$ - Função de Custo: Enquanto $L(y, \hat{y})$ mensura o desempenho da rede em um único exemplo, $J(w, b)$ o faz para todo um conjunto de dados. Minimizar o valor desta função para o conjunto de treino é, consequentemente, o objetivo do treinamento da rede.

A forma mais comum de implementar a função de custo é calcular a média da função de perdas para todos os m exemplos do conjunto, conforme a Equação 2.3.

$$J(w, b) = \frac{1}{m}.\sum_{i=1}^{m} L(y^{(i)}, \hat{y}^{(i)}) \tag{2.3}$$

Para implementar a descida do gradiente deve-se então calcular o gradiente de $J(w, b)$ com relação aos parâmetros w e b, de modo que a cada iteração do processo de treinamento eles serão atualizados segundo as equações 2.4 e 2.5. Os valores iniciais das variáveis usualmente são inicializados de forma aleatória.

O termo α nas Equações 2.4 e 2.5 é denominado de taxa de aprendizagem e define a influência de cada iteração sobre os valores atuais dos parâmetros. Podendo fazer com

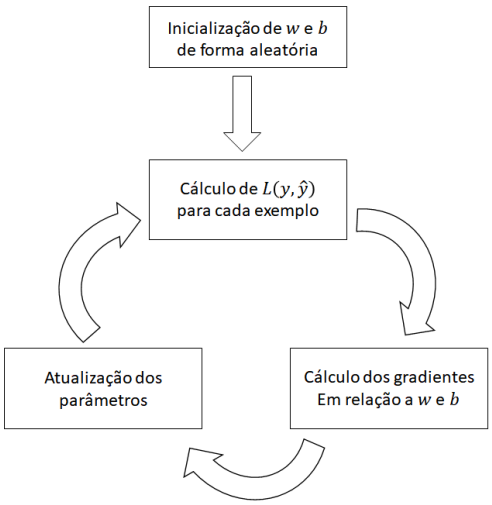
que a cada iteração os parâmetros sofram grandes alterações, se α for alto, ou pequenas caso contrário. Nas equações, o termo k indica a iteração atual.

$$w_i(k+1) = w_i(k) - \alpha.\frac{\partial J(k)}{\partial w_i} \tag{2.4}$$

$$b(k+1) = b(k) - \alpha.\frac{\partial J(k)}{\partial b} \tag{2.5}$$

Para redes com mais de uma camada os gradientes de uma camada dependem dos gradientes da camada seguinte. Essa propagação no sentido inverso (da última camada para a primeira) é denominada retropropagação (*backpropagation*, em inglês) e pode ser vista mais detalhadamente em (FRANCOIS, 2017) e (ROSEBROCK, 2017). O treinamento segue o fluxograma da Figura 10 até que se atinja um critério de parada.

Figura 10 – Fluxograma do processo de treinamento.



Fonte: Próprio autor.

2.3.3 Hiperparâmetros

Em *Deep Learning* existem dois tipos de parâmetros, os que são treináveis e os que são definidos pelo projetista. Os primeiros correspondem aos pesos e viéses dos neurônios, enquanto os outros são características da rede como a taxa de aprendizagem, o número de camadas, o número de neurônios em cada camada, dentre outros.

Boas escolhas para os hiperparâmetros proporcionam melhores soluções e um treinamento mais veloz. Infelizmente, ainda não existe uma forma de determinar exatamente os melhores valores para cada hiperparâmetro, de modo que, para atingir uma performance ótima a rede deve ser treinada várias vezes até que se encontre um conjunto ótimo.

2.4 Funções de Perdas

Uma consideração importante antes de decidir que tipo de função de perdas utilizar é o tipo de problema em questão, geralmente divididos em problemas de classificação ou de regressão. Em problemas de regressão, o objetivo da rede é estimar uma quantidade a partir da entrada, enquanto nos problemas de classificação, o objetivo é identificar, a qual classe pertence uma dada entrada, (BROWNLEE, 2019). Os valores de saída para problemas de regressão são valores tipicamente reais, enquanto nos de classficação a saída normalmente é um número inteiro associado a uma classe.

2.4.1 Regressão

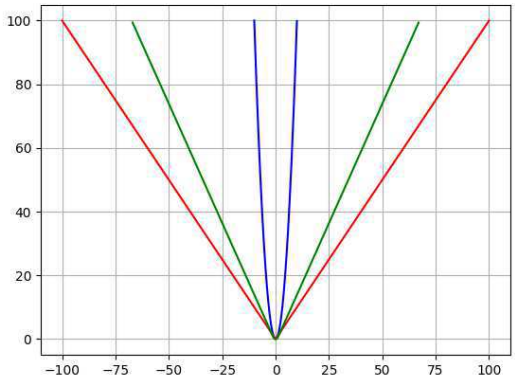
Como nesse tipo de problema a saída prevista, $\hat{y}$, é uma quantidade, a função de perdas, que calcula a distância entre o valor real e o valor previsto, se torna uma função de erro. Nesse sentido, funções de como o erro quadrático (SE) ou o erro absoluto (AE), são alternativas comuns, uma outra opção é a função de Huber, Equação 2.8, que é uma mistura das duas primeiras. A Figura 11 ilustra as funções descritas, sendo a curva azul referente ao SE, a vermelha ao AE e a verde à função de Huber.

$$L(y, \hat{y}) = SE = (y - \hat{y})^2 \tag{2.6}$$

$$L(y, \hat{y}) = AE = |y - \hat{y}| \tag{2.7}$$

$$L(y, \hat{y}) = L_\delta = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & , |y - \hat{y}| \leq \delta \\ \delta.|y - \hat{y}| - \frac{1}{2}.\delta & , |y - \hat{y}| \geq \delta \end{cases} \tag{2.8}$$

Figura 11 – Funções de perdas para problemas de regressão.



Fonte: `https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e`

2.4.2 Classificação

Para problemas de classificação binária, ou seja, quando só existem duas classes, a saída da rede, $\hat{y}$, será um número entre 0 e 1 que pode ser interpretado como a probabilidade de que a entrada pertença à primeira classe. Nesse caso, existem dois valores possíveis para a saída desejada, y, sendo 1 caso a entrada realmente pertença à primeira classe e 0 caso ela não pertença.

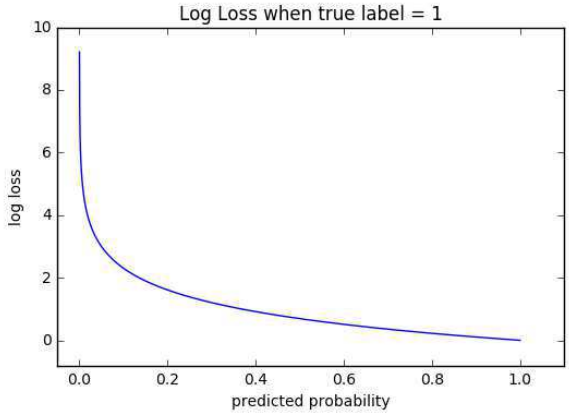
Consequentemente, espera-se que a função de perdas produza valores elevados se para $y = 1$ a saída $\hat{y}$ tende a 0 ou se para $y = 0$ a saída $\hat{y}$ tende a 1 e valores baixos para o contrário. Isso é alcançado pela função de entropia cruzada (Equação 2.9), também conhecida como função de perdas logarítmica (*Log Loss*), cujo gráfico pode ser visto na Figura 12 para $y = 1$.

$$L(y, \hat{y}) = -(y.log\hat{y} + (1 - y).log(1 - \hat{y})) \tag{2.9}$$

A Figura 12 ilustra um crescimento exponencial do valor de $L(y, \hat{y})$ à medida que a probabilidade prevista se afasta do valor da saída desejada, 1. Um comportamento semelhante ocorre caso $y = 0$, no entanto, a função teria um mínimo em $\hat{y} = 0$ e cresceria à medida que o valor estimado se aproximasse da unidade. Devido ao fato de tanto y quanto $\hat{y}$ serem valores entre 0 e 1, a função só está definida para este intervalo.

Cabe ainda uma observação, todo problema de classificação pode ser reduzido a uma série de problemas de classificação binária. Tudo que deve ser feito é avaliar o caso para cada classe, de modo que para cada uma delas será avaliado a probabilidade de que a entrada pertença ou não à classe, permitindo a aplicação da função a cada uma das saídas.

Figura 12 – Função de perdas de entropia cruzada.



Fonte: `https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html`.

2.5 Funções de Ativação

Sigmóide

A função sigmóide (Equação 2.10) ilustrada na Figura 13 em conjunto com sua derivada, foi uma das primeiras escolhas dos pesquisadores para a função de ativação dos neurônios. Uma de suas qualidades notáveis é o fato de ela ser derivável em toda a sua extensão, algo conveniente durante a descida do gradiente, pois simplifica a atualização dos parâmetros por meio da aplicação das Equações 2.4 e 2.5.

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.10}$$

$$g'(z) = \sigma(z).(1 - \sigma(z)) \tag{2.11}$$

Outra característica é o fato do resultado dessa função estar sempre contido entre 0 e 1. Isso a torna interessante para problemas de classificação binária, onde a rede neural busca classificar uma determinada informação de entrada como pertencente a uma de duas classes, neste caso a saída do neurônio pode ser interpretada como a probabilidade de que a informação pertença à primeira das classes.

Porém, o fato de a derivada dessa função rapidamente se aproximar de 0 conforme z se afasta da origem configura uma desvantagem. Como o aprendizado da rede depende

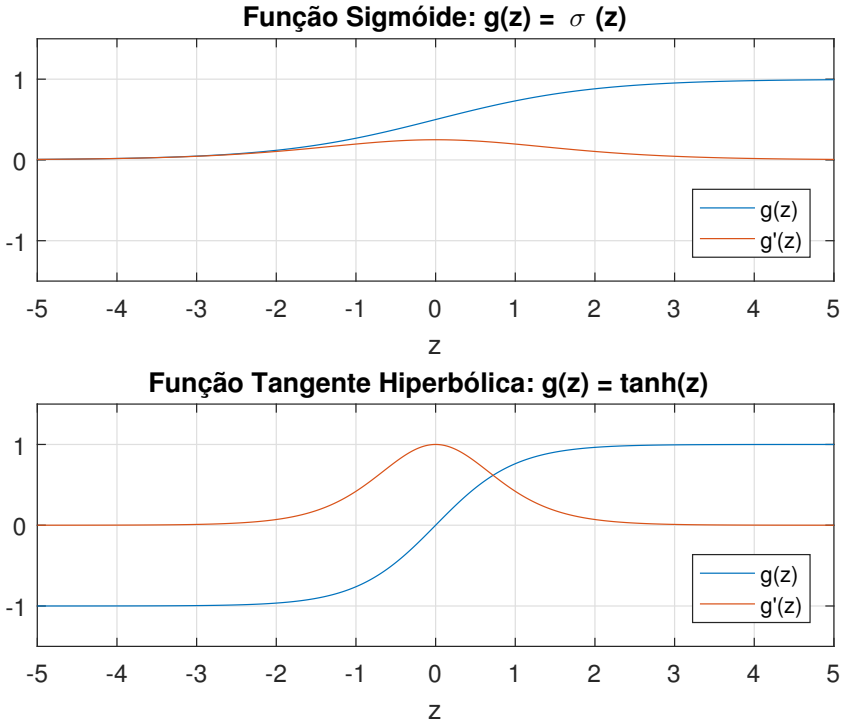


Figura 13 – Gráficos de $\sigma(z)$, $tanh(z)$ e suas derivadas.

do gradiente da função, derivadas proximas de 0 implicam em um aprendizado lento.

Tangente Hiperbólica

A função tangente hiperbólica (Equação 2.12) é um tanto similar à sigmóide, como pode ser visto na Figura 13. No entanto, difere no sentido de que seus resultados variam entre -1 e 1, o que é vantajoso porque a possibilidade de obter resultados negativos faz com que a média das saídas do neurônio seja mais próxima de 0, o que acelera o treinamento da rede neural conforme será visto mais a diante.

$$g(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.12}$$

$$g'(z) = 1 - tanh^2(z) \tag{2.13}$$

Percebe-se também que esta função apresenta uma derivada mais expressiva que $\sigma(z)$, mesmo embora isso se concentre nos entornos de $z = 0$. Assim como a sigmóide, seu gradiente tende a 0 conforme $|z|$ aumenta, o que desacelera a velocidade do aprendizado.

2.5.1 ReLU

A função ReLU (Equação 2.14), cuja sigla significa Unidade Linear Retificadora (*Rectified Linear Unit*, em inglês), foi proposta para lidar com o problema descrito acima. Como pode ser visto na Figura 14, a derivada da função se mantém unitário para $z > 0$ e zero para $z < 0$, de modo que a ocorrência de valores altos não desacelera o aprendizado da RNA.

$$g(z) = ReLU(z) = max(0, z) \tag{2.14}$$

$$g'(z) = \begin{cases} 1 & \text{se } z \geq 0 \\ 0 & \text{caso contrário.} \end{cases} \tag{2.15}$$

Também deve se considerar que os cálculos proporcionados pela função (Equação 2.14) são consideravelmente simples. Como o treinamento da rede neural envolve múltiplas execuções destas operações, o uso de funções simples também implica em uma redução no tempo necessário para treinar a rede. Esses dois motivos fazem com que essa função seja bastante popular,

Todavia, como pode ser visto na Figura 14, a derivada da função apresenta uma descontinuidade em $z = 0$. Para contornar o problema, em geral se considera $g'(0) = 1$, o que não é matematicamente correto, mas não prejudica a execução do programa porque

a probabilidade de que z seja exatamente igual a 0 é efetivamente nula, além de que, caso isso de fato ocorra, implicaria em um pequeno desvio do gradiente para a direção errada, algo que seria corrigido nas iterações seguintes.

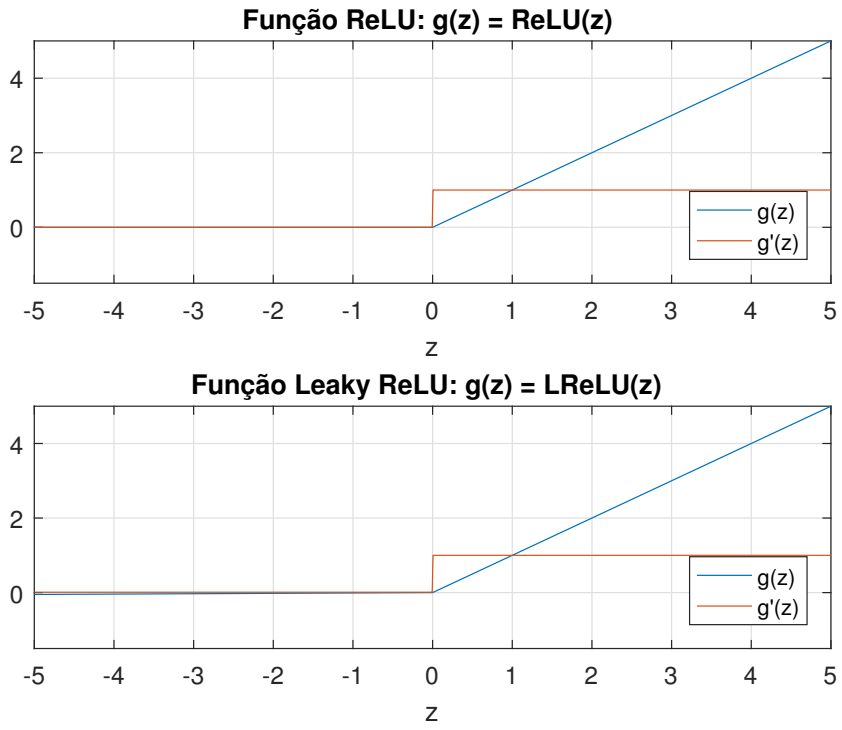
2.5.2 *Leaky* ReLU

Embora a função ReLU não apresente um gradiente nulo para $|z| >> 0$, isso ocorre para todo $z < 0$, o que também pode desacelerar o aprendizado. Isso motivou o surgimento de uma versão modificada, a *Leaky* ReLU ($g(z) = LReLU(z)$), que conforme a Equação 2.16, possui derivada 0,01, e não zero, para $z < 0$. Isso faz com que o aprendizado ocorra mesmo quando z é negativo. A LReLU está plotada na Figura 14 em conjunto com sua derivada.

$$g(z) = LReLU(z) = max(0,01.z, z) \tag{2.16}$$

$$g'(z) = \begin{cases} 1 & \text{se } z \geq 0 \\ 0,01 & \text{caso contrário.} \end{cases} \tag{2.17}$$

Figura 14 – Gráficos de $ReLU(z)$, $LReLU(z)$ e suas derivadas.



Fonte: Próprio autor.

2.5.3 Função *Softmax*

A função *softmax* é uma função especial utilizada na última camada de redes neurais classificadoras e tem a função de normalizar as probabilidades estimadas na saída da rede. Uma diferencial desta função de ativação é que a *softmax* é aplicada a uma matriz e produz uma outra matriz de mesmas dimensões, enquanto as demais funções de ativação apresentadas são aplicadas em um número real e produzem um outro número real.

Em problemas de classificação é comum dividir um problema com muitas classes em problemas de classificação binária, resultando em um número de neurônios igual ao número de classes na camada de saída. Caso seja utilizada a função sigmóide para obter suas ativações, cada um deles produzirá uma estimativa da probabilidade de que a entrada pertença à classe associada ao neurônio, porém, o somatório dessas probabilidades será diferente de 1, algo que não ocorre para a função *softmax*.

Outro problema da função sigmóide em problemas com muitas classes é que $\sigma(z) \approx 1$ para altos valores de z e $\sigma(z) \approx 0$ para baixos valores. Isso faz com que, após a ativação, uma mesma probabilidade possa ser associada a esses elementos ainda que uns sejam muito maiores ou menores que outros. Portanto, na saída desse tipo de RNA é necessário que se tenha um conjunto de probabilidades normalizado.

Para um vetor $Z = \begin{pmatrix} z_1 & z_2 & \dots & z_n \end{pmatrix}^T$, a função *softmax* é definida conforme a Equação 2.18.

$$softmax(Z) = \begin{pmatrix} \frac{e^{z_1}}{\sum_{j=1}^{n} e^{z_x}} \\ \frac{e^{z_2}}{\sum_{j=1}^{n} e^{z_x}} \\ \vdots \\ \frac{e^{z_n}}{\sum_{j=1}^{n} e^{z_x}} \end{pmatrix} \tag{2.18}$$

Como pode ser observado na Equação 2.18, tira-se a exponencial de cada valor de z_j e divide-se pelo somatório de todas as exponenciais, fazendo com que todos os elementos sejam positivos e seu somatório seja igual à unidade. Outra característica desejada é que o valor da saída j cresce à medida que z_j cresce, evitando casos em que dois valores de entrada distintos poderiam levar à mesma probabilidade de saída.

No contexto de redes neurais, o vetor Z referenciado na Equação seria composto pelos valores de z calculados em todos os n neurônios de uma determinada camada, de modo que $softmax(Z)$ produzirá as ativações de todos os neurônios da camada.

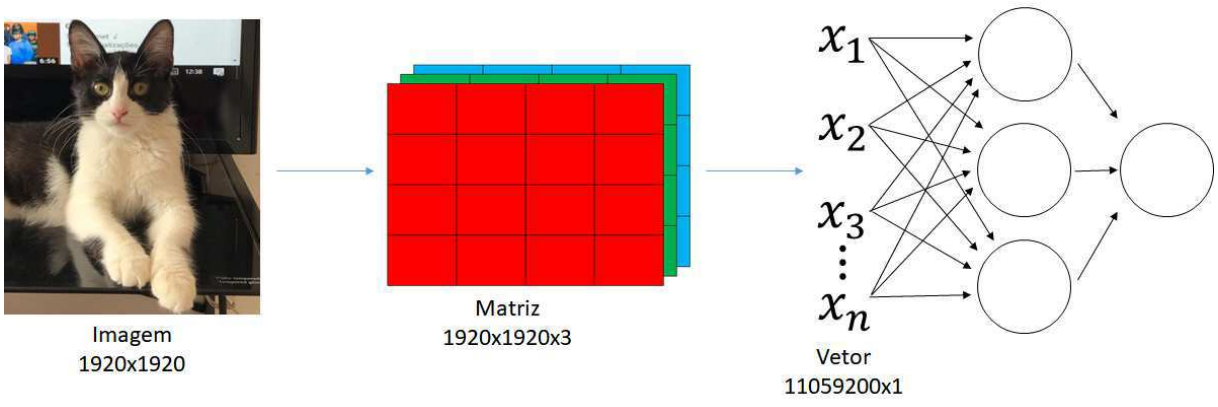
2.6 Redes Neurais Convolucionais

Os primeiros trabalhos sobre redes convolucionais (CNN - *Convolutional Neural Networks*) surgiram no fim da década de 80, (FUKUSHIMA, 1988) e (LECUN et al., 1990) e objetivavam o reconhecimento de números manuscritos. No trabalho, ele menciona que o princípio básico é minimizar o número de parâmetros que devem ser aprendidos pelo algoritmo, sem que haja grande prejuízo na performance da rede. Neste tipo de arquitetura, as multiplicações de matrizes que eram realizadas em cada camada são substituídas por convoluções entre imagens, de modo que a quantidade de parâmetros em cada camada da CNN independe do número de elementos na camada anterior.

Em redes neurais, a quantidade parâmetros treináveis se torna muito grande quando se trabalha com imagens. No computador, uma imagem colorida de $H \times W$ pixels é representada por três canais, cada um sendo uma matriz de H linhas e W colunas. Como pode ser visto na Figura 15, essas matrizes são convertidas em um único vetor de $n = H.W.3$ linhas, correspondentes ao vetor X, indicado na Figura. Para uma imagem de 1920×1920 pixels, o vetor teria mais de 11 milhões de elementos, de modo que cada neurônio da primeira camada oculta teria um vetor de pesos contendo mais de 11 milhões de parâmetros para serem treinados.

Nem toda camada de uma CNN realiza a operação de convolução, as que o fazem são denominadas camadas convolucionais, outros tipos de camadas existentes são as camadas de ativação, de *pooling*, de convolução transposta e as camadas totalmente conectadas. Cada camada desempenha uma operação específica e, na estrutura da CNN, tipos diferentes de camadas são alternados com o objetivo de criar uma rede mais robusta. A escolha dos tipos de camada a serem utilizados não é trivial e tem forte impacto no resultado final, de modo que muitas vezes é necessário se tentar várias configurações diferentes antes de se obter o resultado desejado.

Figura 15 – Ilustração de uma rede neural cuja entrada é uma imagem colorida.



Fonte: Próprio autor.

2.6.1 Camada Convolucional

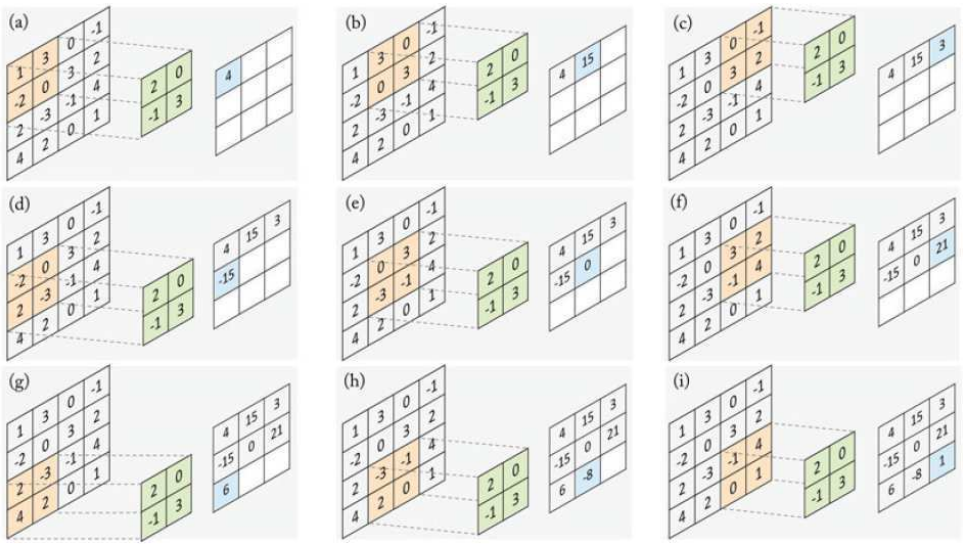
Camadas convolucionais são os componentes mais importantes de uma CNN. Envolve um conjunto de filtros que são convoluídos com sua entrada para produzir a saída, (KHAN et al., 2018). Um filtro (ou *kernel*) corresponde a uma matriz de números discretos e, em redes convolucionais, esses números são os parâmetros aprendidos durante o treinamento. O resultado da convolução com um filtro é chamado de mapa de características.

A convolução é uma operação esparsa, pois apenas algumas partes da entrada contribuem para uma determina porção da saída, e que reutiliza parâmetros, pois o mesmo conjunto de pesos é aplicado em múltiplas porções da entrada, (DUMOULIN; VISIN, 2016). Conforme a Figura 16, a operação pode ser vista como o deslizamento do filtro, representado em verde, por sobre a imagem, multiplicando os elementos que se sobrepõem e somando os resultados para produzir um elemento da saída, em azul. A operação continua até que todos os elementos da imagem de entrada tenham interagido com o filtro.

Existem três parâmetros relevantes para a operação, sendo estes o tamanho do filtro, o *padding*, e o *stride*. É comum o uso filtros com mesma altura e largura, cujo valor designado por f é geralmente ímpar. O *padding* (p) se refere a *zero-padding* e representa o número de bordas com valor 0 adicionadas em torno da imagem a ser convoluída. Já o stride (s) indica de quanto será o deslizamento do filtro sobre a imagem de entrada com relação à imagem de saída em cada etapa da convolução. Na Figura 16 foram utilizados $f = 2$, $p = 0$ e $s = 1$.

A convolução entre uma imagem e um filtro de dimensões $I \times I$ e $f \times f$ terá as

Figura 16 – Operação de convolução entre uma imagem 4x4 e um filtro 2x2.



Fonte: (KHAN et al., 2018).

dimensões de sua saída ($O \times O$) dadas pela Equação 2.19.

$$O = \frac{I + 2.p - f}{s} + 1 \tag{2.19}$$

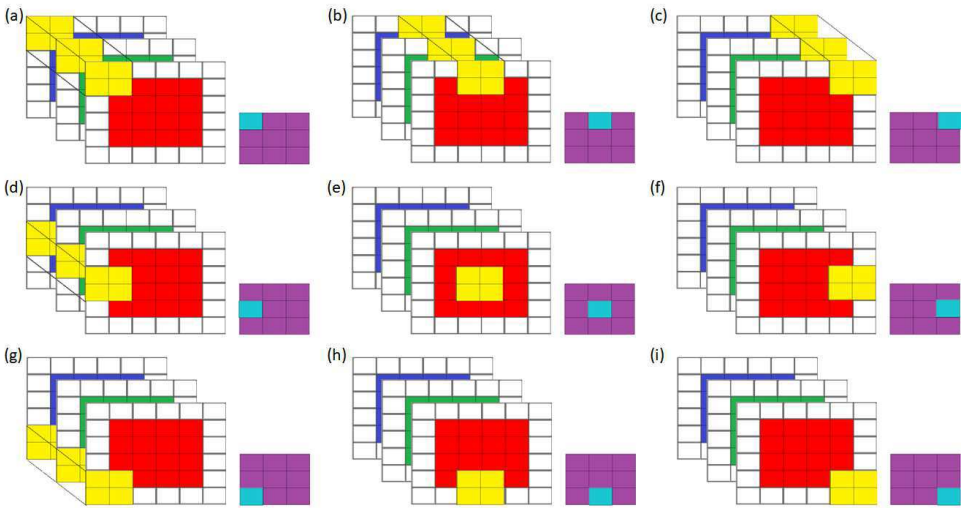
Para imagens com mais de 1 canal, a exemplo das imagens coloridas, os filtros bidimensionais descritos anteriormente deverão ser substituídos por filtros que apresentem o mesmo número de canais que a imagem de entrada. Neste caso, a cada posição do filtro sobre a imagem, ocorre a sobreposição entre elementos de cada um dos canais, esses elementos são multiplicados e então os resultados são somados.

A Figura 17 ilustra a convolução em três dimensões para uma imagem (vermelha, azul e verde) $4 \times 4 \times 3$ e um filtro (amarelo) $2 \times 2 \times 3$ utilizando $p = 1$ e $s = 2$. Tem-se, em branco, um contorno de zeros (*padding*) ao redor dos canais da imagem, a saída (roxo) pode ser vista ao lado com o elemento referente ao passo em questão destacado (em azul). Na Figura, o efeito do stride fica claro, pois agora são pulados 2 pixels por passo em contraste com a Figura 16, onde apenas 1 era pulado por vez.

Percebe-se que toda convolução entre volumes produzirá um resultado bidimensional cujas dimensões são dadas pela Equação 2.19. Porém, em cada camada convolucional são empregados um número n_f de filtros, de modo que os mapas de características resultantes das convoluções da entrada com todos os filtros são concatenados para produzir uma imagem de n_f canais, que será, então, enviada para a próxima camada da CNN. Naturalmente, cada camada pode ter um número de filtros diferentes.

Os parâmetros das camadas convolucionais são inicializados de forma aleatória e

Figura 17 – Operação de convolução tridimensional entre uma imagem colorida 4×4 e um filtro 2×2 com $p = 1$, $s = 2$.



Fonte: Próprio autor.

treinados de forma semelhante aos das RNAs, utilizando retropropagação. A quantidade de parâmetros depende apenas do número de filtros e de suas dimensões, de modo que para n_f filtros de tamanho f e n_c canais são empregados $(f^2.n_c+1).n_f$ parâmetros, sendo o termo $+1$ referente ao viés, onde 1 viés é adicionado ao resultado de cada uma das n_f convoluções. Se forem empregados 64 filtros $5 \times 5 \times 3$ em uma camada convolucional cuja entrada é uma imagem colorida de dimensões $1920 \times 1920 \times 3$, o número de valores a ser aprendidos será 4864, o que é bem menor que os 11 milhões necessários para uma RNA.

Ressalta-se que, conforme (ROSEBROCK, 2017), a operação descrita nessa seção é, tecnicamente, uma correlação cruzada. A diferença se dá justamente em que a convolução propriamente dita envolve a inversão do filtro antes de se realizar a operação, o que, pelo menos no contexto de *deep learning*, não altera o resultado final.

2.6.2 Camada de Ativação

Como o próprio nome sugere, esta camada é responsável por aplicar uma função de ativação sobre sua entrada. Geralmente esse tipo de camada é aplicado após as camadas convolucionais e não apresenta nenhum parâmetro treinável, levando a uma saída com as mesmas dimensões da entrada.

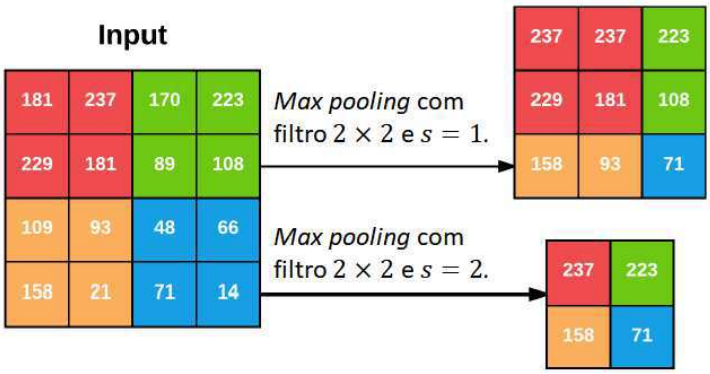
Nos diagramas de muitos trabalhos a camada de ativação é omitida, pois subentende-se que ela segue a camada convolucional, (ROSEBROCK, 2017). Também é comum chamá-la de camada ReLU, devido à ampla utilização dessa função de ativação em redes convolucionais. Neste trabalho a presença das camadas de ativação será indicada em conjunto com a sua localização em todos os modelos analisados.

2.6.3 Camada de Pooling

As camadas de *pooling* reduzem as dimensões dos mapas de características utilizando uma função que condensa subregiões da entrada, como retirar a média (*average pooling*) ou o máximo (*max pooling*) dos valores da subregião, (DUMOULIN; VISIN, 2016). O uso destas camadas reduz a quantidade de parâmetros e de computações na rede, além de ajudá-la a aprender os padrões ao invés de decorar os exemplos do conjunto de treino, (ROSEBROCK, 2017). As dimensões da imagem de saída são dadas pela Equação 2.19, fazendo $p = 0$.

A operação é similar à convolução, uma janela de tamanho f é deslizada pela entrada e, a cada passo, um valor é obtido segundo o tipo de *pooling* empregado. No caso de entradas com mais de um canal, a operação é realizada em cada canal de forma independente, produzindo uma saída com mesmo número de canais. O *max pooling* é o tipo de *pooling* mais comum nas aplicações, (GÉRON, 2017), e essas camadas não fazem uso de nenhum parâmetro treinável, (RASCHKA; MIRJALILI, 2017).

Figura 18 – Aplicação de *max pooling* com diferentes *strides*.



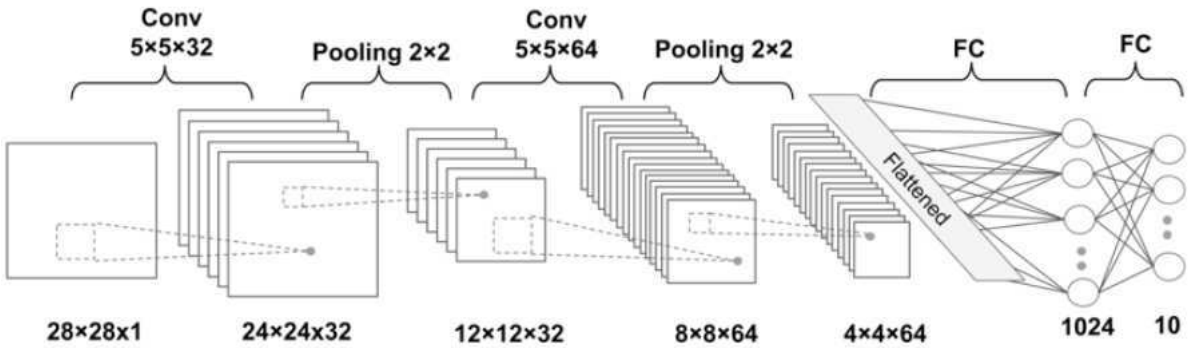
Fonte: Adaptado de (ROSEBROCK, 2017).

2.6.4 Camada Totalmente Conectadas

As camadas totalmente conectadas são posicionadas ao final de CNNs, (ROSEBROCK, 2017). Antes que uma imagem possa servir de entrada para este tipo de camada, ele deve ser redimensionada para se transformar em um vetor coluna, de forma semelhante ao realizado em uma RNA. Deste ponto em diante, a rede convolucional funciona exatamente como as redes neurais artificiais mencionadas anteriormente.

Neste tipo de camada se localizam a maior parte dos parâmetros da rede. Por este motivo, CNNs tendem a reduzir as dimensões das imagens em suas camadas ocultas, minimizando o número de parâmetros nas camadas totalmente conectadas. A Figura 19 ilustra uma arquitetura de CNN utilizando as camadas vistas até o momento. Subentende-se que as ativações seguem as camadas convolucionais.

Figura 19 – CNN com camadas convolucionais, de *pooling* e totalmente conectadas.



Fonte: (RASCHKA; MIRJALILI, 2017).

2.6.5 Camada de Convolução Transposta

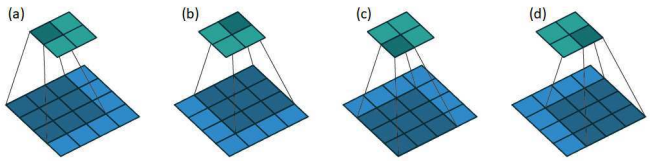
Essa camada emprega a operação de convolução transposta, por vezes chamada de convolução com *stride* fracionário. Geralmente é utilizada quando se quer realizar uma transformação no sentido inverso de uma convolução normal, ou seja, transformar algo com as dimensões da saída de uma convolução para algo com as dimensões da entrada de tal convolução sem perder o padrão de conectividade entre os elementos, (DUMOULIN; VISIN, 2016).

O nome convolução transposta é derivado da forma como a operação é comumente implementada. Uma convolução pode ser representada como uma multiplicação de matrizes, de modo que a forma de implementar a transformação no sentido inverso é a realização da multiplicação com uma das matrizes transposta. Mais informações sobre esta implementação estão disponíveis em (DUMOULIN; VISIN, 2016) e (KHAN et al., 2018).

Como o fato de ser uma convolução ou uma convolução transposta depende apenas das dimensões da matriz a ser multiplicada (se ela foi transposta ou não), uma convolução transposta pode ser expressa como uma convolução. Para tal, são necessários realizar alguns ajustes para garantir que se mantenha o padrão de conectividade entre os elementos.

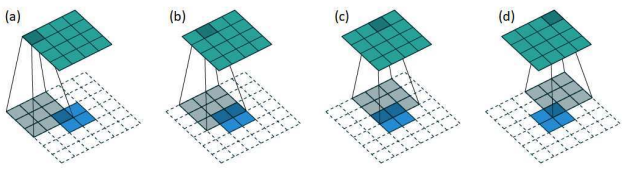
As Figuras 20 e 21 ilustram, respectivamente, uma convolução com $p = 0$, $s = 1$ e a convolução transposta correspondente. A primeira é realizada entre uma imagem 4×4 e um filtro 3×3, produzindo uma saída 2×2, enquanto que a segunda é realizada entre uma imagem 2×2 e um filtro 3×3, produzindo uma saída 4×4, conforme esperado.

Figura 20 – Convolução entre uma imagem 4×4 e um filtro 3×3.



Fonte: (DUMOULIN; VISIN, 2016).

Figura 21 – Convolução transposta entre uma imagem 2×2 e um filtro 3×3.



Fonte: (DUMOULIN; VISIN, 2016).

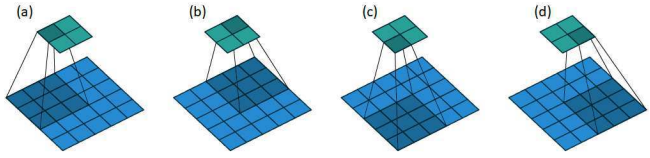
A imagem de entrada está representada em azul e a de saída em verde, o filtro, em cada momento, é representado pela região sombreada.

Percebe-se que, no caso da Figura 20, a convolução transposta pode ser realizada como uma convolução entre uma imagem 2×2 com dois contornos de *zero-padding* e um filtro de mesmas dimensões que o utilizado na convolução. O *padding* garante o mesmo padrão de conectividade entre os elementos da entrada e da saída nas duas operações, de forma que, na Figura 20,o *pixel* do canto superior esquerdo da entrada contribui apenas para o cálculo do *pixel* do canto superior esquerdo da saída e, na Figura 21, o mesmo padrão se repete. De forma análoga, as mesmas associações entre elementos feitas na convolução podem ser observadas na convolução transposta.

As Figuras 22 e 23 mostram o par de operações com a presença de *stride*. Fica claro que o uso de *stride* implica, na convolução transposta, em separar os elementos da imagem de entrada, adicionando zeros entre eles, antes de realizar a convolução com o filtro. Essa mudança faz com que o filtro se mova pela imagem de entrada em um ritmo menor que os elementos da saída são calculados, configurando um "*stride* fracionário".

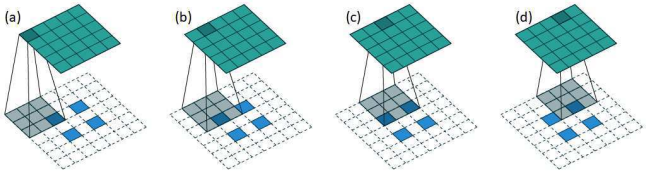
A aplicação mais comum desse tipo de camada é a realização de *upsampling*. Como operações de convolução geralmente reduzem as dimensões da entrada, operações de convolução transposta tendem a aumentá-las, permitindo a implementação de uma camada da CNN que aprende a aumentar as dimensões da imagem de forma otimizada. Como na camada convolucional, os parâmetros dessa camada são os elementos dos filtros, inicializados aleatoriamente, e o número de parâmetros pode ser calculado de forma semelhante.

Figura 22 – Convolução entre uma imagem 5×5 e um filtro 3×3.



Fonte: (DUMOULIN; VISIN, 2016).

Figura 23 – Convolução transposta entre uma imagem 2×2 e um filtro 3×3.



Fonte: (DUMOULIN; VISIN, 2016).

3 Colorização com Deep Learning

O trabalho mais antigo que se utiliza de *Deep Learning* para realizar a colorização automática de imagens é o de Cheng, Yang e Sheng (2015). Esse trabalho utilizou uma rede neural artificial cuja entrada era um vetor de características extraído de cada pixel e cuja saída era um par de valores referentes aos canais de cores do pixel. O procedimento era realizado para todos os pixels da imagem e, ao final, os valores eram concatenados para produzir a imagem de saída.

Em seguida, Iizuka, Simo-Serra e Ishikawa (2016) propuseram uma nova abordagem utilizando uma combinação de redes convolucionais. O objetivo da combinação das redes é extrair características locais e globais da imagem e fundí-las de modo que a colorização realizada fosse mais realista. O fato de a rede ser dividida em dois ramos, um referente às características locais e outro às globais, fez com que ela fosse capaz de efetuar transferência de estilo, caso imagens distintas sejam fornecidas a cada ramo, colorindo a primeira imagem com base nas características globais da segunda, se desejado.

Enquanto os dois trabalhos acima consideram a colorização automática como um problema de regressão, Zhang, Isola e Efros (2016) e Larsson, Maire e Shakhnarovich (2016) projetaram seus modelos considerando um problema de classificação objetivando alcançar resultados com maior diversidade de cores. Nessas CNNs os pixels são classificados em classes que são associadas os valores para os canais de cores que eles podem assumir. Observa-se que, dentre esses trabalhos, o segundo modelo citado se utiliza de uma hipercoluna, fazendo com que, para executá-lo, seja necessário uma máquina com quantidade razoável de memória disponível.

Cao et al. (2017) e Isola et al. (2017) utilizam *conditional generative adversarial networks*, proposta por (RADFORD; METZ; CHINTALA, 2015), para produzir diversas colorizações de uma mesma imagem e, então identificar a mais apropriada. Algo similar é feito por Deshpande et al. (2017), porém utilizando *variant autoencoders*, e por Royer, Kolesnikov e Lampert (2017) e Zhao et al. (2019), que utilizaram PixelCNN, proposta por (OORD et al., 2016). Um último trabalho, de abordagem singular, Chybicki et al. (2019) projetaram uma CNN voltada para a colorização de desenhos animados.

Infelizmente não foi possível utilizar modelos dos artigos consultados, de modo que foi necessário restringir a comparação a três trabalhos, Iizuka, Simo-Serra e Ishikawa (2016), Zhang, Isola e Efros (2016), Larsson, Maire e Shakhnarovich (2016), que disponibilizaram os códigos dos modelos no *github*. Esses trabalhos serão comentados com mais detalhes a seguir, em ordem cronológica, buscando evidenciar as estratégias utilizadas em suas arquiteturas. para lidar com o problema de colorir imagens de forma automática.

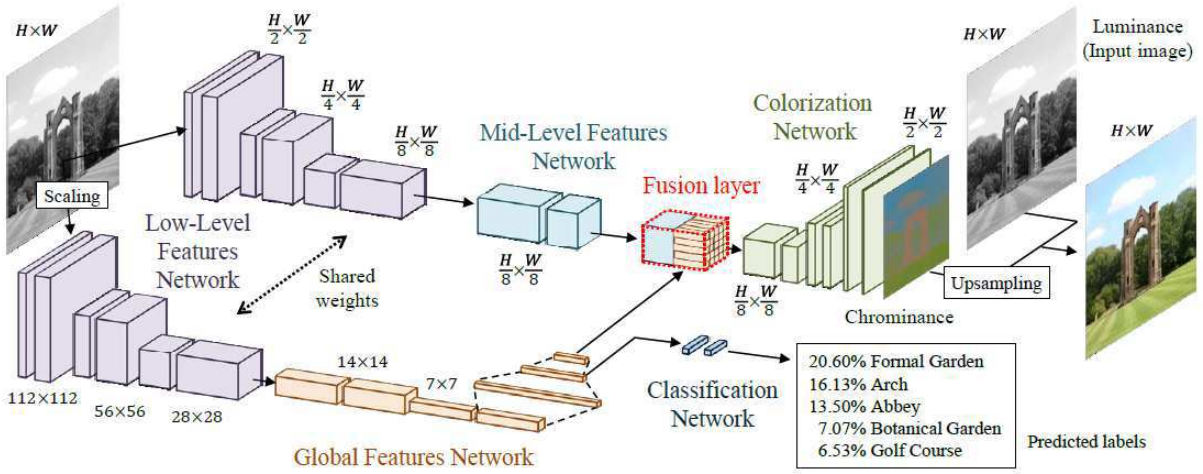
3.1 Iizuka, Simo-Serra e Ishikawa (2016)

A CNN de Iizuka, Simo-Serra e Ishikawa (2016) é composta de quatro componentes principais. Uma rede extratora de características de baixo nível, outra de características de nível intermediário, uma de características globais e uma última responsável pela colorização. Por características de nível baixo, intermediário ou global refere-se à complexidade das mesmas, sendo, para as de nível baixo, atributos simples como bordas horizontais e verticais, para as de nível médio, atributos mais complexos como formas geométricas e, para as características globais, padrões comuns à imagem como um todo. Subentende-se a presença de camadas de ativação após cada camada convolucional.

Como pode ser visto na Figura 24, existem duas redes de baixo nível, com apenas camadas convolucionais, e ambas compartilham os mesmos pesos (parâmetros treináveis), sendo, portanto, duas instâncias da mesma rede. Uma dessas redes é conectada à rede de nível médio, composta de camadas convolucionais, e outra conectada à rede de características globais, composta de camadas convolucionais e camadas totalmente conectadas. Nessa rede optou-se por não utilizar camadas de *pooling*, tendo sido utilizadas camadas convolucionais com *strides* maiores que a unidade para desempenhar uma função similar.

A razão pela qual uma das CNNs extrai características locais e outra o faz para características globais é resultante justamente das camadas utilizadas. Para extrair características de nível médio, são utilizadas camadas convolucionais, onde os elementos de cada camada se conectam apenas a alguns elementos de uma mesma vizinhança da camada anterior por meio da operação da convolução, de modo que essa rede estará muito mais focada nos padrões da imagem a nível local. Em contraste, a rede de características globais apresenta camadas convolucionais seguidas por camadas totalmente conectadas,

Figura 24 – Esquema da CNN proposta por Iizuka, Simo-Serra e Ishikawa (2016).



Fonte: Iizuka, Simo-Serra e Ishikawa (2016).

que se conectam a todos os elementos da camada anterior e permitem uma compreensão da imagem como um todo. Seguindo às duas redes, tem-se uma camada de fusão que, como o nome sugere, concatena as informações a nível local com as de nível global. Isso permite que as informações sejam enviadas em conjunto para a rede colorizadora.

A rede colorizadora é composta de uma sequência de camadas convolucionais e de sobreamostragem (*upsampling*). Nas primeiras, os parâmetros são escolhidos de modo a manter as dimensões da imagem de saída iguais às da de entrada, enquanto que, nas segundas, utiliza-se o método de interpolação pelo vizinho mais próximo para, a cada camada, dobrar as dimensões da imagem de entrada. O resultado é uma imagem de duas camadas referentes aos dois canais a e b no espaço de cores CIE L*a*b*, que são concatenados à imagem em escala de cinza (canal L) para que se obtenha a imagem colorida. A última camada dessa rede utiliza a função de ativação sigmóide, enquanto que as demais camadas utilizam a função ReLU.

Cabem, ainda, algumas observações. Neste modelo, ambas as redes de características de baixo nível são idênticas. Utilizar duas redes foi a solução dos autores para que a rede pudesse ser aplicada a imagens de qualquer tamanho, uma vez que, em toda a rede, a única limitação quanto ao tamanho da imagem se dá nas camadas totalmente conectadas, encontradas na rede extratora de características globais. Portanto, para uma imagem de dimensões arbitrárias $H \times W$, uma cópia redimensionalizada com tamanho 224×224 é enviada para o ramo de características globais enquanto que a imagem original percorre o outro ramo.

Além disso, uma outra rede, esta de classificação, pode ser vista no diagrama. Essa é uma RNA cuja entrada é a penúltima camada da rede de características globais e tem a função de classificar a imagem quanto ao seu contexto geral, como, por exemplo, se a imagem é de uma cena ao ar livre ou dentro de casa. Fornecer esse tipo de contexto ao sistema ajuda no sentido de, por exemplo, desestimular o colorizador a tentar colorir um céu numa imagem que foi identificada como sendo de uma sala de estar.

O problema de colorização, neste trabalho, foi tratado como um problema de regressão, de modo que a função de perdas utilizada foi o erro médio quadrático. Por outro lado, na rede classificadora foi utilizada a função de entropia cruzada somada ao erro médio quadrático do colorizador. Observa-se que, devido à retropropagação, a função referente ao colorizador afeta todas as camadas do modelo, enquanto que a referente ao classificador só afeta as camadas do classificador, do extrator de características globais e do extrator de características de baixo nível.

3.2 Zhang, Isola e Efros (2016)

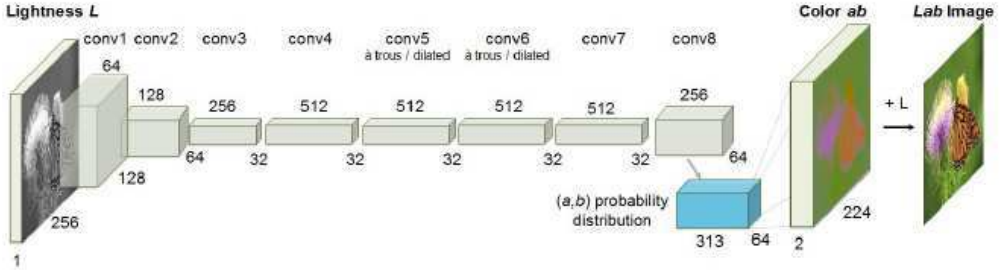
O trabalho proposto por Zhang, Isola e Efros (2016) considera o problema de colorização automática como uma tarefa de classificação. Em seu trabalho, os autores criticam a consideração do problema de colorização automática como uma regressão e o uso de funções de perdas como erro médio quadrático. Segundo eles, o uso desse tipo de função prejudica a variedade de cores na imagem de saída, pois, devido à ambiguidade existente quanto às colorizações possíveis, o MSE levaria ao uso de tonalidades acinzentadas.

Sendo assim, a função de perdas empregada foi uma variação da função de entropia cruzada. O espaço ab, dado pelos valores possíveis das coordenadas a e b do espaço CIE L*a*b*, foi dividido segundo uma grade de 10×10 e cada ponto da grade está associado a uma classe. No total, foram utilizadas 313 classes e, para favorecer a diversidade de cores, a função foi rebalanceada em função de uma distribuição probabilística levantada sobre o *dataset* utilizado, de modo que as classes referentes às cores mais comuns, como o cinza, foram penalizadas.

A CNN empregada, ilustrada na Figura 25 utilizou camadas de ativação após todas as camadas convolucionais. Na Figura, cada bloco convolucional representa duas ou três repetições de camadas convolucionais e a saída da rede é uma distribuição de probabilidades que indica a probabilidade de pertencimento de cada pixel a cada uma das 313 classes. Observa-se que, novamente, não foram utilizadas camadas de *pooling*, de modo que todas as alterações na resolução da imagem foram realizadas por meio de camadas convolucionais e de convolução transposta.

Definiu-se uma função para mapear essa distribuição de probabilidades nos canais de cores a e b da imagem. Teve-se como base a função *softmax*, com leves alterações seguidas da média aritimética dos 313 canais. Como a CNN projetada não tinha como saída os canais a e b, no treinamento era necessário aplicar a inversa da função proposta nas imagens do *dataset* para então aplicar a função de perdas. Mais informações sobre a função aplicada podem ser encontradas em (ZHANG; ISOLA; EFROS, 2016).

Figura 25 – Esquema da CNN proposta por Zhang, Isola e Efros (2016).



Fonte: Zhang, Isola e Efros (2016).

3.3 Larsson, Maire e Shakhnarovich (2016)

O modelo proposto por Larsson, Maire e Shakhnarovich (2016) teve como base a rede convolucional classificadora VGG-16, proposta por (SIMONYAN; ZISSERMAN, 2014) e ilustrada na Figura 26, com camadas de ativação (ReLU) implícitas após as camadas convolucionais. As mudanças realizadas foram com relação à primeira camada, que ao invés de receber uma imagem RGB normalizada receberá uma imagem em escala de cinza, e às últimas camadas da rede, que foram descartadas a partir da camada totalmente conectada fc1000, indicada na Figura 26.

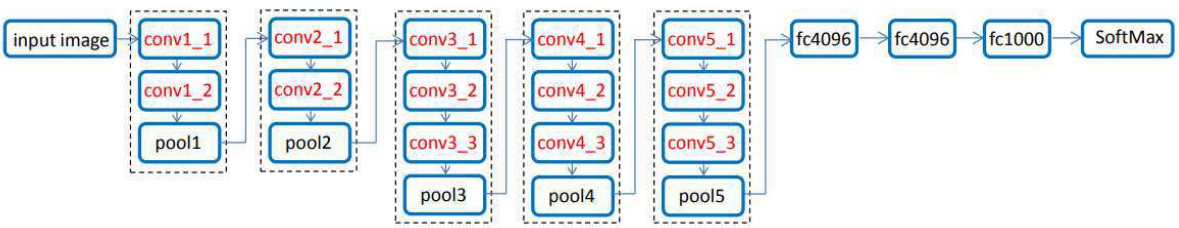
Os pesos da rede VGG não foram inicializados de forma aleatória, tendo sido aproveitados os pesos da rede original. Isso foi realizado na esperança de que ela retivesse parte das suas capacidades de classificação mesmo após o treinamento, visto que, para os autores, informações a respeito do contexto da imagem, como quais objetos estão presentes e onde eles se localizam são importantes para realizar a colorização. Para garantir que a rede mantivesse tais capacidades mesmo para imagens em escala de cinza, a rede VGG-16 foi treinada com uma passagem pelo *dataset* como rede classificadora após as modificações na primeira camada.

Neste trabalho, considera-se a colorização como um problema de classificação e, novamente, o espaço de cores foi subdividido em classes. Foram feitos experimentos os espaços L*a*b* e HCL e, para este modelo, o segundo proporcionou melhores resltados. Foram usadas estruturas denominadas hipercolunas, formadas a partir de dados retirados de todas as camadas da CNN. Tal recurso permite que a classificação seja realizada com base em características extraídas de diversos níveis de abstração.

No modelo proposto, cada hipercoluna é um vetor de 12417 elementos que é seguido por uma camada totalmente conectada de 1024 neurônios. Como o uso desse tipo de estrutura envolve um grande número de parâmetros treináveis e, por conseguinte, tem um alto custo de memória, foram extraídas hipercolunas para apenas 128 *pixels* escolhidos aleatoriamente. Foram aproximadas, por meio de interpolação linear, hipercolunas referentes aos pixels adjacentes, permitindo a colorização da imagem inteira. A Figura 27

Figura 26 – Esquema da CNN VGG-16 proposta por Simonyan e Zisserman (2014)

.

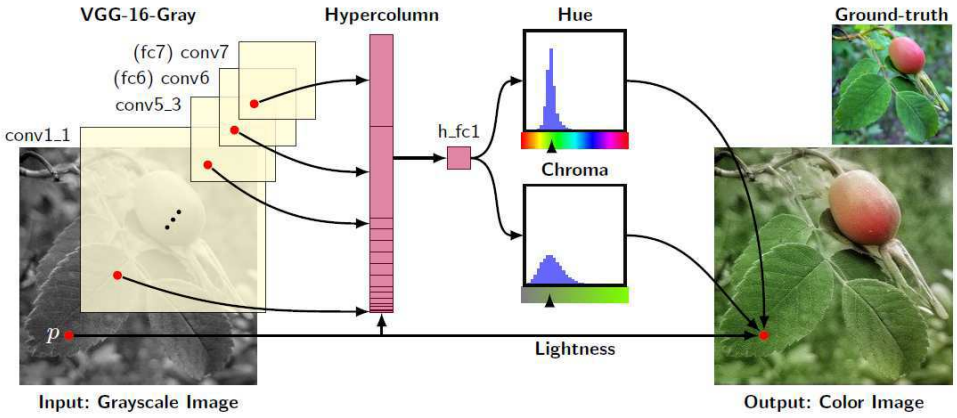


Fonte: Yu et al. (2016).

ilustra a extração da hipercoluna no modelo descrito.

Outra consideração a respeito do uso de hipercolunas é que a alta requisição de memória não está restrito ao treinamento. Durante a utilização da rede, redes convolucionais que utilizam hipercolunas devem reter na memória todos os valores calculados em todas as camadas da rede para que, ao final, sejam extraídas as hipercolunas. Em contrapartida, uma vez que são calculados os valores referentes a uma determinada camada, modelos convencionais de CNN têm a possibilidade de liberar da memória os valores de todas as camadas anteriores, minimizando seu uso de memória e permitindo uma execução mais rápida.

Figura 27 – Extração de uma hipercoluna a partir da CNN VGG-16 adaptada.



Fonte: Larsson, Maire e Shakhnarovich (2016).

4 Metodologia

Objetivando realizar uma comparação, o primeiro passo foi escolher um banco de dados (*dataset*) para que se pudesse realizar a colorização utilizando os modelos expostos previamente. O banco de dados escolhido não poderia ter sido utilizado para treinar nenhuma das CNNs a serem comparadas, de modo que se agrupou na Tabela 1 informações referentes aos conjuntos de dados utilizados pelos trabalhos citados para que fosse possível ter certeza que não houve reutilização de banco de imagens.

Após a definição do conjunto de dados, foi necessário definir indicadores quantitativos para poder avaliar a performance de cada modelo. Para tal, foram analisados os índices utilizados em todos os trabalhos descritos na seção anterior para determinar quais eram mais apropriados para realizar as comparações entre os trabalhos.

Observa-se que ressalvas quanto à utilização deste tipo de métrica são feitas por diversos autores, a exemplo de Zhang, Isola e Efros (2016), Larsson, Maire e Shakhnarovich (2016), Zhao et al. (2019). Segundo os autores, tais índices penalizam colorizações plausíveis que divergem das cores originais da imagem, o que não é desejado. Esse problema faz com que neste nicho exista grande disparidade quanto às formas de se avaliar a performance dos modelos e, consequentemente, comparar diferentes trabalhos numericamente torna-se uma tarefa desafiadora.

Pelas razões expostas no parágrafo anterior, autores como Iizuka, Simo-Serra e Ishikawa (2016), Zhang, Isola e Efros (2016), Cao et al. (2017), Zhao et al. (2019), recorrem a avaliações realizadas por grupos de pessoas quanto ao realismo das colorizações, os grupos continham entre 10 e 80 indivíduos. Uma análise semelhante foi realizada com objetivo determinar a capacidade dos modelos de convencer espectadores da autenticidade das suas colorizações. Os dados colhidos foram aproveitados para comparar o desempenho dos índices quantitativos e determinar os que melhor representavam a opinião dos participantes.

Tabela 1 – Informações sobre os bancos de dados dos trabalhos avaliados.

	Dataset	**Conjunto de Treino**	**Conjunto de Teste**	**Conjunto de Validação**	**Referencia**
Iizuka	Places scene dataset	2.327.958	19.546	-	(ZHOU et al., 2014)
Zhang	ImageNet	1.300.000	10.000	10.000	(DENG et al., 2009)
Larsson	ImageNet	1.200.000	10.000	1.000	(DENG et al., 2009)

Fonte: (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016), (ZHANG; ISOLA; EFROS, 2016), (LARSSON; MAIRE; SHAKHNAROVICH, 2016).

4.1 *Dataset*

Levando em conta os dados da Tabela 1, o banco de imagens escolhidos foi o LabelMe, (OLIVA, 2019). O banco de dados é composto de 2688 imagens coloridas, divididas em 8 classes (costa, campo aberto, floresta, montanha, estrada, rua, centro de cidade, arranha-céu), cada uma contendo entre 250 e 400 imagens de resolução $256 \times 256 \times 3$. Foram utilizadas imagens coloridas para possibilitar a realização de comparações entre os resultados obtidos com os modelos e o resultado esperado, as cores verdadeiras das imagens.

Cada classe tem imagens únicas e representam o ambiente descrito pela classe. Como existem 8 classes, pode-se esperar que sejam descritos vários tipos diferentes de ambientes. Outra vantagem da divisão do *dataset* em classes é que cada classe pode ser avaliada independentemente, permitindo identificar quais tipos de ambientes são mais facilmente coloridos pelos modelos.

Observa-se que, devido ao alto uso de memória requisitado pelo modelo de Larsson, Maire e Shakhnarovich (2016), as imagens foram redimensionadas para $128 \times 128 \times 3$. Além disso, para que todas as classes tivessem o mesmo peso nas análises realizadas, optou-se por utilizar um mesmo número de imagens em todas elas, reduzindo o número de imagens utilizadas para 250 por classe.

4.2 Indicadores Quantitativos

Analisando os trabalhos mencionados na seção anterior, constatou-se que os índices mais comuns para avaliar a performance dos modelos eram o erro RMS médio, a relação sinal ruído de pico (PSNR) e o índice de similaridade estrutural (SSIM). Tais valores foram computados para todos os modelos utilizando o *dataset* escolhido.

4.2.1 Erro RMS

O erro RMS (*root mean square*), é utilizado por Larsson, Maire e Shakhnarovich (2016) para o erro entre os canais a e b da imagem recolorida em relação aos da imagem original. Uma avaliação melhor implica em um menor valor de $\overline{E_{RMS}}$, sendo 0 o caso ideal.

Para duas imagens I e C de dimensões $H \times W$, o erro RMS médio é dado pela raiz do erro médio quadrático entre os canais **a** (a_I e a_C) e **b** (b_I e b_C) de cada imagem para cada *pixel* (i, j), sendo equacionado como:

$$\overline{E_{RMS}} = \sqrt{\frac{1}{H.W} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{[a_I(i,j) - a_C(i,j)]^2 + [b_I(i,j) - b_C(i,j)]^2}{2}} \tag{4.1}$$

4.2.2 Relação Sinal Ruído de Pico

A relação sinal ruído de pico (PSNR) mensura a razão entre a máxima potência de um sinal e o ruído presente em sua reconstrução. Esse indicador é comum na teoria de processamento de sinais e, dentre os trabalhos de colorização automática, é utilizado por Cheng, Yang e Sheng (2015), Larsson, Maire e Shakhnarovich (2016), Zhao et al. (2019) e Chybicki et al. (2019). Para este índice, devem ser utilizados todos os canais da imagem de modo que a forma mais comum de aplicá-lo é comparando os canais RGB da imagem original com os da reconstrução.

Para duas imagens I e C de dimensões $H \times W$, o PSNR é dado, em escala logarítmica, pela razão entre a potência máxima do sinal e a do ruído, definido em termos do erro médio quadrático. Deste modo, para as imagens I e C definidas, segue-se que:

$$MSE = \frac{1}{3.H.W} \sum_{k=1}^{3} \sum_{i=1}^{H} \sum_{j=1}^{W} [I(i,j,k) - C(i,j,k)]^2 \tag{4.2}$$

$$PSNR = 10.log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20.log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{4.3}$$

Observa-se que o termo MAX_I se refere ao valor máximo permitido para os elementos de I, de modo que, para as imagens utilizadas, $MAX_I = 255$. Quanto maior o valor de $PSNR$, melhor a avaliação do modelo.

4.2.3 Índice de Similaridade Estrutural

Este índice, proposto em (WANG et al., 2004), objetiva realizar uma avaliação da degradação da qualidade da imagem levando em conta sua perceptibilidade ao sistema visual humano, o que não é feito no erro RMS e nem no PSNR. Outro aspecto interessante deste indicador é que ele varia entre 0 e 1, sendo o valor ideal correspondente à unidade.

O índice é utilizado por Zhao et al. (2019) e Chybicki et al. (2019) e pode ser calculado para as imagens I e C por meio da Equação 4.4.

$$SSIM = \frac{(2.\mu_I.\mu_C + c_1).(2.\sigma_{IC} + c_2)}{(\mu_I^2 + \mu_C^2 + c_1).(\sigma_I^2 + \sigma_C^2 + c_2)} \tag{4.4}$$

Os termos μ_I e μ_C se referem às médias das imagens, σ_I^2 e σ_C^2 se referem às suas variâncias e σ_{IC} à covariância. As constantes c_1 e c_2 evitam divisões por zero no denominador e são proporcionais ao valor máximo que pode ser assumido pelos elementos da imagem, que neste caso é 255 e, portanto, são utilizados $c_1 = 6,5025$ e $c_2 = 58,5225$. O SSIM produz melhores resultados se aplicado às imagens localmente, de modo que, neste trabalho, o índice será calculado a partir de uma janela 5×5 e o valor global será a média dos valores calculados.

4.3 Avaliação Qualitativa

Até o momento, a forma mais comum de estimar o sucesso dos colorizadores automáticos tem sido através de consultas a observadores humanos para determinar o quão próximos da realidade são os resultados obtidos. Neste quesito são comumente utilizadas duas abordagens.

A primeira envolve mostrar uma sequência de imagens recoloridas e perguntar aos observadores, em uma escala de 0 a 10, o quanto as imagens mostradas parecem realistas. As imagens são escolhidas de forma aleatória e, quanto maior o número fornecido pelos espectadores, melhor a avaliação do modelo. Esse tipo de abordagem é utilizada nos trabalhos de Iizuka, Simo-Serra e Ishikawa (2016) e de Zhao et al. (2019).

Já a segunda envolve mostrar pares de imagens, a original e a recolorida, e pedir que os observadores identifiquem qual das imagens contém cores artificiais. Novamente, as imagens são escolhidas aleatoriamente e, quanto maior for o número de vezes que as imagens recoloridas forem escolhidas, melhor a avaliação do modelo. Esse tipo de abordagem é utilizada por Zhang, Isola e Efros (2016) e, com algumas modificações para adaptar o teste ao seu modelo, por Cao et al. (2017).

Neste trabalho, optou-se pelo segundo modelo uma vez que, no primeiro, as avaliações podem variar muito a depender da pessoa consultada, podendo proporcionar resultados inconsistentes. Tomou-se como base o experimento realizado por (ZHANG; ISOLA; EFROS, 2016), que teve inspiração no teste de Turing. No experimento, cada imagem ficava visível aos participantes durante um segundo e, uma vez que as duas imagens eram exibidas, não existia limite de tempo para que uma resposta fosse fornecida. Em cada realização do experimento, os 10 primeiros pares eram desconsiderados da análise e objetivavam familiarizar o participante com a tarefa, de modo que as colorizações artificiais eram evidentes, os 40 pares subsequentes correspondiam ao teste de fato. Ao todo, quarenta pessoas participaram do experimento.

As mudanças implementadas no experimento foram, mostrar a imagem em escala de cinza antes das versões coloridas e alterações no número de imagens. Foram utilizados 5 pares de imagens durante a primeira etapa, cujos resultados não eram considerados, e 48 pares na segunda, sendo 16 pares referentes a cada modelo, constituídos por 2 imagens de cada classe. O experimento foi implementado a partir de um programa escrito em python e disponibilizado em `https://drive.google.com/drive/folders/1w74ackKAyXbgp5zorksmtmxX9XBbyf_Q?usp=sharing`. Ao todo, 20 pessoas participaram do experimento, número superior ao de participantes no trabalho de Iizuka, Simo-Serra e Ishikawa (2016) e igual ao de participantes no estudo realizado por Zhao et al. (2019).

5 Resultados e Discussões

5.1 Resultados Quantitativos

Nesta seção os índices apresentados no capítulo anterior foram aplicados para proporcionar avaliações quantitativas dos resultados. As imagens coloridas do banco de dados foram convertidas para escala de cinza e recoloridas a partir dos modelos comentados anteriormente. A Tabela 2 mostra as médias dos cada um desses índices.

Pelas médias obtidas, observa-se que os melhores resultados foram alcançados pelo modelo de Iizuka, Simo-Serra e Ishikawa (2016) e os piores pelo de Zhang, Isola e Efros (2016). Entretanto, conforme mencionado no início do trabalho, o problema abordado tem uma natureza extremamente subjetiva, de modo que a análise mais aprofundada será necessária.

Nesse sentido, foram realizadas outras análises a respeito de como os resultados são distribuídos em cada modelo e, também, em cada classe. Os resultados dessas análises foram então comparados com as imagens reconstruídas do *dataset* escolhidas para representar as faixas de valores específicas de cada índice, permitindo observações a respeito da aplicabilidade dos referidos índices às colorizações.

Tabela 2 – Valores médios calculados para os três modelos.

	Iizuka	**Zhang**	**Larsson**
RMSE	5,85872	6,724098	5,838691
PSNR	30,07084	29,45938	29,73202
SSIM	0,893903	0,876892	0,885077

Fonte: Próprio autor.

5.1.1 Erro RMS

Foram calculados os erros RMS entre os canais a e b das imagens originais e reconstruídas. Para compreender a distribuição dos resultados, foram identificados os valores referentes ao primeiro, segundo e terceiro quartis, determinando assim em que faixa de valores se enquadra a maior parte das colorizações de cada modelo. Os resultados estão disponíveis na Tabela 3, sendo o i-ésimo quartil representado por $Q_{i/4}$.

Ao analisar a Tabela, percebe-se que o modelo proposto em (ZHANG; ISOLA; EFROS, 2016) apresenta os maiores erros em todos os quartis, enquanto os outros dois modelos apresentam valores bastante próximos um do outro. Levando em conta que o

valor máximo que cada elemento da imagem pode assumir é 255, os desvios da Tabela 3 estão situados entre 1,63% e 3,22% do valor máximo.

Tabela 3 – Valores dos erros RMS calculados para os três modelos.

	Iizuka	**Zhang**	**Larsson**
$Q_{3/4}$	7,498	8,204	7,337
$Q_{2/4}$	5,497	6,702	5,532
$Q_{1/4}$	4,152	5,19	4,144

Fonte: Próprio autor.

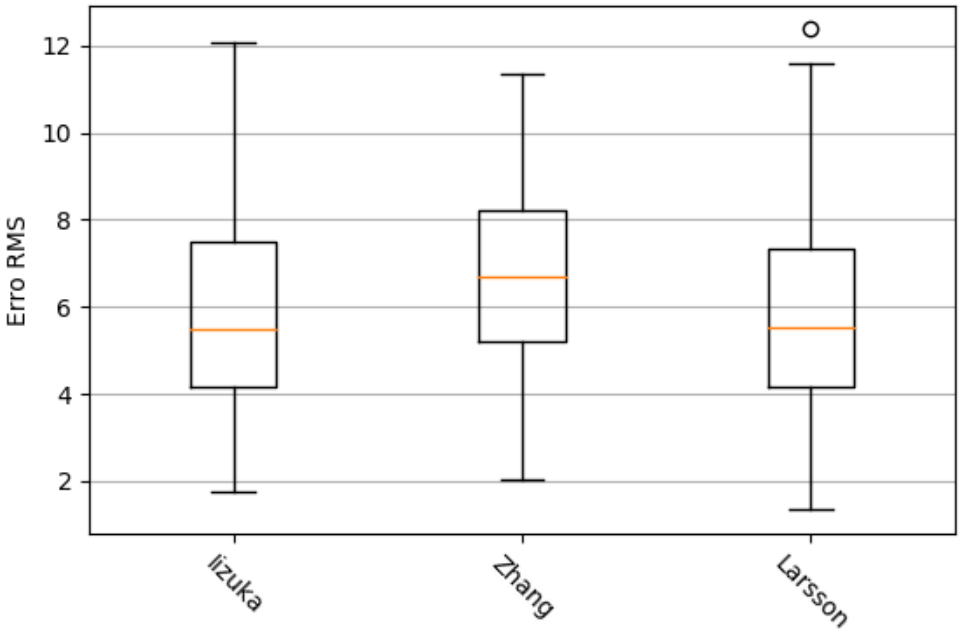
Em seguida, foram plotados lado a lado os diagramas de caixa referentes aos modelos (Figura 28) evidenciando resultados com distribuições semelhantes. O gráfico reforça os comentários feitos a respeito da Tabela 3 e mostra que a maioria dos resultados corresponde a erros de menos de 5%, 12 unidades, do valor máximo, 255. Também foram plotados diagramas de caixa em função das classes, Figura 29, cujos nomes foram traduzidos para facilitar a compreensão. As classes **estrada** e **rua** tipicamente apresentam menores erros, enquanto **costa**, **floresta** e **campo** apresentam erros mais elevados.

Para observar os efeitos da variação do erro RMS sobre as reconstruções, foram selecionadas imagens com diferentes valores de RMSE e agrupadas na Figura 30. Embora os valores calculados sejam similares para as imagens, colorizações realizadas apresentam diferenças significativas e algumas características dos modelos se destacam. Em alguns casos, o erro nas imagens em escala de cinza é menor que o das reconstruções, demonstrando certa inaptidão do índice na avaliação do problema, pois imagens colorizadas não necessariamente apresentam melhor avaliação do que as imagens dessaturadas.

As imagens coloridas pelo modelo de Iizuka, Simo-Serra e Ishikawa (2016) apresentam resultados mais realistas, o que condiz com o resultado apresentado nos diagramas das Figuras 28 e 29. Em contraste, colorizações realizadas por (LARSSON; MAIRE; SHAKHNAROVICH, 2016) são de certa forma dessaturadas quando comparadas às outras, se assemelhando às imagens em escala de cinza nas últimas três imagens, tornando as cores produzidas menos realistas se comparadas às de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016). As colorizações utilizando o modelo de (ZHANG; ISOLA; EFROS, 2016) são mais coloridas, mas frequentemente apresentam manchas.

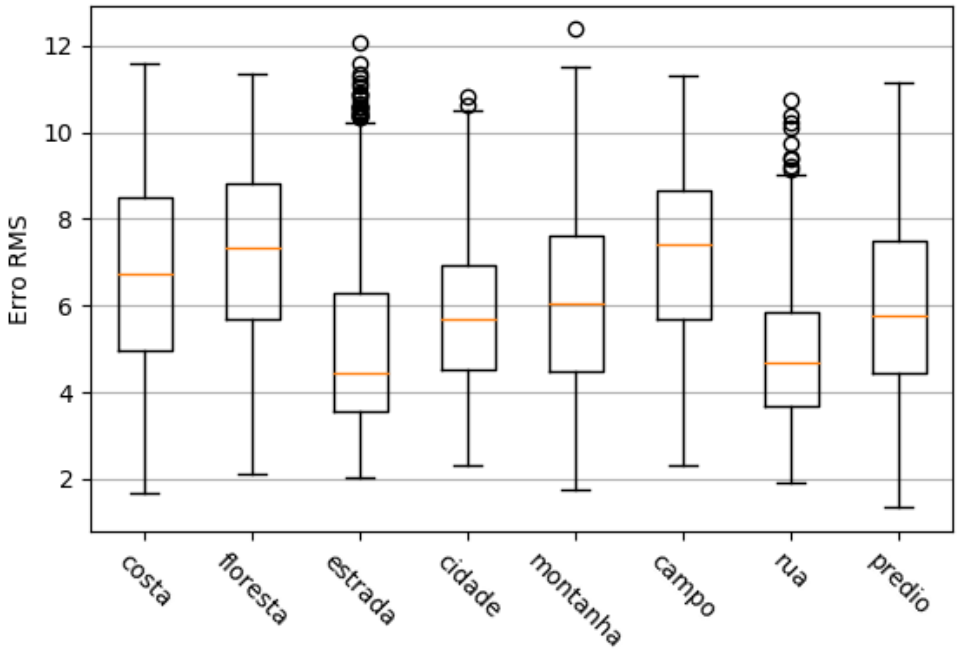
Resultados realistas e não realistas se distribuem nas diferentes faixas de valores apresentadas na Figura 30, indicando mais uma vez a inpatidão desse índice para avaliar as colorizações. Como o erro RMS corresponde à raiz quadrada do erro quadrático, espera-se desse último propriedades similares, indicando que ele também não seja uma boa alternativa como função de perdas neste tipo de CNN.

Figura 28 – Distribuição do erro RMS por modelo.



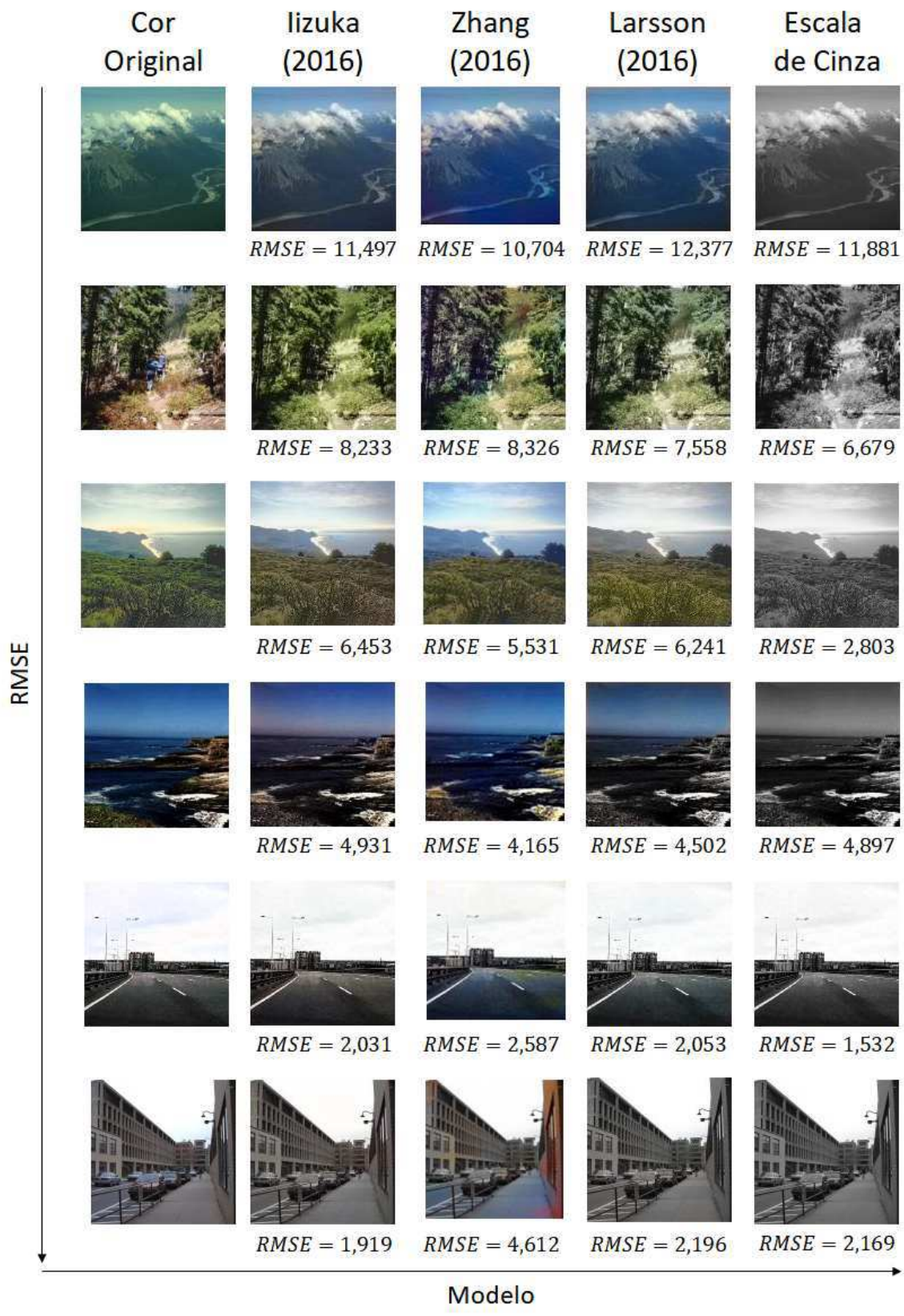
Fonte: Próprio autor.

Figura 29 – Distribuição do erro RMS por classe.



Fonte: Próprio autor.

Figura 30 – Comparação de imagens colorizadas e erro RMS corresponde.



Fonte: Próprio autor.

5.1.2 Relação Sinal Ruído de Pico

De forma semelhante à seção anterior, os valores de PSNR foram calculados, em dB, para as imagens recoloridas. Os dados foram agrupados na Tabela 4, que contém os quartis $Q_{1/4}$, $Q_{2/4}$ e $Q_{3/4}$ para cada modelo. Também foram plotados os diagramas de caixa, em função dos modelos (Figura 31) e em função das classes (Figura 32). A Tabela 4 reforça a similaridade entre os resultados dos modelos, sendo o de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) o mais bem avaliado e o de (ZHANG; ISOLA; EFROS, 2016) o pior. Nos diagramas da Figura 31 tem-se a grande maioria dos resultados entre 28 e 35 dB, situando o ruído introduzido pela reconstrução em uma magnitude de 25 a 56 vezes menor que o nível máximo do sinal. A Figura 32, também, reforça os resultados da seção anterior, indicando uma maior facilidade na colorização de imagens das classes **estrada** e **rua**.

Na Figura 33, as imagens foram agruapadas para que os efeitos de diferentes valores de PSNR sobre as imagens fossem observados. Pouco se pode dizer quanto ao realismo das imagens a partir deste índice, pois, além de algumas imagens em escala de cinza apresentarem um desempenho melhor, colorizações verossímeis são observadas em todas as faixas de valores apresentadas. Embora representada de forma mais realista pelo modelo de Larsson, Maire e Shakhnarovich (2016), a coloração atípica da água na última imagem não foi reproduzida com sucesso por nenhum dos modelos. Devido à existência de diversas colorações possíveis para certos objetos, fenômenos desta natureza serão mais frequentes em classes com esse tipo de elemento de coloração ambígua, como **floresta** e **campo**, e menos em classes com objetos de cores previsíveis, como **estrada** e **rua**, que só podem apresentar o mesmo tipo de coloração.

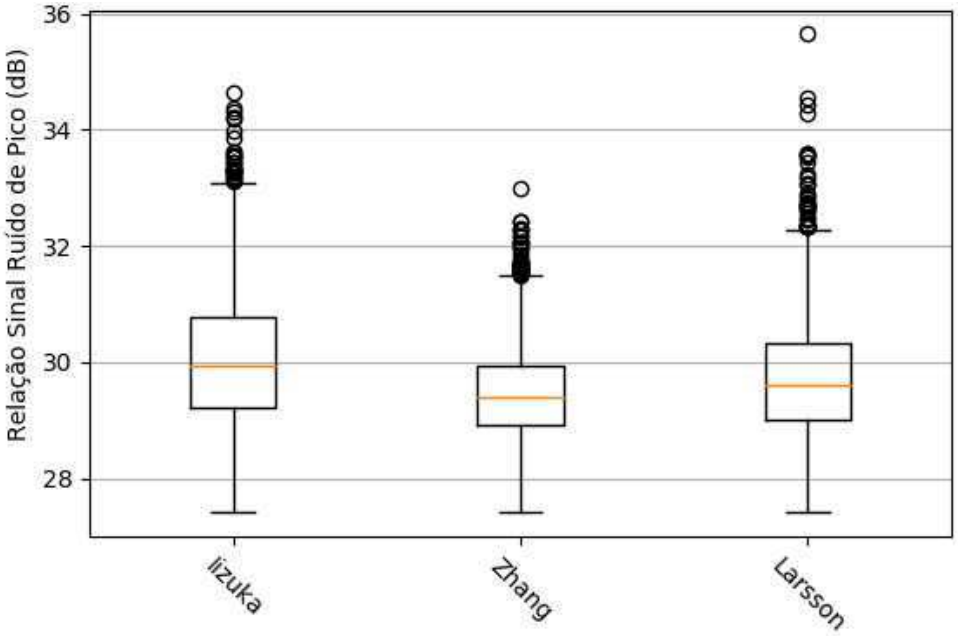
As reconstruções de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) novamente apresentaram consistência nas cores produzidas, não produzindo manchas ou imagens sem saturação. As manchas produzidas por (ZHANG; ISOLA; EFROS, 2016), produzindo tons de azul nas porções inferior e superior da imagem e verde na inferior, indicam *overfitting*, ou seja, que a CNN decorou características comuns aos exemplos de treinamento, como o azul do céu ou do mar e o verde de gramados. As cores obtidas a partir de (LARSSON; MAIRE; SHAKHNAROVICH, 2016) novamente são mais acinzentadas.

Tabela 4 – Valores de Relação Sinal Ruído de Pico, em dB, calculados para os três modelos.

	Iizuka	**Zhang**	**Larsson**
$Q_{3/4}$	30,763	29,946	30,326
$Q_{2/4}$	29,922	29,405	29,602
$Q_{2/4}$	29,21	28,905	29,003

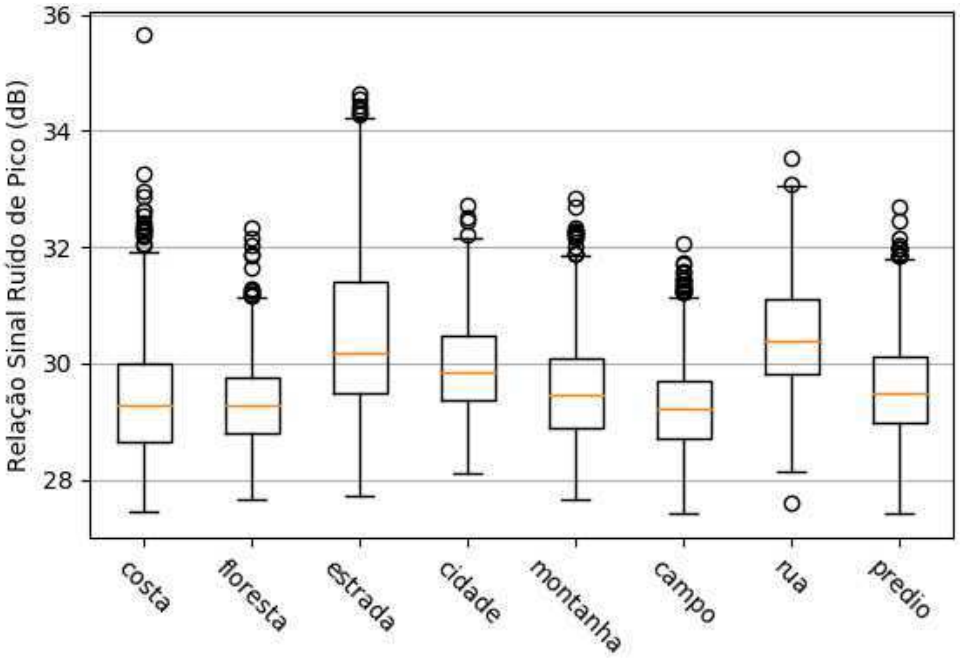
Fonte: Próprio autor.

Figura 31 – Distribuição do PSNR por modelo.



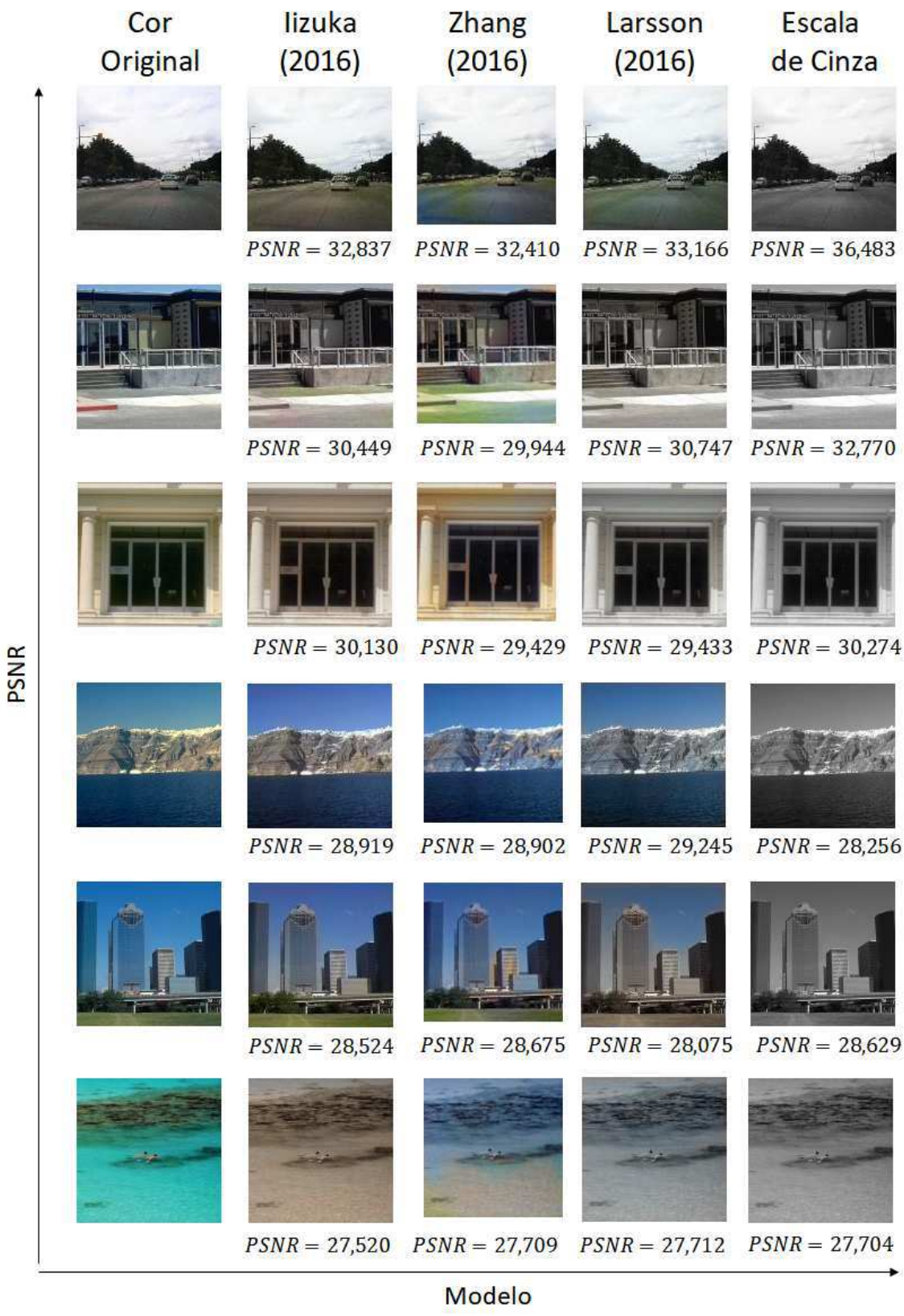
Fonte: Próprio autor.

Figura 32 – Distribuição do PSNR por classe.



Fonte: Próprio autor.

Figura 33 – Comparação de imagens colorizadas e PSNR corresponde.



Fonte: Próprio autor.

5.1.3 Índice de Similaridade Estrutural

A Tabela 5 apresenta os valores dos quartis $Q_{1/4}$, $Q_{2/4}$, $Q_{3/4}$ para o índice de similaridada estrutural. Observando os resultados, nota-se uma conformidade com os outros índices. O mesmo ocorre com os diagramas das Figuras 34 e 35, fornecendo altos valores para todos os modelos e uma indicação de qualidade mais alta para o modelo proposto por Iizuka, Simo-Serra e Ishikawa (2016), seguido pelo de Larsson, Maire e Shakhnarovich (2016) e, novamente com a pior avaliação, Zhang, Isola e Efros (2016).

Os exemplos agrupados na Figura 36 apresentam, na maioria dos casos, um SSIM maior para as imagens em escala de cinza. Isso indica que, este índice é mais influenciado pela presença de manchas de cor na imagem do que pela ausência total de pigmentação. Imagens verossímeis são produzidas pelos modelos em todas as faixas de SSIM apresentadas e, mesmo nos casos em que $SSIM \approx 1$, algumas imagens apresentam manchas e colorizações incorretas.

Em imagens onde uma tonalidade predomina sobre as demais, como nos prédios e o pôr do Sol das últimas imagens, as imagens em escala de cinza apresentaram baixos valores de SSIM. Caso esse padrão se repita, pode-se derivar uma forma de diferenciar as imagens coloridas das imagens em escala de cinza a partir do valor calculado, simplificando a avaliação dos modelos.

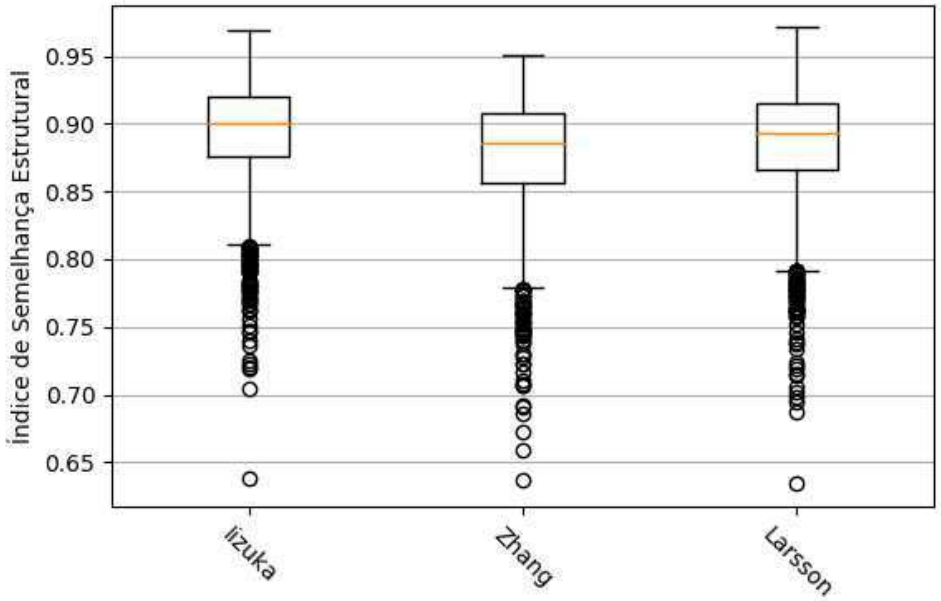
No conjunto da Figura 36, o modelo de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) dá alguns sinais de *overfitting*, apresentando manchas em algumas imagens e, na segunda ilustração, colorindo uma rodovia como um gramado. As cores produzidas por (ZHANG; ISOLA; EFROS, 2016) também apresentaram manchas e, na segunda ilustração, metade da rodovia foi colorida como um mar e a outra metade como um gramado. O único modelo que não produziu manchas foi o de (LARSSON; MAIRE; SHAKHNAROVICH, 2016), que foi capaz de colorir totalmente os objetos das imagens, mesmo que de forma dessaturada.

Tabela 5 – Valores do Índice de Similaridade Estrutura calculados para os três modelos.

	Iizuka	**Zhang**	**Larsson**
$Q_{3/4}$	0,921	0,908	0,915
$Q_{2/4}$	0,901	0,885	0,893
$Q_{1/4}$	0,876	0,856	0,866

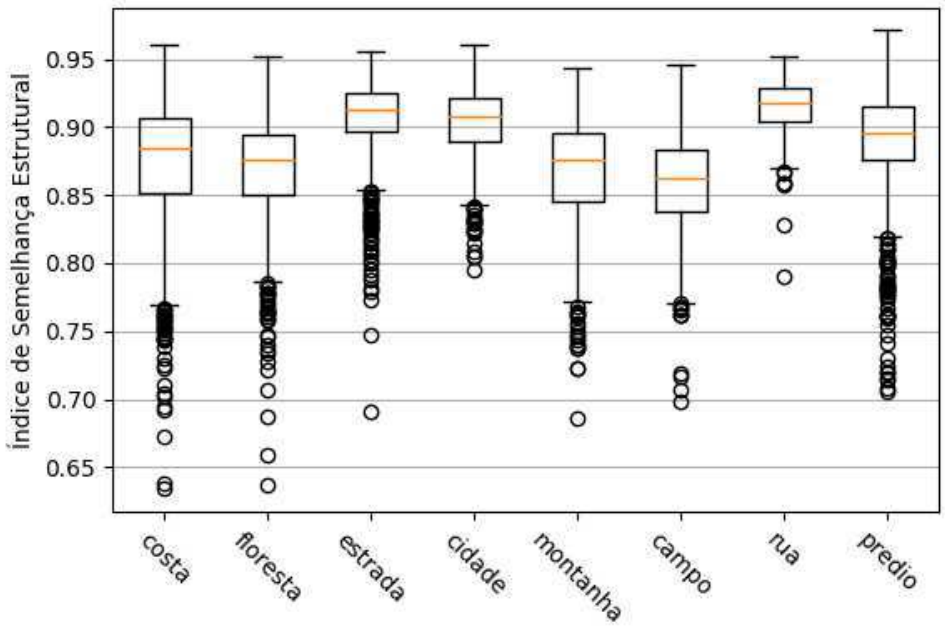
Fonte: Próprio autor.

Figura 34 – Distribuição do SSIM por modelo.



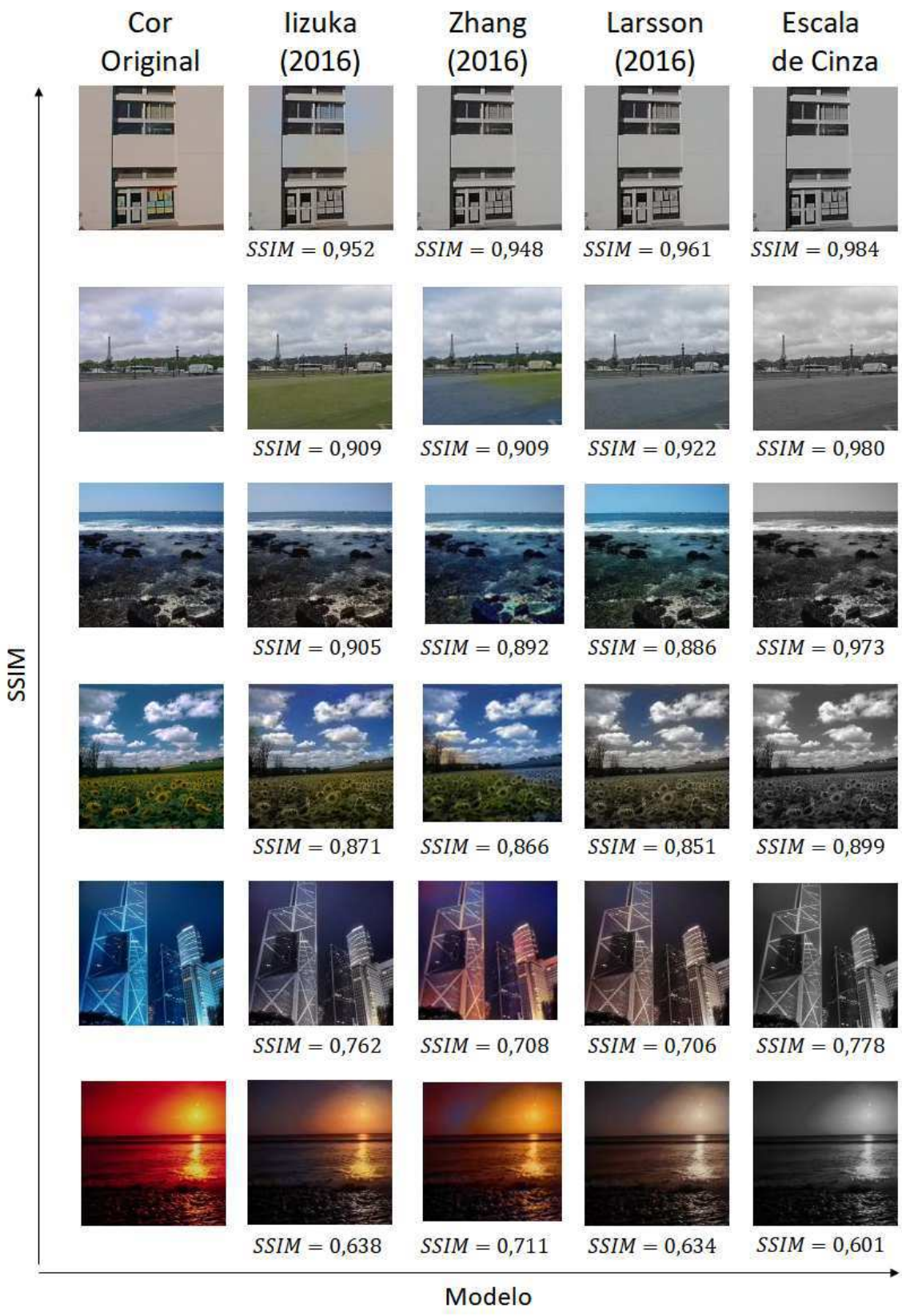
Fonte: Próprio autor.

Figura 35 – Distribuição do SSIM por classe.



Fonte: Próprio autor.

Figura 36 – Comparação de imagens colorizadas e SSIM corresponde.



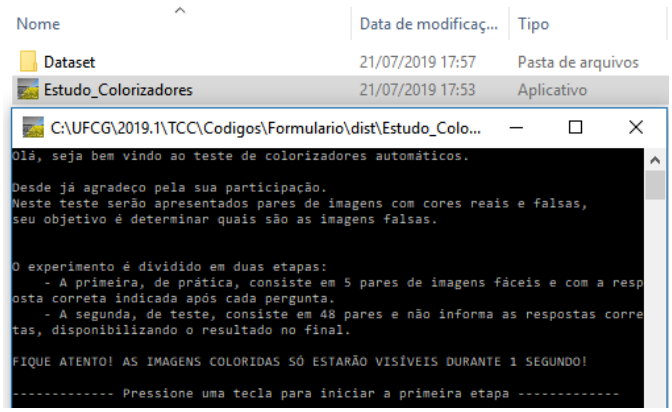
Fonte: Próprio autor.

5.2 Avaliação Qualitativa

Os resultados do experimento descrito na seção anterior mostraram que, em média, os usuários identificaram imagens reconstruídas como originais em 41,46% dos pares apresentados. Obteve-se, em média, um erro em 50% das classificações nas imagens coloridas pelo modelo de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) e para os demais modelos, (ZHANG; ISOLA; EFROS, 2016) e (LARSSON; MAIRE; SHAKHNAROVICH, 2016), as taxas de erro médio foram respectivamente de 43,18% e 31,25%.

O experimento foi realizado com o auxílio de 20 voluntários. Imagens do programa utilizado podem ser vistas a seguir, como a mensagem introdutória (Figura 37), as posições onde cada imagem aparecia (Figuras 39 e 40) e os resultados divulgados uma vez que o experimento houvesse terminado (Figura 38).

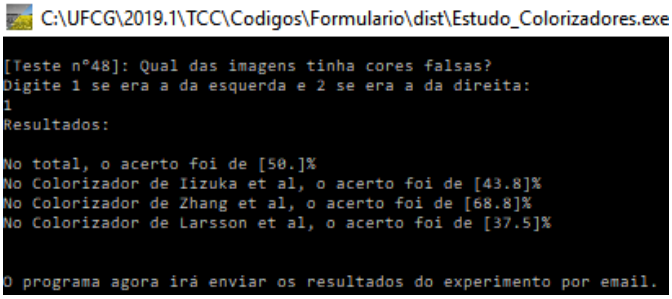
Figura 37 – Mensagem introdutória do programa referente ao Teste de Usabilidade.



Fonte: Próprio autor.

Neste teste, um erro de 50% indica que as imagens colorizadas são tão convincentes quanto as próprias imagens originais. À medida que a taxa de erro cai, mais fácil é para os participantes de identificar as imagens com cores artificiais, de modo que nesta avaliação o

Figura 38 – Resultados exibidos ao final do programa.



Fonte: Próprio autor.

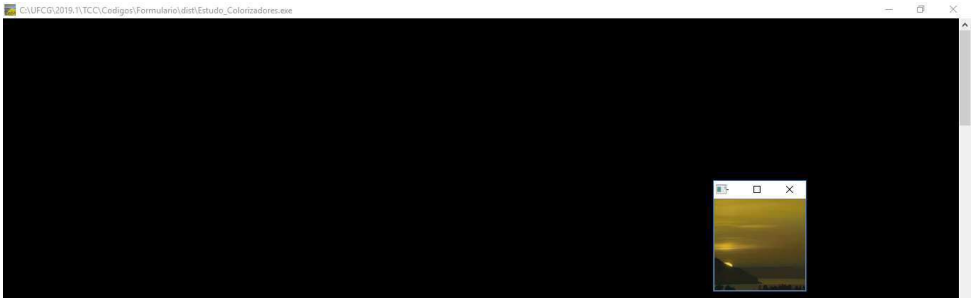
modelo mais bem avaliado foi o de Iizuka, Simo-Serra e Ishikawa (2016) e o pior avaliado foi o de Larsson, Maire e Shakhnarovich (2016), possivelmente devido às colorizações acinzentadas produzidas, permitindo assim a identificação das colorizações artificiais.

Por outro lado, o modelo proposto por (ZHANG; ISOLA; EFROS, 2016) teve uma boa avaliação por parte dos observadores mesmo embora tenha apresentado os piores resultados nos índices quantitativos. De fato, os resultados apresentados por esse modelo são bastante realistas quando não são produzidas manchas em meio às cores da imagem. Além disso, as imagens produzidas por esse modelo chamam atenção devido às suas cores, especialmente em casos onde a imagem original já possui baixa saturação. Como pode ser visto na Figura 30, tais imagens com baixa saturação podem ser confundidas com falhas no processo de colorização se mostradas ao lado da imagem recolorida pelo modelo de (ZHANG; ISOLA; EFROS, 2016).

Figura 39 – Imagem com imagem colorida exibida à esquerda.

Fonte: Próprio autor.

Figura 40 – Imagem com imagem colorida exibida à direita.



Fonte: Próprio autor.

6 Considerações Finais

Em conformidade com os indicadores calculados e as observações realizadas, o modelo proposto por (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) obteve os melhores resultados dos trabalhos apresentados. Apesar desses resultados, os trabalhos realizados em (ZHANG; ISOLA; EFROS, 2016) e em (LARSSON; MAIRE; SHAKHNAROVICH, 2016) não devem ser desconsiderados, visto que, em todas as análises realizadas seus resultados também foram mais que satisfatórios e seus modelos apresentam características que poderiam ser incorporadas em novos projetos.

Utilizando um computador com CPU Dual core Intel Core i3-3110M 2.4GHz para colorir as 2000 imagens do *dataset* foram necessárias, aproximadamente, 1,5 horas ($2,7s/img$) com a rede de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016), 5 horas ($8,8s/img$) com a rede de (ZHANG; ISOLA; EFROS, 2016) e 70 horas ($126s/img$) com a rede de (LARSSON; MAIRE; SHAKHNAROVICH, 2016). Observa-se que a utilização dos modelos citados neste trabalho exige alta disponibilidade de recursos computacionais, sendo o uso de uma GPU recomendado. A colorização das imagens do banco de dados, a depender dos requisitos do modelo empregado, podia demorar até três minutos (LARSSON; MAIRE; SHAKHNAROVICH, 2016) para colorir uma única imagem de dimensões 128×128.

Com relação a trabalhos futuros, recomenda-se que, além da incorporação de novos modelos às comparações realizadas, sejam buscados novos métodos avaliativos. Uma tendência interessante nesse sentido é aplicada em (ZHAO et al., 2019), que utiliza uma rede neural artificial especializada em segmentação semântica para medir a capacidade das redes colorizadoras de incorporar informação semântica às imagens reconstruídas. Com relação ao projeto de uma arquitetura voltada para colorização, uma ideia que vem à mente é a utilização de uma arquitetura semelhante à proposta por (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016), porém, treinada para realizar a colorização como um problema de classificação.

Bibliografia

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. Citado na página 11.

BROWNLEE, J. *Difference Between Classification and Regression in Machine Learning*. 2019. Machine Learning Mastery. Disponível em: <https://machinelearningmastery.com-/classification-versus-regression-in-machine-learning/>. Acesso em: 14.07.2019. Citado na página 21.

CAO, Y. et al. Unsupervised diverse colorization via generative adversarial networks. *arXiv preprint arXiv:1702.06674*, 2017. Citado 4 vezes nas páginas 11, 34, 40 e 43.

CHARTRAND, G. et al. Deep learning: A primer for radiologists. *RadioGraphics*, v. 37, n. 7, p. 2113–2131, 2017. PMID: 29131760. Disponível em: <https://doi.org/10.1148/rg-.2017170077>. Citado na página 11.

CHENG, Z.; YANG, Q.; SHENG, B. Deep colorization. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2015. p. 415–423. Citado 3 vezes nas páginas 11, 34 e 42.

CHYBICKI, M. et al. Deep cartoon colorizer: An automatic approach for colorization of vintage cartoons. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 81, p. 37–46, 2019. Citado 3 vezes nas páginas 14, 34 e 42.

DENG, J. et al. ImageNet: A Large-Scale Hierarchical Image Database. In: *CVPR09*. [S.l.: s.n.], 2009. Citado na página 40.

DESHPANDE, A. et al. Learning diverse image colorization. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 6837–6845. Citado 2 vezes nas páginas 11 e 34.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. Citado 4 vezes nas páginas 28, 30, 32 e 33.

FRANCOIS, C. *Deep learning with Python*. [S.l.]: Manning Publications Company, 2017. Citado 3 vezes nas páginas 16, 19 e 20.

FUKUSHIMA, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, Elsevier, v. 1, n. 2, p. 119–130, 1988. Citado na página 27.

GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. [S.l.]: "O'Reilly Media, Inc.", 2017. Citado 2 vezes nas páginas 16 e 30.

HANNUN, A. et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014. Citado na página 11.

IIZUKA, S.; SIMO-SERRA, E.; ISHIKAWA, H. Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (TOG)*, ACM, v. 35, n. 4,

p. 110, 2016. Citado 15 vezes nas páginas 5, 9, 11, 12, 34, 35, 40, 43, 44, 45, 48, 51, 54, 55 e 56.

ISOLA, P. et al. Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1125–1134. Citado na página 34.

JUNCZYS-DOWMUNT, M. et al. Marian: Fast neural machine translation in c++. *arXiv preprint arXiv:1804.00344*, 2018. Citado na página 11.

KHAN, S. et al. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, Morgan & Claypool Publishers, v. 8, n. 1, p. 1–207, 2018. Citado 3 vezes nas páginas 17, 28 e 32.

LARSSON, G.; MAIRE, M.; SHAKHNAROVICH, G. Learning representations for automatic colorization. *arXiv preprint arXiv:1603.06668*, 2016. Citado 15 vezes nas páginas 9, 10, 11, 34, 38, 39, 40, 41, 42, 45, 48, 51, 54, 55 e 56.

LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. In: *Advances in neural information processing systems*. [S.l.: s.n.], 1990. p. 396–404. Citado na página 27.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 17.

MINSKY, M.; PAPERT, S. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969. Citado na página 17.

OLIVA, A. *LabelMe*. 2019. Computational Visual Cognition Laboratory. Disponível em: <http://cvcl.mit.edu/database.htm>. Acesso em: 10.07.2019. Citado na página 41.

OORD, A. Van den et al. Conditional image generation with pixelcnn decoders. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2016. p. 4790–4798. Citado na página 34.

RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. Citado na página 34.

RASCHKA, S.; MIRJALILI, V. *Python machine learning*. [S.l.]: Packt Publishing Ltd, 2017. Citado 2 vezes nas páginas 30 e 31.

ROSEBROCK, A. Deep learning for computer vision with python. pyimagesearch. 2017. Citado 3 vezes nas páginas 20, 30 e 31.

ROSENBLATT, F. *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*. [S.l.], 1961. Citado na página 17.

ROYER, A.; KOLESNIKOV, A.; LAMPERT, C. H. Probabilistic image colorization. *arXiv preprint arXiv:1705.04258*, 2017. Citado 2 vezes nas páginas 10 e 34.

SAMUEL, A. *Some studies in machine learning using the game of checkers. Reprinted in EA Feigenbaum & J. Feldman (Eds.)(1963). Computers and thought*. [S.l.]: McGraw-Hill, 1959. Citado na página 16.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado 2 vezes nas páginas 5 e 38.

STOKES, M. et al. A standard default color space for the internet-srgb. *Microsoft and Hewlett-Packard Joint Report*, 1996. Citado na página 13.

SUN, X.; WU, P.; HOI, S. C. Face detection using deep learning: An improved faster rcnn approach. *Neurocomputing*, Elsevier, v. 299, p. 42–50, 2018. Citado na página 11.

TOET, A. Multiscale color image enhancement. *Pattern Recognition Letters*, Elsevier, v. 13, n. 3, p. 167–174, 1992. Citado na página 13.

VOULODIMOS, A. et al. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, Hindawi, v. 2018, 2018. Citado na página 11.

WANG, X.; TAKAKI, S.; YAMAGISHI, J. Investigating very deep highway networks for parametric speech synthesis. *Speech Communication*, Elsevier, v. 96, p. 1–9, 2018. Citado na página 11.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, v. 13, n. 4, p. 600–612, 2004. Citado na página 42.

WIDROW, B.; HOFF, M. E. *Adaptive switching circuits*. [S.l.], 1960. Citado na página 17.

YU, W. et al. Visualizing and comparing alexnet and vgg using deconvolutional layers. In: *Proceedings of the 33 rd International Conference on Machine Learning*. [S.l.: s.n.], 2016. Citado na página 38.

ZHANG, R.; ISOLA, P.; EFROS, A. A. Colorful image colorization. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 649–666. Citado 15 vezes nas páginas 5, 9, 10, 12, 34, 37, 40, 43, 44, 45, 48, 51, 54, 55 e 56.

ZHAO, J. et al. Pixelated semantic colorization. *arXiv preprint arXiv:1901.10889*, 2019. Citado 5 vezes nas páginas 34, 40, 42, 43 e 56.

ZHOU, B. et al. Learning deep features for scene recognition using places database. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2014. p. 487–495. Citado na página 40.