UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO VITOR ELIAS CARVALHO

COLORIZAÇÃO DE IMAGENS UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA PROFUNDO

RIO DE JANEIRO
2020

JOÃO VITOR ELIAS CARVALHO

COLORIZAÇÃO DE IMAGENS UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA PROFUNDO

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. João Carlos Pereira da Silva

RIO DE JANEIRO

2020

CIP - Catalogação na Publicação

C331c

Carvalho, João Vitor Elias
Colorização de imagens utilizando técnicas de aprendizado de máquina profundo / João Vitor Elias Carvalho. -- Rio de Janeiro, 2020.
47 f.

Orientador: João Carlos Pereira da Silva.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2020.

1. Inteligência artificial. 2. Redes neurais profudas. I. Silva, João Carlos Pereira da , orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

JOÃO VITOR ELIAS CARVALHO

COLORIZAÇÃO DE IMAGENS UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA PROFUNDO.

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 14 de janeiro de 2020

BANCA EXAMINADORA:

João Carlos Pereira da Silva
D.Sc.(UFRJ)

João Antônio Paixão
D.Sc.(PUC-RIO)

Fabrício Firmino de Faria
M.Sc.(UFRJ)

Dedico este trabalho a amigos e família pelo apoio e amor incondicional que tanto me deram.

AGRADECIMENTOS

Agradeço ao meu professor orientador João Carlos, por todo o apoio durante o projeto e pela paciência. Agradeço ao Departamento de Ciência da Computação pela disponibilidade do ambiente de desenvolvimento deste trabalho.

"As convicções são inimigas mais perigosas da verdade do que as mentiras."

Friedrich Nietzsche

RESUMO

Técnicas de aprendizado profundo tem se mostrado muito promissoras no campo da área visão computacional. Pesquisas foram desenvolvidas no problema de colorização de imagens preto e branco. Esse problema se trata de gerar cores plausíveis a uma imagem preto e branco. Tal aplicação pode ser feita a imagens histórias e também para uso artísticos, como uma mudança de colorização de uma imagem. Utilizando-se de grandes bases de dados disponíveis publicamente, tais pesquisas desenvolvem modelos capazes de resolver o problema da colorização, através da aplicação de aprendizado profundo. Este trabalho propõe-se a ser uma introdução a visão computacional, utilizando técnicas de aprendizado profundo e apresenta um modelo, baseado em trabalhos anteriores, capaz de colorir imagens preto e branco. O modelo desenvolvido pode colorir elementos simples como vegetação, céu, árvores, etc, entretanto tem dificuldade para objetos mais complexos como pessoas, objetos artificiais e menos frequentes na base de dados. O modelo implementado utiliza uma arquitetura semelhante aos trabalhos de referência, alterando-se a base de dados, função de erro e discretização do espaço de cores.

Palavras-chave: colorização. imagens. preto e branco. redes neurais profundas convolucionais.

ABSTRACT

Deep learning techniques have been shown to be promising in the field of computer vision. Research has been developed in the problem of black and white image colorization. This problem is about generating plausible colors for an black and white image. We can use this to color historical images or for artistic use, as in changing the colorization of an already colored image. Using big databases available to the public, it was created models capable of solving the colorization problem, through deep learning. This work focus on being an introduction to the field of computer vision, using deep learning techniques, and develop a model, based on related work, capable of adding color to an black and white picture. The model created can color simple elements like grass, sky, trees, but has difficult coloring more complex objects like people, artificial objects and less frequent objects in the database. The model implemented uses a similar architecture to the related works, changing the database, objective function and discretization of the color space.

Keywords: latex. abntex. text editoration.

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação dos pixeis em uma imagem 15
Figura 2 – Espaço de cores L*a*b. É a concatenação de 5 imagens de tamanho 256 de altura e largura. Da esquerda para direita cada imagem possui o canal L com valores que variam de 0 até 100 em intervalos de 25. Cada imagem varia os canais A e B de acordo com o eixo vertical e horizontal respectivamente. 16
Figura 3 – Desafios - retirado do curso CSN231 de Stanford. Possíveis variações: ponto de vista, iluminação, escala, deformação, oclusão, intra-classe. Essa são só algumas das múltiplas variações que podemos encontrar em imagens. 17
Figura 4 – Esquema de camadas de uma rede neural 18
Figura 5 – Exemplo de aplicação de uma convolução a uma imagem retirado do curso CSN231 de Stanford. 20
Figura 6 – Processo de convolução. Imagem criada por DUMOULIN; VISIN . . . 20
Figura 7 – Exemplo de convolução aplicado a resolução de um problema específico como a detecção de bordas. 21
Figura 8 – Comparação de saturação entre imagens utilizando diferentes algoritmos e funções de perda. 24
Figura 9 – Distribuição de cores no espaço ab 25
Figura 10 – Arquitetura da rede profunda Faça-se cor! 26
Figura 11 – Exemplo de colorização por (XIAO; ZHOU; ZHENG, 2018). A primeira coluna são as imagens preto e branco, a segunda coluna são as entradas locais na primeira linha e entradas globais na terceira linha. A terceira coluna se trata dos resultados utilizando apenas a entrada local e global, respectivamente. Da quarta coluna em diante são resultados combinando entrada local e global ao mesmo tempo. 27
Figura 12 – Arquitetura da rede desenvolvida por (XIAO; ZHOU; ZHENG, 2018). . 28
Figura 13 – Exemplo do processo de super-amostragem para uma imagem utilizando a estratégia do vizinho mais próximo. 33
Figura 14 – Arquitetura da rede neural desenvolvida nesta monografia. Cada bloco significa uma convolução seguida por uma ativação RELU e em alguns casos uma normalização de lote. A altura do bloco significa o tamanho da imagem e a largura significa a profundida de cada cama. Toda vez que há uma mudança de tamanho é feita utilizando a sub-amostragem ou super-amostragem. 33

Figura 15 – A esquerda: discretização com valores definidos a priori de (ZHANG; ISOLA; EFROS, 2016). A direita: discretização linear subdividindo o espaço de cores AB. 34
Figura 16 – Comparação entre as funções de transformação de probabilidades em cor. Da esquerda para direita: imagem preto e branco, função esperança, probabilidade máxima, média das 3 cores de maior probabilidade, original. 35
Figura 17 – A esquerda: função de erro durante as épocas de treinamento. A direita: função de erro no conjunto de validação 36
Figura 18 – Exemplo de colorização de uma imagem. A esquerda: mapa de colorização. Centro: colorização do modelo. Direita: cores originais da imagem. 37
Figura 19 – Exemplo de colorização para visualização das probabilidades. A esquerda: imagem preto e branco. Centro: colorização do modelo. Direita: cores originais da imagem. 38
Figura 20 – Probabilidades da cor verde para a imagem exemplo da figura 23. Quanto mais clara a cor, maior a probabilidade daquele pixel conter aquela cor. 38
Figura 21 – Probabilidades de múltiplas cores para a imagem exemplo da figura 23. 39
Figura 22 – Probabilidades para toda a imagem exemplo da figura 23 de todas as cores possíveis. 39
Figura 23 – Evolução da acurácia de acordo com um limite(t). Imagem colorida pelo modelo à esquerda e original à direita. Área sobre a curva: 94.97% 40
Figura 24 – Exemplos de alto valor no gráfico de acurácia acumulada por limite de erro. Imagem colorida pelo modelo à esquerda e original à direita. . . . 41
Figura 25 – Exemplos de baixo valor no gráfico de acurácia acumulada por limite de erro. Imagem colorida pelo modelo à esquerda e original à direita. . 42
Figura 26 – Resultados da colorização realizada pela rede neural. A imagem a esquerda é o mapa de cores, a do meio se trata da colorização da rede neural e a direita a imagem original. 43
Figura 27 – Resultados da colorização realizada pela rede neural. A imagem a esquerda é o mapa de cores, a do meio se trata da colorização da rede neural e a direita a imagem original. 44

LISTA DE ABREVIATURAS E SIGLAS

CIE	Commission internationale de l'éclairage
RGB	Red, Green and Blue
RNA	Rede neural artificial
SGD	Stochastic Gradient Descent
RELU	Rectified Linear Unit
MSE	Mean square error

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Problema da Colorização de Imagens 13
1.2 Trabalhos relacionados 13
1.3 Proposta 14

2 CONCEITOS BÁSICOS 15
2.1 Representação de Imagens 15
2.2 Distância Semântica e abordagem baseada em dados 16
2.3 Aprendizado de Máquina 17
2.4 Redes Neurais Artificiais 18
2.5 Redes Neurais Convolucionais 19
2.6 Otimização 21
2.7 Preparando os dados e função de perda 23

3 TRABALHOS RELACIONADOS 24
3.1 Colorização de Imagens 24
3.1.1 Colorização de Imagens Vibrantes 24
3.1.2 Faça-se cor! 25
3.1.3 Colorização profunda interativa com entradas globais e locais 26

4 ABORDAGEM 29
4.1 Proposta inicial 29
4.2 Base de Dados e tratamento dos dados 29
4.3 Regime de treinamento 30
4.4 Arquitetura da Rede 30
4.4.1 Convolução 31
4.4.2 Ativação RELU 31
4.4.3 Normalização de lote 31
4.4.4 Super-amostragem 32
4.4.5 Softmax 32
4.4.6 Arquitetura em camadas 33
4.4.7 Saída da rede 34
4.5 Transformando distribuição de probabilidades em cores . 35
4.6 Função de erro 36
4.7 Segmentação de elementos 36
4.8 Visualizando a distribuição de cores prevista 37

5 RESULTADOS 40
5.1 Acurácia 40
5.2 Exemplos de resultados 41
5.3 Conclusão 42

6 CONCLUSÃO 45
6.1 Trabalhos futuros 45

REFERÊNCIAS 46

1 INTRODUÇÃO

A área da visão computacional foi muito afetada pelos avanços em técnicas de aprendizado de máquina profundo. A grande disponibilidade de dados, avanços nas técnicas e maior poder computacional, permitiram que novos modelos fossem treinados para resolver problemas complexos. Uma das aplicações desenvolvidas foi a da colorização de imagens preto e branco, que será o objeto de pesquisa deste trabalho.

1.1 PROBLEMA DA COLORIZAÇÃO DE IMAGENS

Dada uma imagem preto e branco, podemos imaginar como gerar cores para preenchê-la. Baseando-se em pistas semânticas e de textura pode-se imaginar quais cores seriam prováveis. Tipicamente a grama é verde, o céu é azul e a joaninha é vermelha. Nem tudo pode ser inferido de tal maneira, como por exemplo as cores de uma bola. A bola pode ser azul, vermelha, roxa, etc. Não só objetos artificiais podem variar de cor, mas também artefatos naturais como árvores variam em tons de verde, amarelo e marrom. Percebemos então que tal problema é ambíguo por natureza.

O objetivo da colorização automática de imagens é produzir uma imagem plausível e não uma única colorização da imagem correta. Esta tarefa então se torna factível: a de modelar as dependências estatísticas entre a semântica e textura de imagens com suas respectivas cores de modo a produzir cenas plausíveis. Dado uma imagem preto e branco, que possui informações sobre tons de cinza, queremos inferir as respectivas cores.

Essa tarefa não pode ser realizada sem conhecimento prévio sobre os padrões semânticos da imagem, ou seja, seu significado. Padrões como onde se localiza, que elementos estão presentes ou entendimento da cena. Em trabalhos mais antigos (LEVIN; LISCHINSKI; WEISS, 2004), (LUAN et al., 2007) utilizou-se conhecimento a priori de usuários para guiar essa colorização, através de marcações feitas na imagem preto e branco. Outros trabalhos mais recentes (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016), (ZHANG; ISOLA; EFROS, 2016), (XIAO; ZHOU; ZHENG, 2018) utilizam grandes bases de dados para obter esse conhecimento prévio de forma automática. O problema da colorização necessita de um entendimento visual da imagem dado que as cores devem fazer sentido de acordo com o que está contido na imagem.

1.2 TRABALHOS RELACIONADOS

Em computação gráfica, existem duas formas de fazer colorização comumente utilizadas: propagação guiada pelo usuário, onde o usuário é responsável por indicar quais cores devem ser usadas em certos pontos e o algoritmo as propaga para o restante da

imagem, como em (LEVIN; LISCHINSKI; WEISS, 2004) e (LUAN et al., 2007), ou totalmente automática, onde existe uma base de dados e utiliza-se técnicas de aprendizado para aprender padrões para colorização. Neste trabalho foca-se no estudo e aplicação da segunda forma.

Técnicas de aprendizado de máquina profundo aplicadas a visão computacional tem-se popularizado muito nos últimos anos. Competições como (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), onde se compara a acurácia de algoritmos, mostraram que aplicar técnicas de aprendizado de máquina profundo baseado em redes neurais traz resultados melhores do que dos métodos aplicados anteriormente.

Nos últimos anos, tem ocorrido muitos avanços nas técnicas de colorização automática baseadas em rede neurais profundas, como os propostos por (ZHANG; ISOLA; EFROS, 2016), (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016), (LARSSON; MAIRE; SHAKHNAROVICH, 2016) e (XIAO; ZHOU; ZHENG, 2018). Uma explicação mais detalhada dos trabalhos será realizada no capítulo 3.

O problema da colorização possui disponível uma quantidade abundante de dados a serem utilizados. Qualquer imagem pode ser convertida para preto e branco e utilizada como instância de treinamento para um modelo. Bases de dados de imagens já foram construídas (ZHOU et al., 2014) e são disponibilizadas para trabalhos acadêmicos.

1.3 PROPOSTA

Esse trabalho propõe-se a estudar os modelos que representam o estado da arte na área de colorização de imagens e que utilizam aprendizado de máquina profundo. Com base no trabalho de (ZHANG; ISOLA; EFROS, 2016) reproduziremos um modelo que seja capaz de realizar a tarefa de colorização com algumas variações. O modelo desenvolvido utiliza uma arquitetura semelhante a da referência, alterando-se a base de dados, função de erro e discretização do espaço de cores.

Esse trabalho está organizado da seguinte maneira: uma introdução ao aprendizado de máquina, assim como conhecimentos básicos sobre processamento de imagens e cores será fornecido no capítulo 2. No capítulo 3 é mostrado um resumo dos principais trabalhos relacionados utilizados como inspiração para esta monografia. Em seguida no capítulo 4 será detalhado o modelo implementado por esse trabalho. No capítulo 5 mostraremos os resultados obtidos. Por fim no capítulo 6 teremos a conclusão e trabalhos futuros.

2 CONCEITOS BÁSICOS

2.1 REPRESENTAÇÃO DE IMAGENS

Imagens são representadas digitalmente através de uma discretização em "pixeis". Esses são o componente mínimo de uma imagem, que é composta por $A \times L$ pixeis, onde A é a altura e L é a largura da imagem. Para o caso mais simples de uma imagem preto e branco cada pixel representa a intensidade da claridade naquela posição. Cada pixel possui um número associado que representa a claridade.

A figura 1 mostra uma imagem de tamanho $A \times L$ no lado esquerdo e um recorte de tamanho 10×10 no direito, onde cada número representa a intensidade de cor em cada posição. Os valores variam de 0, mais escuro, até 255 que é o mais claro. Os valores da parte da direita foram retirados do quadrado branco, indicado no lado esquerdo.

Cor é como o sistema visual humano mensura uma parte do espectro eletromagnético, que é conhecido como espectro visível. Para trabalhar com imagens, é preciso converter o espectro eletromagnético de cores que é contínuo em um espaço discreto. O espaço de cores é o modelo matemático em que cores são representadas numericamente permitindo a reprodução igual em diferentes dispositivos e contextos. Sendo um modelo matemático arbitrário possui características definidas pela modelagem. Um espaço de cores pode ser otimizado para ser utilizados por aparelhos digitais como o RGB, definido por seus três componentes: *red* (vermelho), *green* (verde) e *blue* (azul), ou para otimizar a percepção

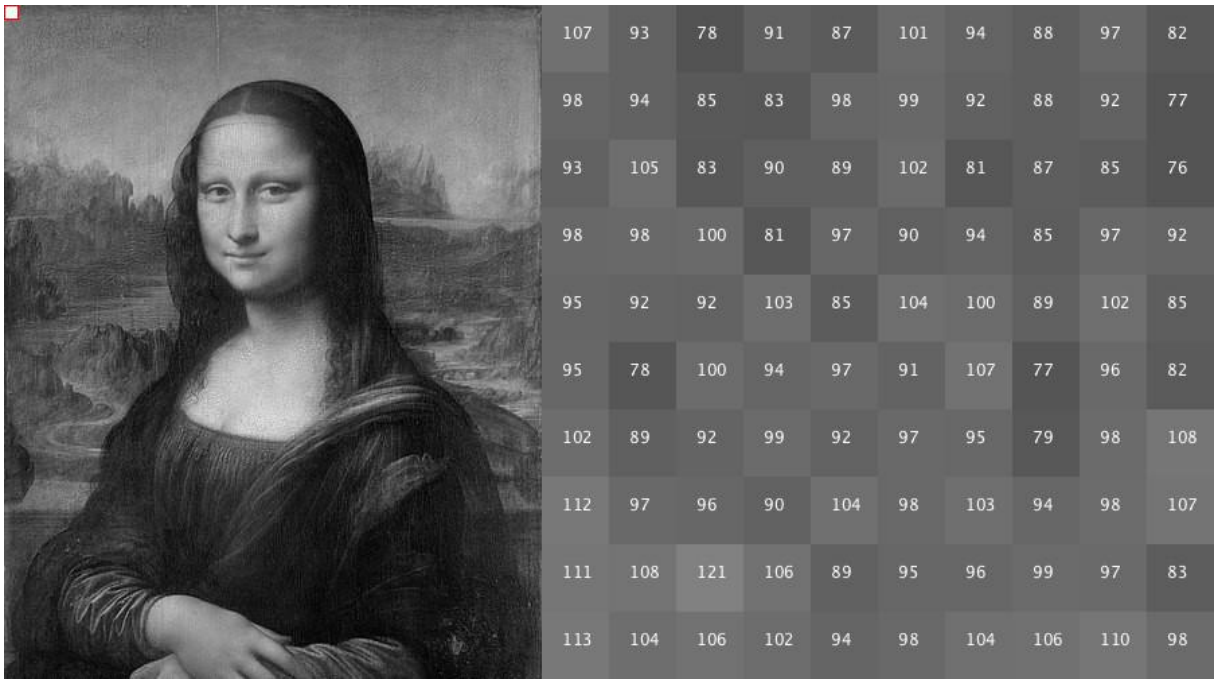


Figura 1 – Representação dos pixeis em uma imagem

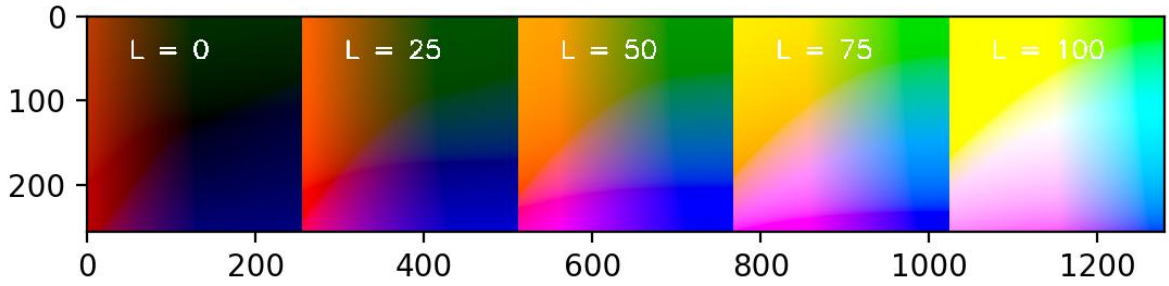


Figura 2 – Espaço de cores L*a*b. É a concatenação de 5 imagens de tamanho 256 de altura e largura. Da esquerda para direita cada imagem possui o canal L com valores que variam de 0 até 100 em intervalos de 25. Cada imagem varia os canais A e B de acordo com o eixo vertical e horizontal respectivamente.

humana como, por exemplo o espaço CIE L*a*b*, que separa o componente de luminância do de matiz.

Cada pixel da imagem possui então um vetor de tamanho N, onde N é a quantidade de canais, que é o que representa a cor do pixel. Por exemplo, no espaço de cores RGB cada pixel possui um valor de 0 até 255 para cada uma das cores primárias. A combinação da intensidade de cada cor primária pode gerar as outras cores. Por exemplo, a combinação entre vermelho e azul geram a cor violeta.

Outro espaço de cor comumente utilizado em trabalhos de tratamentos de imagens é o L*a*b. O espaço de cor L*a*b separa a luminosidade em um canal L e a cor em dois canais: A e B. O espaço de cores L*a*b possui uma conversão direta do RGB. Este espaço demonstra como podemos transformar os valores de um espaço de cor em outro, que pode ser mais conveniente.

2.2 DISTÂNCIA SEMÂNTICA E ABORDAGEM BASEADA EM DADOS

Entre a representação computacional de uma imagem e seu significado para os seres humanos, existe uma grande distância. Duas imagens podem representar a mesma coisa e serem computacionalmente bem diferentes. Um exemplo é ter uma imagem com as suas cores trocadas e posições dos pixeis rotacionadas. Para o sistema visual de um ser humano, aquela imagem ainda continua sendo semanticamente a semelhantes, apesar de os valores de cada pixel mudarem drasticamente. Isso implica que qualquer algoritmo, que tenha que trabalhar com imagens em um nível semântico, deve ser capaz de lidar com essa variabilidade.

Isso nos leva a alguns dos principais desafios ao lidar com essa variabilidade de imagens:

Então precisamos considerar todas essas categorias de variações ao desenvolver um algoritmo para lidar com imagens. Esse problema se mostra inviável de ser resolvido por um conjunto definido de instruções a serem implementadas por um ser humano. Ainda

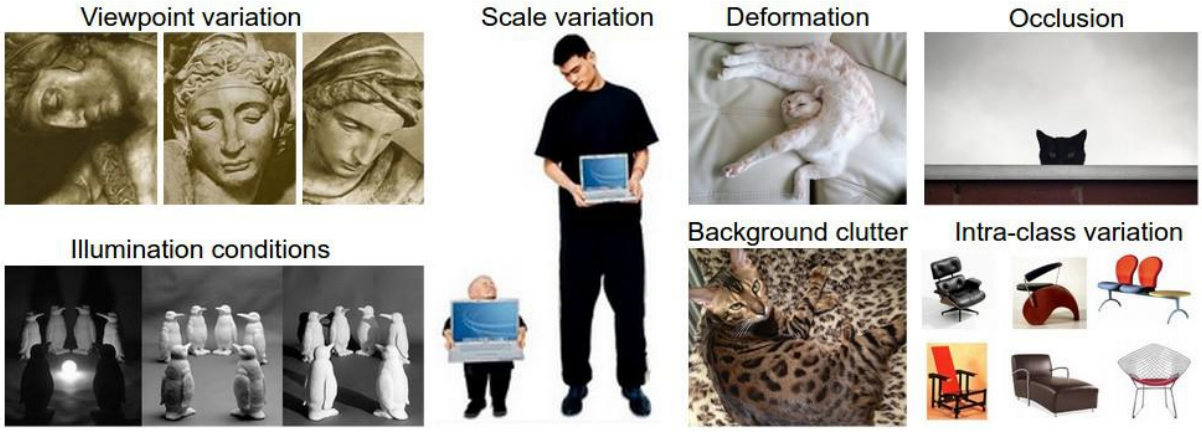


Figura 3 – Desafios - retirado do curso CSN231 de Stanford. Possíveis variações: ponto de vista, iluminação, escala, deformação, oclusão, intra-classe. Essa são só algumas das múltiplas variações que podemos encontrar em imagens.

que isso pudesse ser feito, seria um processo que não possuiria escala para expandir para diferentes classes ou problemas. O aprendizado de máquina é uma alternativa para esta situação.

2.3 APRENDIZADO DE MÁQUINA

Um algoritmo de aprendizado de máquina é aquele que pode aprender com os dados. (MITCHELL, 1997) nos provê uma definição mais precisa: "Um programa de computador é capaz de aprender com a experiência E em relação a alguma classe de tarefas T e métrica de performance P, se sua performance em tarefas em T, como medido por P, melhora com a experiência E".

No âmbito desse trabalho, vamos tratar da classe de tarefas de regressão. O objetivo é aproximar uma função de regressão desconhecida, denominada de função objetivo. Regressão é um tipo de tarefa que o algoritmo é requisitado a prever um ou mais valores numéricos.

$$f : X \to Y \tag{2.1}$$

Na equação 2.1, X corresponde ao conjunto de entradas possíveis do problema e Y representa o de saídas. Possuímos um conjunto de treinamento T que é uma coleção de dados em que é sabido $f(x) = y$ para todo $x \in T$. O conjunto de dados possui associado também uma distribuição $P(x)$ que representa a probabilidade associada a aparição de uma variável x.

$$H = \{h_1, h_2, h_3, ..., h_m\} \tag{2.2}$$

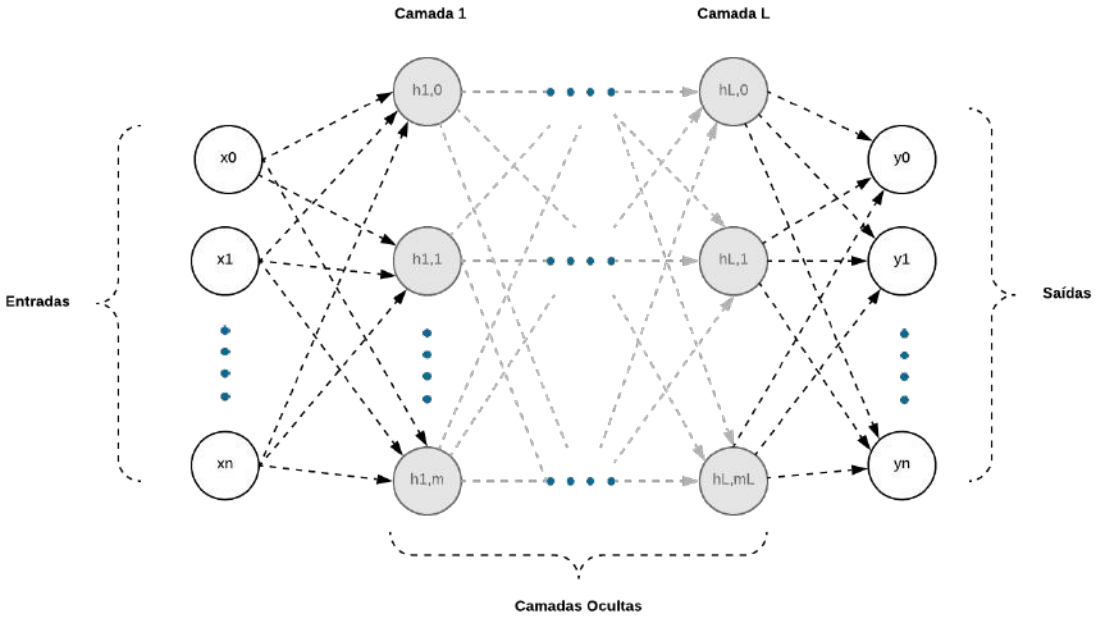


Figura 4 – Esquema de camadas de uma rede neural

Possuímos um espaço de hipóteses descritos pela equação 2.2, que podemos usar para aproximar a função de objetivo. O critério para distinguir qual hipótese representa melhor a função de objetivo é a função de erro. Um exemplo comumente utilizado para tarefas de regressão é a função erro médio quadrático, representada na equação 2.3.

$$\frac{1}{n}\sum_{t=1}^{n}(\hat{y} - y)^2 \tag{2.3}$$

Onde $\hat{y}$ é a solução proposta pela hipótese, enquanto y é a resposta correta provida pelo conjunto de treinamento. No problema do aprendizado procura-se minimizar a função de erro para cada $x \in T$ escolhendo-se a melhor hipótese disponível.

2.4 REDES NEURAIS ARTIFICIAIS

Uma rede neural artificial (“RNA”) é um modelo computacional utilizado em Aprendizado de Máquina. Ela é composta de uma estrutura de funções de transferências chamadas neurônios (neurons), que são aplicadas sucessivas vezes em um conjunto de entradas, também chamadas de características (features), gerando um conjunto de saídas, que podem ser utilizadas para realizar classificação ou regressão (GAMA et al., 2011). A estrutura de uma rede neural é organizada em camadas, sendo a primeira a camada de entrada, a última de saída e as intermediárias de ocultas.

Cada camada recebe o resultado da anterior e realiza operações em cima disso gerando a entrada para a próxima, até que o resultado chegue na de saída. As camadas

são compostas por neurônios que recebem como entrada o resultado da camada anterior e aplicam sua operação em cima deles. Cada neurônio possui seus parâmetros que diferenciam-no dos outros neurônios na mesma camada. Matematicamente cada neurônio pode ser definido pela relação abaixo.

$$a(x) = \sum_{i=0}^{n} w_i * x_i + b = w^T * x + b \tag{2.4}$$

$$y = h(x) = g(a(x)) \tag{2.5}$$

Onde $x = (x_1, x_2, ..., x_n)$ é o conjunto de dados fornecido a camada da rede neural e $w = (w_1, w_2, ..., w_n)$ é o vetor de pesos da camada. A função a é a como é calculado cada neurônio da camada. O termo b é denominado de fator de polarização.

A função g é chamada de função de ativação e é utilizada para introduzir não-linearidade na rede neural. A escolha da função de ativação afeta o resultado da rede e é considerado como um aspecto da arquitetura da rede neural. Alguns exemplos de função de ativação são: sigmoide logística, sigmoide tangente e a retificada linear.

O neurônio gera uma resposta y para ser usado por camadas posteriores ou, caso seja a última camada, para ser a saída da rede neural. Os parâmetros a serem treinados, durante a fase de treinamento desse algoritmo de aprendizado de máquina, são os pesos de cada neurônio e também os fatores de polarização. Essa é a arquitetura clássica de uma rede neural. A seguir vamos ver que redes neurais possuem outros tipos e arquiteturas.

2.5 REDES NEURAIS CONVOLUCIONAIS

Aprendizagem Profunda (Deep Learning) é o termo dado para se referir a redes neurais com grande número de camadas e com capacidade de construir hierarquicamente características intermediárias para representar dados. Um exemplo de rede neural profunda são as redes neurais convolucionais, que se baseiam em uma operação simples aplicada a imagens: a convolução.

No contexto de redes neurais estamos interessados em uma convolução de uma imagem. Considerando a imagem como uma matriz de tamanho A por L e o filtro de convolução uma matriz N por M, onde N e M são tamanhos arbitrários do filtro. O filtro de convolução funciona como o padrão que desejamos detectar na matriz da imagem.

Na figura 4 podemos ver um exemplo simples da aplicação de uma convolução. Possuímos a imagem na esquerda em azul, dois filtros de convolução no meio em vermelho e o resultado da operação a direita em verde. O filtro é aplicado pela multiplicação dos pesos a imagem na posição. Após isso movemos o filtro pela imagem para obter a saída final.

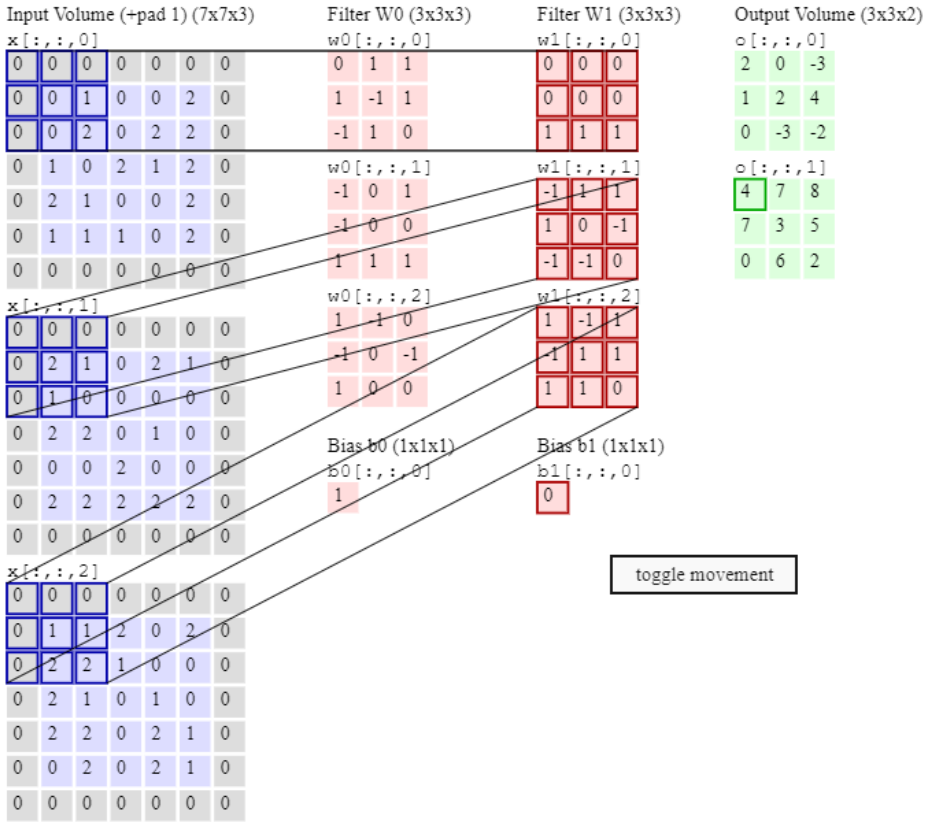


Figura 5 – Exemplo de aplicação de uma convolução a uma imagem retirado do curso CSN231 de Stanford.

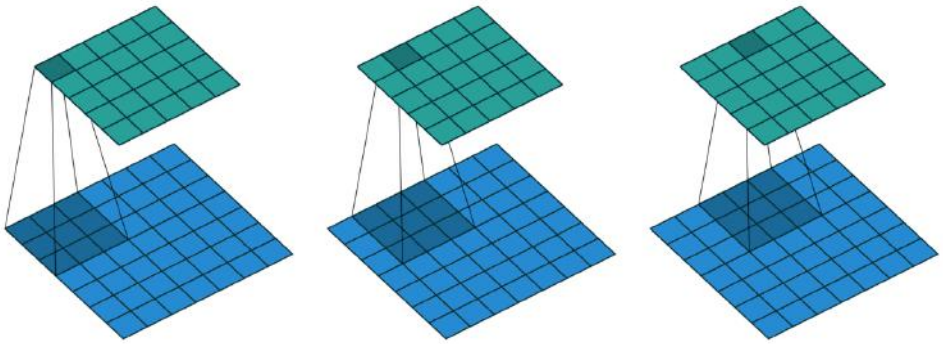


Figura 6 – Processo de convolução. Imagem criada por DUMOULIN; VISIN

Como mostrado na figura 12 o filtro é aplicado a cada pixel. Nesta figura temos a matriz da imagem em azul e a resultante em verde. Nessa figura podemos ver como o filtro é movido pela imagem para ser aplicado em todos os pixeis da imagem.

A função de convolução pode ser utilizada para detectar padrões em sinais utilizando sua característica de semelhança entre vetores. Podemos ver um exemplo de aplicação a detecção de bordas na figura 6. Nesse caso o filtro definido foi idealizado para a identificação de bordas. Este filtro dá peso maior para o pixel central e retira valor quando os pixeis adjacentes são valores mais altos.

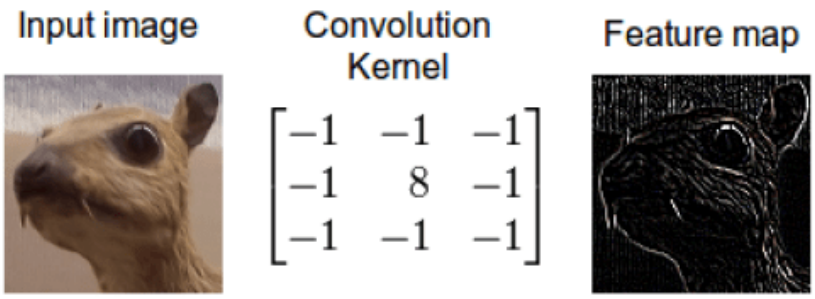


Figura 7 – Exemplo de convolução aplicado a resolução de um problema específico como a detecção de bordas.

Dado que é possível definir filtros para resolver problemas, podemos utilizar um algoritmo de aprendizado para otimizar os valores do filtro, para que resolvam melhor qualquer problema. Possuindo a imagem de entrada e o resultado de saída, podemos utilizar o fato da operação possuir um gradiente definido para otimizar os pesos, com o objetivo de definir a função que mapeia entrada para saída. Essa ideia foi explorada para visão computacional inicialmente por (LECUN et al., 1989) para identificação de dígitos escritos a mão.

Utilizando de uma série de convoluções aplicadas a imagem e posteriormente compondo camadas, podemos enxergar uma hierarquia de convoluções que nos fornecerá características mais complexas das imagens analisadas. Diferente de redes neurais tradicionais, cada neurônio é aplicado a apenas uma janela por vez, não tendo acesso a todo o conjunto de entrada da camada. Os pixeis são conectados localmente, então por vez cada convolução tem acesso apenas pixeis próximos espacialmente. Esse processo foi utilizado para gerar algoritmos de reconhecimento como os aplicados por (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) no conjunto de dados Imagenet com bons resultados.

2.6 OTIMIZAÇÃO

Ao treinar uma rede neural é necessário achar os melhores parâmetros que minimizem a função de erro. Uma rede neural possuí uma quantidade grande de parâmetro, então otimizá-la se torna um problema que pode levar horas, dias ou até meses. Foram desenvolvidos algoritmos especialmente para tal tarefa (GOODFELLOW; BENGIO; COURVILLE, 2016).

Existem meios de otimizar a rede neural utilizando parte do conjunto do treinamento ou todo o conjunto por vez. Os métodos que processam apenas uma parte são chamados estocásticos. O algoritmo mais utilizado para tal tarefa é o Stochastic Gradient Descent (SGD) ou Descida do Gradiente Estocástica. Versões modificadas desse algoritmo ainda são dominantes em relação à otimização de redes neurais((GOODFELLOW; BENGIO; COURVILLE, 2016)).

Ao otimizar uma rede neural queremos achar $\theta^* = argminf(x, \theta)$ onde f é a rede neural recebendo como parâmetro a entrada x e θ é o conjunto de parâmetros da rede. Para tal cálculo, utilizamos o gradiente da função de erro f em relação aos parâmetros θ. O gradiente é o vetor composto pela derivada parcial em relação a cada parâmetro. O gradiente nos dá informação sobre em qual direção há a maior variação da função. Iremos na direção contrária ao gradiente para minimizar a função de erro.

Atualizamos os parâmetros da seguinte forma: $\theta' = \theta - \epsilon * sinal(\nabla_\theta f(x, \theta)$, onde ϵ é a taxa de aprendizado e é um valor escalar positivo que determina o tamanho do passo de atualização. A taxa de aprendizado é um hiper-parâmetro do treinamento da rede.

A otimização converge quando cada elemento do gradiente é zero ou, na prática, próximo a zero. Esse algoritmo é conhecido como Descida do Gradiente. Ao aplicar esse algoritmo podemos utilizar todo o conjunto de dados ou parte dele. Ao utilizar todo o conjunto de dados esse algoritmo é conhecido como Descida de Gradiente Determinística ou em Lote. Ao utilizar apenas parte do conjunto por vez e fazer pequenos passos em direção a otimização esse algoritmo é definido como Descida de Gradiente Estocástica((GOODFELLOW; BENGIO; COURVILLE, 2016)).

Outros algoritmos surgiram para tornar o SGD mais robusto. Um avanço realizado foi o desenvolvimento dos algoritmos de taxa de aprendizado variável. Como originalmente no SGD a taxa de aprendizado é a mesma para todos os parâmetros, isso pode levar problemas em parâmetros que variam pouco ou muito, onde há uma diferença nesses gradientes. O algoritmo Adagrad (DUCHI; HAZAN; SINGER, 2011) foi desenvolvido para solucionar esse problema possuindo uma taxa de aprendizado diferente para cada parâmetro, em cada instante de tempo t.

Seja g_t, i o gradiente da função de custo $J(\theta)$ em relação ao parâmetro θ_i no instante de tempo t. Seja g_t, i regido pela seguinte equação:

$$g_{t,i} = \nabla_\theta J(\theta_i) \tag{2.6}$$

A atualização dos parâmetros originalmente pelo SGD se dá através da seguinte forma:

$$\theta_t + 1, i = \theta_t, i - \mu g_{t,i} \tag{2.7}$$

No algoritmo Adagrad a atualização possui uma taxa de aprendizado μ variável de acordo com o parâmetro.

$$\theta_t + 1, i = \theta_t, i - \frac{\mu}{\sqrt{G_t, ii + \epsilon}} g_{t,i} \tag{2.8}$$

G_t é uma matriz diagonal cujo elemento i, i é a soma dos quadrados dos gradientes com respeito a θ_i até o instante de tempo t. O valor ϵ é um fator colocado no denominador para suavizá-lo (não zerar o denominador). Esse valor pode ser qualquer valor

próximo a zero, como, por exemplo $10^{-}8$. Com isso temos um algoritmo com a taxa sensível a cada parâmetro o que colabora para uma convergência mais rápida. Além disso, algoritmos baseados em momento foram desenvolvidos para acelerar a convergências de algoritmos baseados em SGD. A técnica de momento na otimização do SGD acumula uma média móvel com decaimento exponencial dos gradientes anteriores e continua a mover os parâmetros nessa direção.

Essa técnica é nomeada a partir do conceito de momento da física. O momento pode ser definido através da variável v que é calculada a partir da seguinte equação:

$$v = \alpha v - \epsilon \nabla_\theta (\frac{1}{m} \sum_{i=1}^{m} L(f(x^i; \theta), y^i)) \tag{2.9}$$

$$\theta = \theta + v \tag{2.10}$$

Onde L é a função de erro para o conjunto de dados analisado. Os parâmetros α e ϵ influenciam o quão os gradientes anteriores e o gradiente atual influencia na atualização de parâmetros respectivamente.

2.7 PREPARANDO OS DADOS E FUNÇÃO DE PERDA

Ao preparar as imagens para o treinamento, podemos normalizar as instâncias para que estejam no intervalo de 0 a 1, apenas dividindo pelo valor máximo de um pixel.

Além disso, podemos centralizar os dados em volta do valor zero subtraindo a média de cada pixel. Isso faz com que o pixel com um valor igual à média fique com valor zero e todos diferentes disso em volta do zero.

Durante um processo de treinamento de uma rede neural pode ocorrer um efeito conhecido com sobre-ajuste (overfitting). Isso se dá quando o algoritmo possui um desempenho alto no conjunto de dados de treino, mas não no conjunto de dados de validação. O que isso significa é que o algoritmo especializou o seu conhecimento apenas ao conjunto de treinamento e não generalizou para ser aplicado a outras instâncias, fora do conjunto de treinamento. Existem diversas formas de evitar que isso aconteça e essas técnicas são conhecidas como regularizações.

3 TRABALHOS RELACIONADOS

3.1 COLORIZAÇÃO DE IMAGENS

3.1.1 Colorização de Imagens Vibrantes

No trabalho de (ZHANG; ISOLA; EFROS, 2016) é introduzida uma técnica de colorização automática de imagens, que utiliza o processo de geração de cores plausíveis por uma rede neural profunda. O modelo utiliza grande base de dados para treinar seu modelo. Utilizando o conjunto de imagens ImageNet o algoritmo discutido a seguir foi treinado em mais de um milhão de imagens.

O modelo criado para previsão realiza uma discretização do espaço de cores L*a*b em 313 possíveis valores. A saída da rede consiste em um vetor de 313 valores para cada pixel na imagem, indicando a probabilidade daquela cor. Cada exemplo no conjunto de imagens é codificado em uma distribuição utilizando cada pixel e descobrindo a divisão na discretização em que a cor dele pertence. Dada a divisão atribuída ao pixel então atribui-se o valor 1 a esta e 0 as demais divisões permanecem zero.

Esse trabalho introduz uma métrica de perda diferenciada para o problema da colorização que considera o aspecto multi modal, onde múltiplas soluções possíveis existem para a mesma entrada. A métrica de perda utiliza a função de entropia cruzada multimodal. A equação 3.1 define a função de perda $L_{cl}(\hat{Z}, Z)$, onde $\hat{Z}$ é a saída do algoritmo, Z é a resposta esperada e $v(.)$ é a função que rebalanceia a perda de acordo com a raridade da classe da cor.

$$L_{cl}(\hat{Z}, Z) = -\sum v(Z_{h,w}) \sum Z_{h,w,q} log(\hat{Z}_{h,wq}) \tag{3.1}$$

Podemos ver na figura 9 a comparação entre usando diferentes métodos de geração de cores como regressão (previsão do valor da cor), classificação (probabilidade da cor), classificação com rebalanceamento de pesos e por fim a foto original para comparação. A figura 9 mostra como o resultado é influenciado pela função de perda.

Figura 8 – Comparação de saturação entre imagens utilizando diferentes algoritmos e funções de perda.

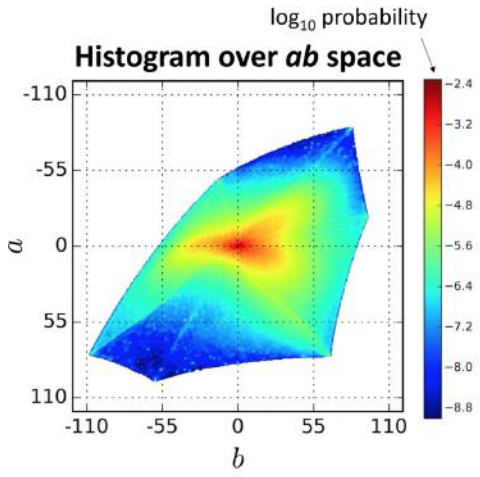


Figura 9 – Distribuição de cores no espaço ab

Além da função de perda diferenciada, foi realizado um rebalanceamento das cores para que a distribuição desigual das cores na base de dados não influencie na previsão realizada pelo modelo. Um histograma da distribuição de cores no espaço definido pode ser visto na Figura 4. Isso faz com que o algoritmo favoreça cores mais raras no conjunto de imagens criando fotos mais vibrantes e menos monocromáticas.

Ao executar o algoritmo para uma nova imagem após gerar a distribuição, o algoritmo utiliza uma função denominada de "annelead-mean" para produzir a cor de cada pixel. A função é dada pela fórmula abaixo:

$$H(Z_h, w) = E[f_T(Z_h, w)], f_T(z) = \frac{exp(log(z)/T)}{\sum_q exp(log(z_q)/T)} \tag{3.2}$$

A solução dos autores para a avaliação da efetividade do algoritmo foi desenvolver um "teste de Turing para colorização". Esse teste consiste em mostrar ao usuário uma figura colorida artificialmente e a original lado a lado. Então é perguntando ao usuário para que ele identifique qual é a original. O algoritmo convence os usuários em 32% dos casos.

Para o regime de treinamento o artigo utilizou imagens de tamanho 64 por 64 em lotes de 40 imagens durante 450 mil iterações. O artigo utilizou a técnica "Adam" (KINGMA; BA, 2015) para otimizar os parâmetros da rede.

3.1.2 Faça-se cor!

No trabalho de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016), um dos principais artigos voltados para o problema de colorização automática de imagens, os autores exploraram o aprendizado profundo para colorização automática de imagens.

A técnica apresentada utiliza o aprendizado de características locais (texturas, segmentos de reta, formas) assim como características globais da imagem (classificação de cena) para um melhor resultado na colorização. O espaço de cores escolhido para re-

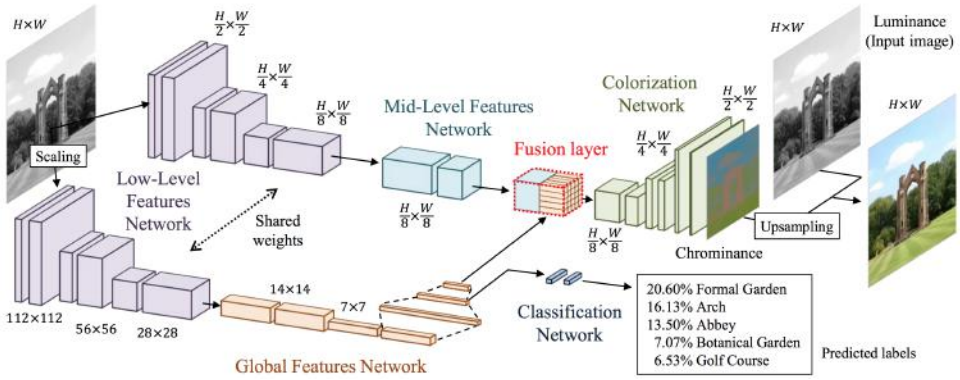


Figura 10 – Arquitetura da rede profunda Faça-se cor!

presentar a imagem foi o CIE L*a*b*. Segundo (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) o espaço de cores CIE L*a*b é o mais adequado para a tarefa de colorização de imagem de acordo com experimentos empíricos. A rede recebe a imagem em apenas uma dimensão em tons de cinza ou apenas o canal de luminância e pode gerar dois outros canais responsáveis pela cor (a* e b*). Foi utilizado a base de dados Places (ZHOU et al., 2014) que possui 2,448,872 imagens para treino e 20,500 para validação.

A rede neural apresentada pode ser dividida em diferentes sub-redes de convoluções: rede de baixo nível, rede de classificação locais, rede de classificação global, combinação de globais e locais, e por último rede de colorização. As sub-redes compõem a rede final. A arquitetura pode ser visualizada na Figura 3

A função de erro a ser minimizada pelo processo de treinamento da rede consiste em uma combinação entre o erro da colorização da imagem (pixel-a-pixel) e o erro de classificação da rede de classificação. Na equação 3.3 descreve a função de erro onde y^{color} é a previsão da cor e y^{color*} é a cor correta. As variáveis y^{class} e $y^{class}_{l^{class}}$ são a classe prevista pelo algoritmo e a classe correta respectivamente.

$$L(y^{color}, y^{class}) = || \, y^{color} - y^{color*} \, || - \alpha(y^{class}_{l^{class}} - log(\sum_{i=0}^{N} exp(y^{class}_i))) \tag{3.3}$$

A forma de avaliação utilizada foi um estudo com pessoas que envolvia a pergunta: “O quão natural essa imagem parece?”. O estudo foi realizado com 10 usuários, a cada um foi mostrado um total de 500 imagens em um total de 1500 imagens avaliadas. O resultado obtido foi 92,6%, em comparação com as imagens originais que obtiveram 97,7%.

3.1.3 Colorização profunda interativa com entradas globais e locais

No trabalho realizado por (XIAO; ZHOU; ZHENG, 2018) observamos uma evolução na habilidade do usuário interagir com a colorização das imagens. Diferente dos outros artigos deste capítulo, este trabalho permite a colorização de imagens não apenas com aprendizado a partir dos dados, mas com interferência do usuário.

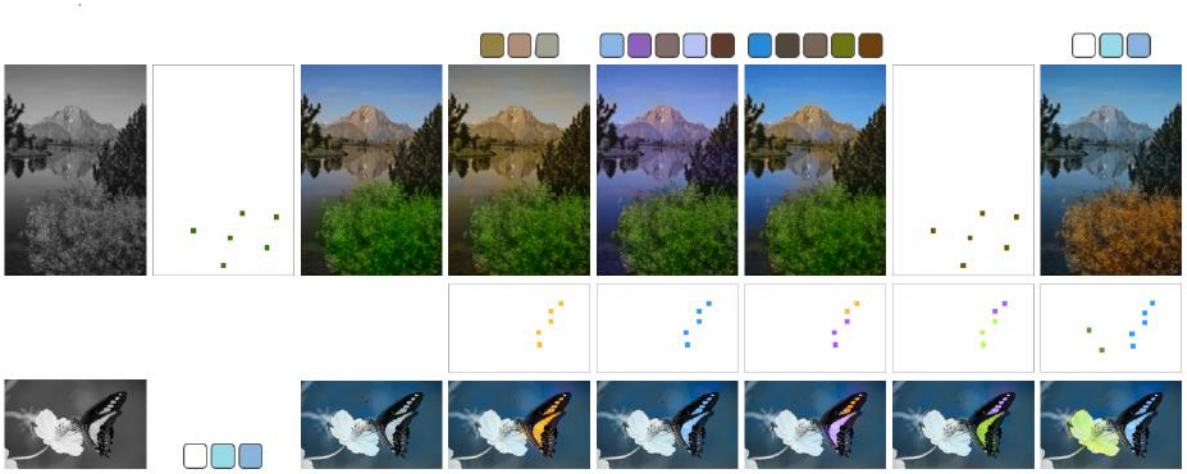


Figura 11 – Exemplo de colorização por (XIAO; ZHOU; ZHENG, 2018). A primeira coluna são as imagens preto e branco, a segunda coluna são as entradas locais na primeira linha e entradas globais na terceira linha. A terceira coluna se trata dos resultados utilizando apenas a entrada local e global, respectivamente. Da quarta coluna em diante são resultados combinando entrada local e global ao mesmo tempo.

A rede neural desenvolvida por (XIAO; ZHOU; ZHENG, 2018) possui três possíveis entradas: imagem preto e branco, entrada global e entrada local. A imagem preto e branco sempre está presente, mas as outras duas entradas são opcionais. A entrada global consiste em uma paleta de cores de até 7 cores para servirem como um tema de colorização. Já a entrada local se trata de pontos específicos na imagem em preto e branco onde o usuário indica as cores daqueles pontos, a partir daí a rede deve colorir com consistência utilizando essa informação.

A arquitetura da rede é definida em formato de "U"como indicado na Figura 12. Ela pode ser separada em 4 grandes módulos: extração de características, entrada global, fusão e reconstrução.

Ao se preparar a função de erro para o problema precisa-se levar em consideração as diferentes entradas do algoritmo. Existem 4 possibilidades de entrada: apenas a imagem preto e branco, imagem com entradas locais, imagem com a entradas globais e imagem com entradas globais e locais. Uma função de perda levando em consideração cada um desses casos foi elaborada para extrair a melhor performance do modelo. Definir um método capaz de levar em consideração tantas entradas diferentes, pode ser um desafio que foi atacado por esse trabalho. A ferramenta final criada é capaz de facilitar muito a colorização provendo tanto ferramentas complexas de detalhes como temas mais abrangentes de cores a serem aplicados, juntamente com a recomendação baseada nos dados.

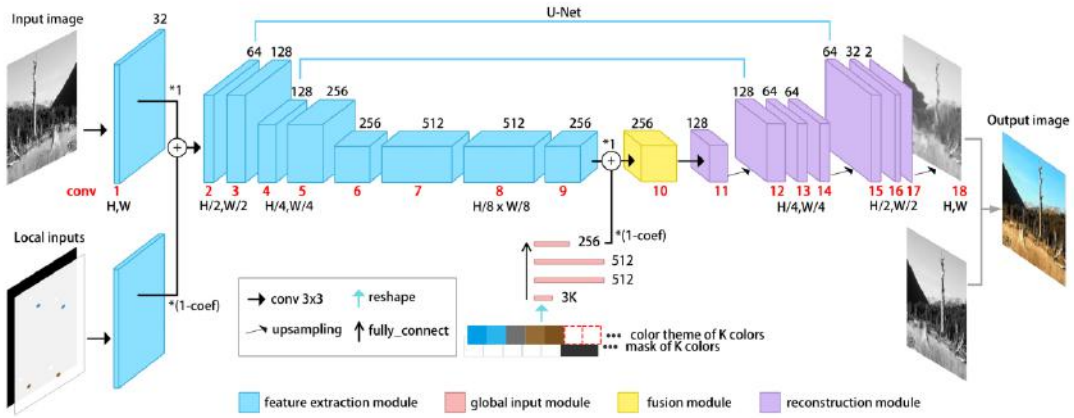


Figura 12 – Arquitetura da rede desenvolvida por (XIAO; ZHOU; ZHENG, 2018).

4 ABORDAGEM

4.1 PROPOSTA INICIAL

A proposta inicial desde trabalho foi reproduzir um modelo mais simples da rede desenvolvida por (ZHANG; ISOLA; EFROS, 2016), para verificar se poderíamos obter os mesmos resultados. O objetivo desta reprodução foi que ela servisse de base para possíveis melhorias. Após validação inicial, o modelo implementado evoluiu para tal qual descrito neste capítulo. Não foi utilizado nenhum código do trabalho original e apenas a inspiração do modelo, através de seu artigo.

Foram utilizadas as bibliotecas Keras e Tensorflow (ABADI et al., 2016) para a implementação do modelo. A biblioteca Keras permite a descrição do modelo via código e seu treinamento. A biblioteca Tensorflow é responsável pela computação numérica e é utilizada pela biblioteca Keras. Nas próximas seções entraremos em detalhes sobre o modelo desenvolvido.

4.2 BASE DE DADOS E TRATAMENTO DOS DADOS

A base de dados utilizada no projeto é a Places do MIT (https://www.mturk.com/). Essa base de dados é constituída de 2,5 milhões de imagens classificadas em 205 classes de cenas. Para construir a base de dados os autores utilizaram 696 adjetivos comuns da língua inglesa em combinação com o nome da categoria da cena para compor uma busca. Essa busca foi enviada aos principais buscadores de imagens (Google Images, Bing Images e Flickr). Foi utilizada a plataforma Amazon Mechanical Turk (https://www.mturk.com/) para classificar as imagens. Essa plataforma permite que autores paguem um valor para que pessoas entrem e respondam a certas perguntas. Nesse caso para cada imagem foi exigido que o usuário confirmasse, se esta pertencia à categoria da busca que a gerou. Para o treinamento da rede neste trabalho, não foi utilizado a informação da classe das imagens. O tamanho original de cada imagem na base de dados é de 256 $\times$ 256 pixeis.

Neste trabalho foi utilizado o tamanho inicial de imagem de 64 $\times$ 64 pixeis para viabilizar um treinamento em tempo hábil. Foram utilizadas 2,3 milhões de imagens para realizar o treinamento. Além disso, na arquitetura do artigo de (ZHANG; ISOLA; EFROS, 2016) foi feito uma subamostragem das imagens sendo a saída uma imagem 2 vezes menor. A base de dados foi alterada para verificar se o modelo poderia se adaptar a uma diferente. Dado que o modelo de (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) obteve um bom desempenho utilizando essa base de dados, então o trabalho de (ZHANG; ISOLA; EFROS, 2016) também deveria ser capaz de utilizar esses dados para aprendizado. Essa base de dados também foi utilizada por (XIAO; ZHOU; ZHENG, 2018).

A entrada do algoritmo consiste em uma imagem preto e branco, que é obtida extraindo o canal L do espaço de cor L*a*b. A imagem possui valores entre 0 e 255 e deve ser normalizada em uma escala menor para acelerar o treinamento da rede neural. Isso evita também que existam valores muitos altos na rede neural, o que pode afetar o cálculo e o uso dos gradientes. A normalização aplicada considera a média e desvio padrão da base de dados da seguinte maneira:

$$\hat{x} = \frac{x - E(X)}{std(X)} \tag{4.1}$$

Onde $\hat{x}$ é o dado utilizado durante treinamento, $E(X)$ é a média da base dados e $std(X)$ é o desvio padrão da base dados.

4.3 REGIME DE TREINAMENTO

A máquina utilizada no treinamento foi a Tesla, que é uma Microway Xeon + Tesla GPU Tower Workstation. A placa gráfica utilizada para treinamento do algoritmo é NVIDIA Tesla K80 "Kepler" M-class GPU Accelerator. Para o treinamento foi utilizada a técnica de otimização de lotes, onde a rede é treinada sempre utilizando um lote de dados por vez, já que a base não pode ser completamente mantida em memória. O tamanho do lote foi de 30 imagens e com esses dados foram ajustados os parâmetros da rede. Este trabalho utilizou imagens de 64×64 e lotes de 30 exemplos por vez. Cada treinamento demorava em média 16 horas para processar todos os dados contidos na base. O modelo atual foi treinado durante 198 mil iterações de 30 imagens.

O algoritmo escolhido para realizar a otimização da rede neural foi o Adam por (KINGMA; BA, 2015) (Adaptive Momentum Estimation, ou Estimativa de Momento Adaptativa), baseado na técnica de descida de gradiente estocástica. Já o método escolhido foi estocástico devido a limitações em relação a manter os dados em memória. A taxa de aprendizado utilizada foi $3e-5$ com β_1 de 0.9, β_2 de 0.99 e decaimento de $1e-3$ que são parâmetros do algoritmo de otimização.

4.4 ARQUITETURA DA REDE

A seguir vamos abordar os blocos constituintes da rede neural reproduzida neste trabalho. A rede é constituída por diferentes camadas aplicadas uma após a outra. As camadas utilizadas foram: convolução, normalização de lote, ativação, super-amostragem e softmax. A seguir entraremos em detalhes sobre como cada uma dessas camadas funciona e como foram aplicadas na arquitetura da rede neural.

4.4.1 Convolução

A camada de convolução explicada no capítulo 2 foi aplicada com tamanho de 3 por 3. Isso significa que cada convolução possui 9 valores a serem otimizados em todas as camadas. Redes neurais de convolução foram demonstradas capazes de extrair informações semânticas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) o que é essencial para o problema de colorização.

A maioria das camadas da rede mantém o tamanho de entrada para a saída. Apenas em casos em que há a diminuição do tamanho intencional, onde isso é alcançado utilizando a técnica de transposição ("strides"). Transposição é como movemos a convolução pela imagem. Se a camada possui uma transposição de 1 então significa que as aplicações da convolução são realizadas pixel a pixel, movendo 1 por vez. Se a camada possui transposição de 2 significa que aplicamos a um pixel e depois movemos 2 pixeis para o lado, ativamente pulando um pixel. Dessa maneira a imagem final é reduzida em altura e largura pela metade, ignorando certas etapas em aplicação do filtro de convolução.

Toda camada de convolução é seguida por uma função de ativação, que introduz não linearidade na rede neural. A função ativação utilizada por este trabalho é a RELU que será descrita a seguir.

4.4.2 Ativação RELU

A função de ativação utilizada após cada camada de convolução ou normalização de lote foi a RELU (Rectified Linear Unit). A função RELU é definida pela seguinte fórmula:

$$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{4.2}$$

Essa ativação é comumente utilizada quando tratando-se de redes neurais convolucionais. Mais notoriamente foi utilizada por (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

4.4.3 Normalização de lote

A camada de normalização de lote, criada por (IOFFE; SZEGEDY, 2015), é constituída de uma normalização da entrada da camada utilizando a média e desvio padrão do lote a ser processado. Isso colabora com a velocidade do treinamento tornando o valor de cada elemento com variância 1 após essa camada. Dado que cada entrada de uma camada são características de uma camada anterior, unificar essas características em uma única escala garante que o algoritmo possa aprender a importância de cada características sem

ser influenciada pela sua escala. O cálculo dos valores se dá pela seguinte fórmula, onde x_k é o lote a ser processado durante cada etapa do treinamento:

$$\hat{x}_k = \frac{x_k - E(x_k)}{\sqrt{Var(x_k)} + \epsilon} \tag{4.3}$$

O termo definido por $E(x_k)$ significa a média do lote enquanto $Var(x_k)$ significa a variância. Além disso, dois parâmetros são treinados com o algoritmo. São eles γ e β para cada camada. Abaixo podemos ver a fórmula que define a saída da camada dado o cálculo acima.

$$y_k = \gamma \hat{x}_k + \beta_k \tag{4.4}$$

O termo ϵ foi adicionado para ter estabilidade numérica em caso de $\sqrt{Var(x_k)}$ seja zero. A camada de normalização de lote foi aplicada em algumas etapas da arquitetura como descrito na seção "Arquitetura em camadas".

4.4.4 Super-amostragem

A super amostragem é realizada para aumentar as dimensões de largura e altura de uma imagem. Nesse contexto é utilizada para aumentar os dados durante o processamento da rede neural. Isso é feito porque o processo de passagem pela rede neural diminui as dimensões espaciais da imagem devido à utilização de “strides” como explicado na seção sobre camadas de convolução.

O algoritmo utiliza a estratégia de proximidade para definir os valores a serem utilizados no novo tamanho. Isso significa que aumentado o tamanho da imagem e posicionando os pixeis novos entre os antigos, dependendo da proximidade com os antigos, estes possuíram os novos valores do pixel mais próximo. Esse processo pode ser visualizado na figura 7.

4.4.5 Softmax

A função “softmax” é a operação que realizamos para transformar resultados numéricos em probabilidades. Probabilidades só podem possuir valores de 0 a 1. A saída da rede neural possui valores numéricos diversos com as operações realizadas. A função é definida pela seguinte fórmula:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))} \tag{4.5}$$

Essa operação é aplicada a cada pixel da saída. Como a saída da rede neural é no formato $LxAxQ$ onde L é a largura, A é a altura e Q é a quantidade de faixas da discretização. A operação é aplicada em cada pixel por toda a sua discretização para transformar cada discretização na probabilidade da cor estar naquela discretização.

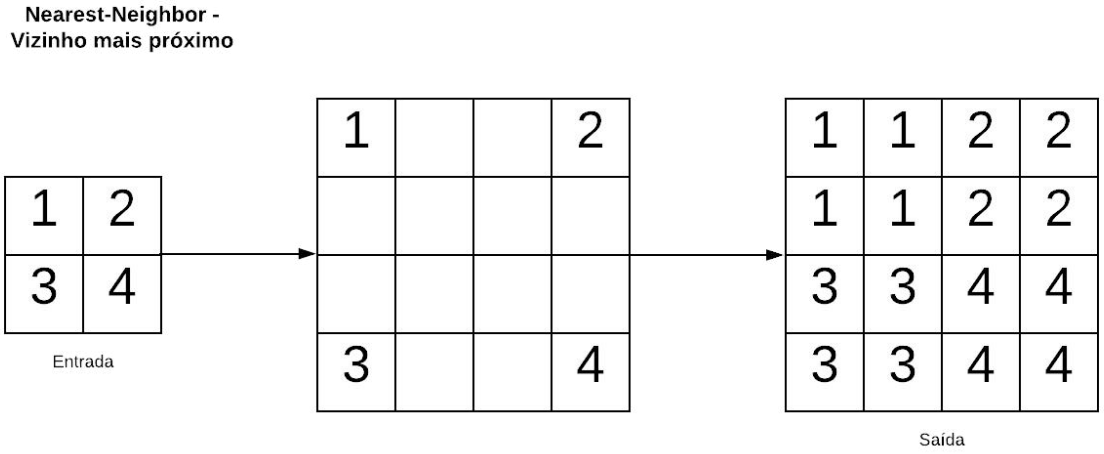


Figura 13 – Exemplo do processo de super-amostragem para uma imagem utilizando a estratégia do vizinho mais próximo.

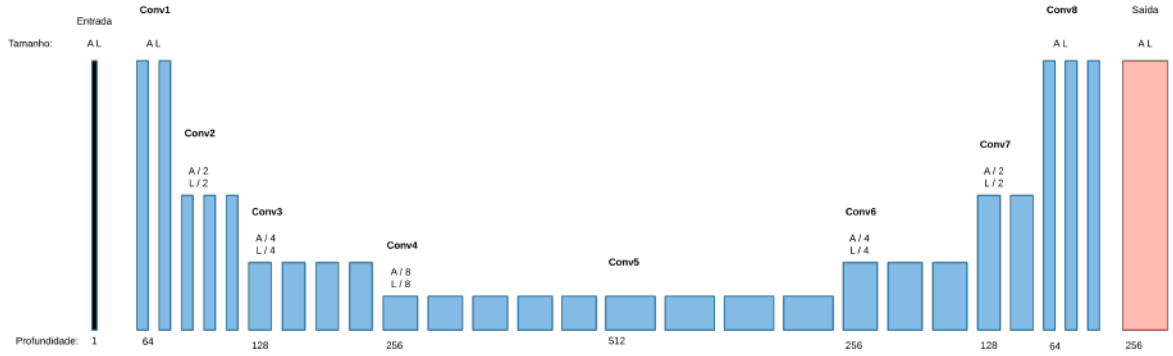
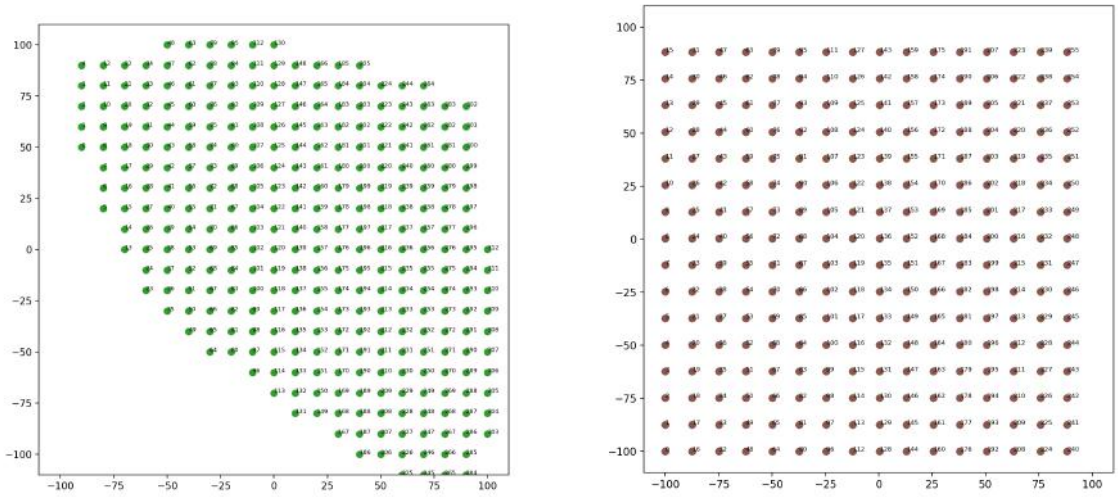


Figura 14 – Arquitetura da rede neural desenvolvida nesta monografia. Cada bloco significa uma convolução seguida por uma ativação RELU e em alguns casos uma normalização de lote. A altura do bloco significa o tamanho da imagem e a largura significa a profundida de cada cama. Toda vez que há uma mudança de tamanho é feita utilizando a sub-amostragem ou super-amostragem.

4.4.6 Arquitetura em camadas

Combinando os blocos descritos nas seções anteriores este trabalho montou a seguinte arquitetura como mostrado na figura. Outra visualização é fornecida pela Figura 14 onde podemos enxergar as camadas que constituem a rede neural e sua ordem. A arquitetura possui 20 camadas de convolução e 6 camadas de normalização de lote. A rede total possui 16.946.304 parâmetros treináveis. A arquitetura pode ser visualizada na figura 14.



(4.6)

Figura 15 – A esquerda: discretização com valores definidos a priori de (ZHANG; ISOLA; EFROS, 2016). A direita: discretização linear subdividindo o espaço de cores AB.

4.4.7 Saída da rede

A técnica descrita por (ZHANG; ISOLA; EFROS, 2016) de discretização foi adaptada para simplificação de implementação. O espaço de discretização de 131 possíveis valores descritos por (ZHANG; ISOLA; EFROS, 2016) foi adaptado para um espaço de 256 valores. Isso aumenta o tamanho da distribuição final do algoritmo e possui mais cores para representar a imagem. Entretanto, o cálculo dos espaços a serem divididos foi apenas dividir o espaço possível de cores em 16 faixas, diferente do cálculo realizado por (ZHANG; ISOLA; EFROS, 2016).

O processo de discretização consiste em transformar um vetor de cor, constituído pelos canais AB, em uma distribuição de cores com N valores possíveis. Cada valor vai possuir 1.0 caso seja aquela cor ou 0.0, caso não seja. O vetor de cor AB é um vetor com duas dimensões e valores que variam de -100 a 100.

Por exemplo, no artigo de (ZHANG; ISOLA; EFROS, 2016) foram definidas 313 subdivisões para a distribuição, ou seja, o valor N é 313. Dada um valor a_i e b_i a função de discretização vai determinar o índice no vetor final y em que este vai ser preenchido com o valor 1.0 enquanto os outros serão preenchidos com 0.0.

Neste trabalho a função de discretização subdivide cada dimensão de valores de 0 a 255 em 16 espaços e dado um valora a_i e b_i contido em um desses espaços então é atribuído um índice como pode ser visto na figura abaixo.

Figura 16 – Comparação entre as funções de transformação de probabilidades em cor. Da esquerda para direita: imagem preto e branco, função esperança, probabilidade máxima, média das 3 cores de maior probabilidade, original.

A discretização implementada foge da discretização utilizando conhecimentos a priori sobre a distribuição das cores da base de dados feita por (ZHANG; ISOLA; EFROS, 2016), que era utilizada como referencial para se aplicar um algoritmo de vizinho mais próximo para definir a discretização dos pixeis. Dada um valor a_i e b_i determina o ponto mais próximo dele, nos valores definidos a priori e é dado para tal o índice atribuído a esse valor mais próximo. Os valores definidos por (ZHANG; ISOLA; EFROS, 2016) podem ser vistos na figura 17.

4.5 TRANSFORMANDO DISTRIBUIÇÃO DE PROBABILIDADES EM CORES

Como última etapa para gerar a colorização da imagem, precisamos converter a distribuição de probabilidades em valores numéricos para as cores, no espaço de cores L*a*b. Existem algumas formas para se fazer isso e para o modelo foi experimentado as seguintes: probabilidade máxima, esperança da distribuição e média das 3 maiores probabilidades. A operação de esperança é definida pela multiplicação da probabilidade de cada cor pelo seu valor numérico, definido de acordo com a discretização. Ao final todos os valores são somados e obtém-se a cor resultante. A probabilidade máxima a operação mais simples que converte a posição de maior probabilidade em sua cor. Já a média das 3 maiores é um híbrido entre as duas operações em que as 3 maiores probabilidades são combinadas utilizando a esperança apenas sobre essas cores, ou seja, multiplicando sua cor pela probabilidade e fazendo a média entre elas.

A figura 16 podemos observar o resultado aplicando as operações definidas. Nos experimentos realizados a operação de esperança foi a que obteve resultados mais consistentes e suavizados.

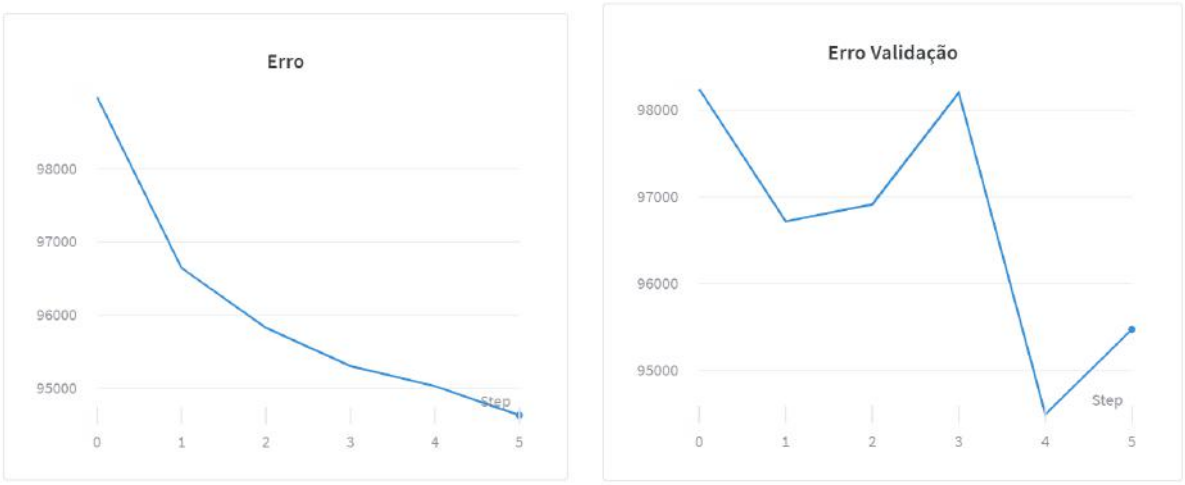


Figura 17 – A esquerda: função de erro durante as épocas de treinamento. A direita: função de erro no conjunto de validação

4.6 FUNÇÃO DE ERRO

A função implementada para o modelo é a da soma dos mínimos quadrados da diferença. A função se dá pela fórmula:

$$y = \sum_{a,l,q} (x_{verdadeiro} - x_{previsto})^2 \tag{4.7}$$

A utilização de outras funções como da entropia cruzada foram utilizados sem resultados significativos. Devido à distribuição de cores do conjunto de dados outras funções se mostraram impotentes em determinar bons erros para serem otimizados.

A função de erro implementada por (ZHANG; ISOLA; EFROS, 2016) dependia de ter calculado a distribuição de cores para cada segmentação dentro da discretização de cores. Ao implementar uma função de erro MSE este trabalho pode ser capaz de treinar uma rede eficiente mesmo não possuindo essa distribuição calculada.

Dada as mudanças implementadas no algoritmo, foi necessário adaptar os principais parâmetros da rede neural. Com a mudança da função de erro a taxa de aprendizado foi modificada de 0.00003 para 0.00001, o que é uma diferença muito pequena. Não houveram mudanças nos hiper-parâmetros além dessa.

4.7 SEGMENTAÇÃO DE ELEMENTOS

Ao treinar um algoritmo de visão, podemos verificar que o modelo deve ser capaz de realizar certas tarefas para atingir o resultado. Uma das sub-tarefas associadas ao problema da colorização se trata da segmentação da imagem. Para que o algoritmo seja capaz de colorir corretamente este deve conseguir classificar de formas diferentes regiões da imagem que possuem elementos distintos. Como podemos observar na figura 23 o modelo

Figura 18 – Exemplo de colorização de uma imagem. A esquerda: mapa de colorização. Centro: colorização do modelo. Direita: cores originais da imagem.

distinguiu corretamente os objetos da cena como a estrutura metálica de centro, apesar de não acertar em sua cor, o chão e o céu. Esse foi um dos resultados iniciais importante para identificar que o modelo estava generalizando de acordo com a situação da imagem. Apesar de termos exemplos que demonstram como a segmentação foi alcançada, também possuímos diversas falhas que mostram que não foi atingido resultados de estado da arte.

4.8 VISUALIZANDO A DISTRIBUIÇÃO DE CORES PREVISTA

Podemos analisar o resultado de uma colorização do ponto de vista de apenas uma cor, dentre as 256 cores possíveis. Escolhida uma cor podemos verificar qual a sua probabilidade para cada pixel. Dada a saída do modelo, acessamos a posição N para obter a probabilidade da cor de número N daquele pixel. Para toda imagem então obtém-se um mapeamento onde maior o número, maior a probabilidade da cor escolhida ser prevista. Na figura 23 podemos ver um exemplo de colorização de imagem. Na figura 24 podemos verificar o mapa de probabilidade para o verde, como descrito previamente neste parágrafo. Em seguida na figura 25 podemos ver o mapa de probabilidade para múltiplas cores e por fim, na figura 22 vemos a completa representação de todos os mapas de probabilidade possíveis.

Após fazer essa visualização para uma cor, podemos realizar esse mesmo processo para todas as cores. Isso nos dará uma visualização completa da extensão de probabilidades da cor da colorização. Ao enxergar a distribuição de todas as cores, podemos entender que o modelo não prevê uma distribuição variada.

Figura 19 – Exemplo de colorização para visualização das probabilidades. A esquerda: imagem preto e branco. Centro: colorização do modelo. Direita: cores originais da imagem.

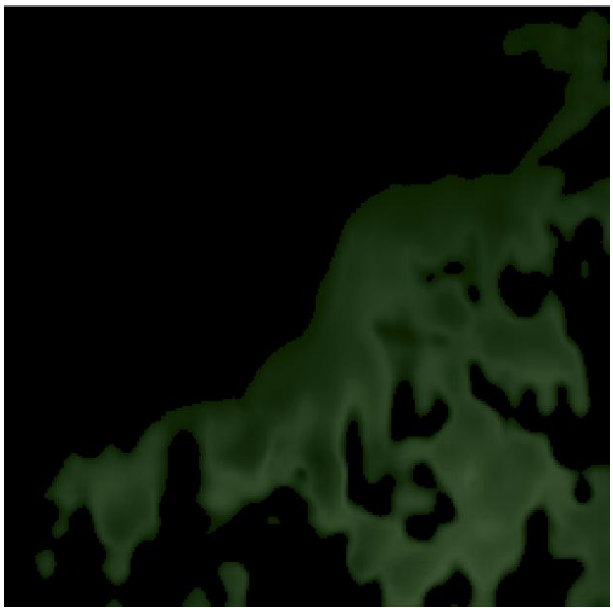


Figura 20 – Probabilidades da cor verde para a imagem exemplo da figura 23. Quanto mais clara a cor, maior a probabilidade daquele pixel conter aquela cor.

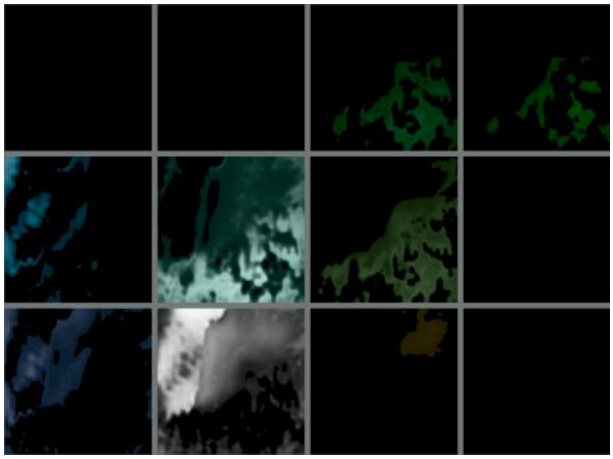


Figura 21 – Probabilidades de múltiplas cores para a imagem exemplo da figura 23.

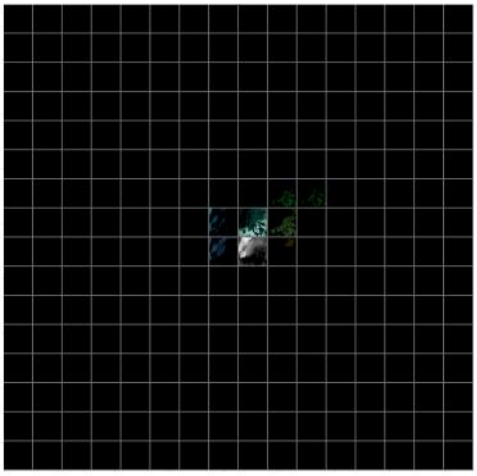


Figura 22 – Probabilidades para toda a imagem exemplo da figura 23 de todas as cores possíveis.

5 RESULTADOS

5.1 ACURÁCIA

É difícil analisar o resultado de uma colorização, devido as diversas possibilidades que uma imagem pode ser colorida. Existem diversas colorizações plausíveis para uma imagem, como, por exemplo, diversos tons de verde podem ser usados para colorir uma vegetação. Se utilizarmos apenas a diferença entre a cor original e a gerada podemos considerar um exemplo de plausível como incorreto.

Definir um limite aceitável entre a cor original e gerada, para que o pixel possa ser considerado correto, é algo arbitrário. Entretanto, podemos utilizar essa métrica para entender, superficialmente, se o modelo está colorindo corretamente. Seja a_g e b_g as cores geradas pela rede e a_o e b_o as originais, então a cor estará correta se: $\sqrt{\sum (a_g - a_o)^2 + (b_g - b_o)^2}$ for menor que o limite t. Esta mesma métrica foi utilizada por (ZHANG; ISOLA; EFROS, 2016) para calcular a função de erro cumulativa, variando-se a variável t de 0 até 150.

Na figura 23 podemos ver um exemplo de imagem colorida e original. Na parte inferior podemos ver o gráfico da acurácia, quantidade de pixeis certos, dividida pela quantidade total de pixeis, conforme o limite t varia de 0 até 150, assim como em (ZHANG; ISOLA; EFROS, 2016).

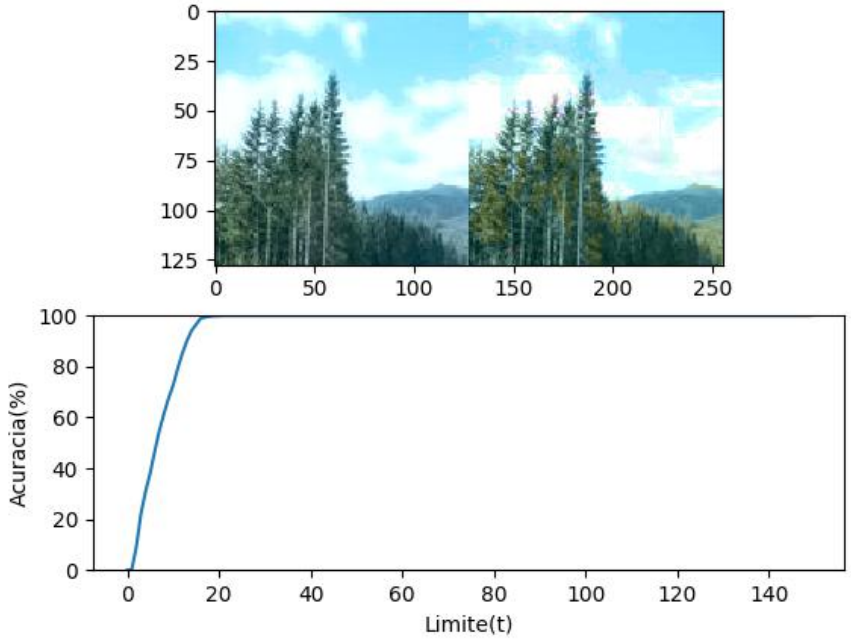


Figura 23 – Evolução da acurácia de acordo com um limite(t). Imagem colorida pelo modelo à esquerda e original à direita. Área sobre a curva: 94.97%

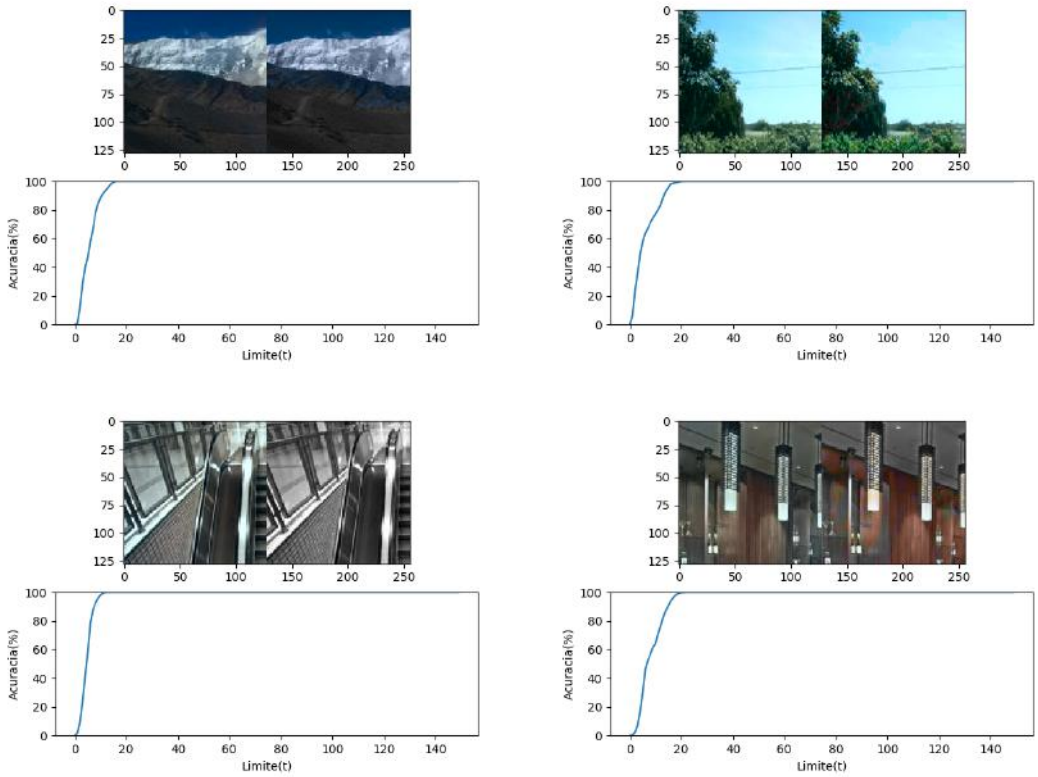


Figura 24 – Exemplos de alto valor no gráfico de acurácia acumulada por limite de erro. Imagem colorida pelo modelo à esquerda e original à direita.

Utilizando-se de uma base de validação de 1000 exemplos podemos averiguar a área sobre a curva de acurácia. Essa métrica é calculada somando-se o valor de acurácia para cada limite t na curva cumulativa de erro, dividida pele área total. Estes exemplos não foram fornecidos ao modelo durante a fase de treinamento descrita no capítulo 4.O resultado obtido para área sobre a curva na base de validação foi de 92.64%.

Observando os exemplos de baixa acurácia na figura 25 podemos ver que é possível ter uma imagem colorida corretamente, avaliando-se de forma subjetiva, entretanto ter a acurácia baixa. No exemplos de alta acurácia na figura 24 podemos ver casos simples que possuem alta acurácia mesmo sem ter uma colorização significativa.

Utilizando-se da mesma métrica para analisar os resultados de (ZHANG; ISOLA; EFROS, 2016) foi obtido um resultado médio, no mesmo conjunto de imagens, de 91.55%.

5.2 EXEMPLOS DE RESULTADOS

Foi possível verificar que o modelo é reproduzível em outras bases de dados do que os aplicados no artigo de (ZHANG; ISOLA; EFROS, 2016). As mudanças no algoritmo não causaram quedas de performance. Apesar das diferenças na quantidade de iterações, que foi de 450 mil no artigo de (ZHANG; ISOLA; EFROS, 2016) para 198 mil nesse trabalho,

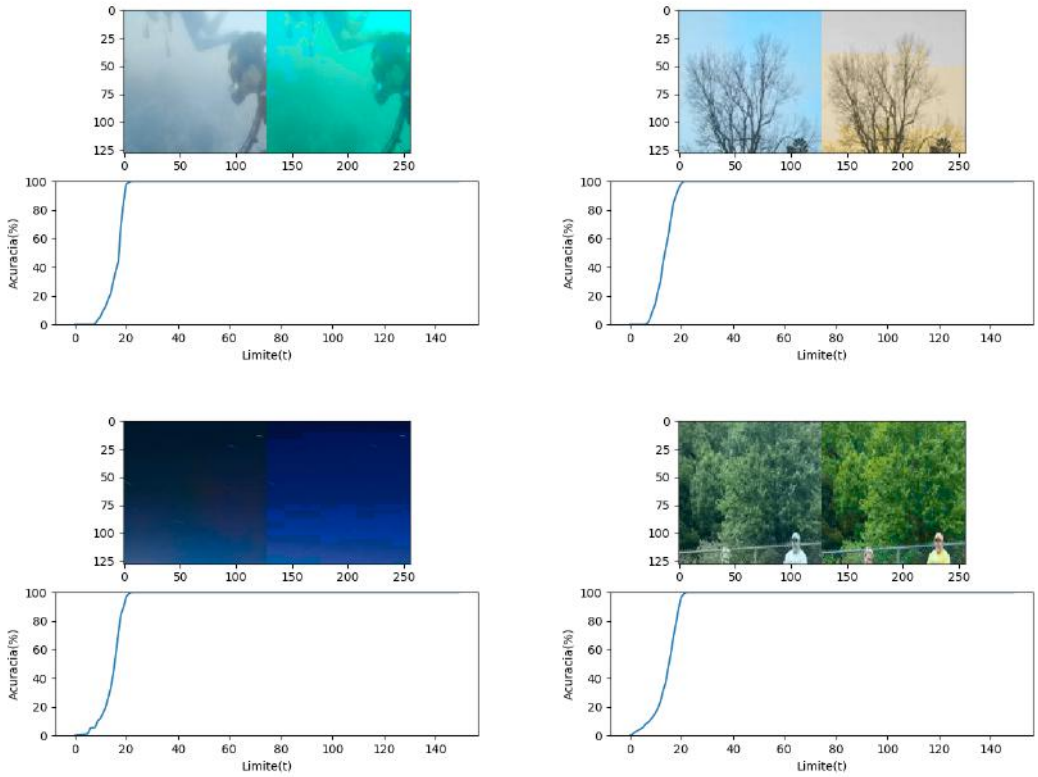


Figura 25 – Exemplos de baixo valor no gráfico de acurácia acumulada por limite de erro. Imagem colorida pelo modelo à esquerda e original à direita.

e da diferença de tamanho, de imagens de 224×224 para 64×64, podemos verificar que a rede neural foi capaz de performar bem em novas imagens.

Como podemos ver nas figuras 26 e 27 a rede conseguiu aprender padrões como vegetações, céus e outros artefatos naturais. Outros como pessoas ou elementos mais incomuns como aviões não foram coloridos corretamente.

5.3 CONCLUSÃO

Utilizando a métrica de avaliação de área da curva da função de erro cumulativa conseguimos comparar o resultado deste trabalho com um das referências ((ZHANG; ISOLA; EFROS, 2016)). No conjunto de teste de 1000 exemplos de validação, o resultado de 92.64% aproximou-se do obtido por (ZHANG; ISOLA; EFROS, 2016) de 91.55%.

A métrica de avaliação utilizada neste trabalho foi incapaz de identificar corretamente exemplos plausíveis de colorização. Essa dificuldade foi comum nos trabalhos relacionados. Devido essa dificuldade tanto os trabalhos de (ZHANG; ISOLA; EFROS, 2016) e (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016) definiram métricas subjetivas com avaliações humanas. Testes com seres humanos foram utilizados para avaliar a qualidade da colorização, tanto o quão natural se pareciam e sua comparação com as imagens originais.

Figura 26 – Resultados da colorização realizada pela rede neural. A imagem a esquerda é o mapa de cores, a do meio se trata da colorização da rede neural e a direita a imagem original.

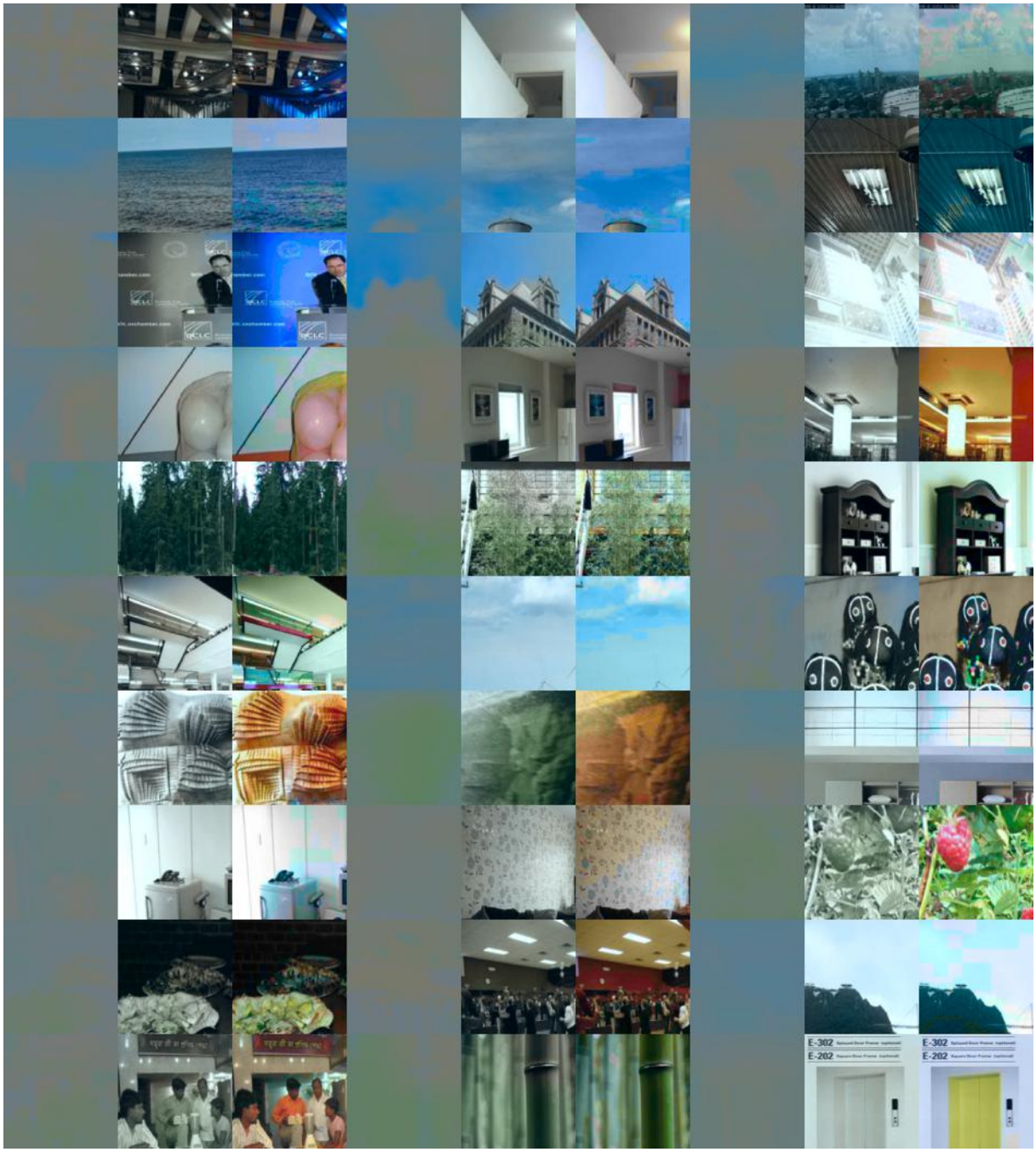


Figura 27 – Resultados da colorização realizada pela rede neural. A imagem a esquerda é o mapa de cores, a do meio se trata da colorização da rede neural e a direita a imagem original.

6 CONCLUSÃO

Uma das maiores dificuldades em se treinar uma rede neural desse porte é a utilização de grandes conjuntos de dados. Experimentos com menos que 1 milhão de imagens se mostraram ineficientes e insignificativos. Mesmo utilizando de imagens com tamanhos reduzidos como 64 por 64 tínhamos um período de treinamento de em torno de 16 horas para uma época. Obter a base de dados também se mostrou um desafio para manter e manipular esses dados. A base utilizada possui 2,4 milhões de imagens. Cada operação a ser feita em cima desses dados pode demorar muitas horas de processamento mesmo utilizando-se uma máquina potente como a utilizada neste trabalho.

Podemos constatar o poder dos algoritmos e da quantidade massiva de dados que são introduzidos a estes. Mesmo em condições diversas o algoritmo foi capaz de atacar o problema e extrair dados semânticos significativos para o problema da colorização. Tal problema passa pelo problema do entendimento da cena na imagem para que seja possível realizar a colorização de forma adequada. Entretanto, sem o tempo de treinamento maior e com imagens menores o algoritmo não foi capaz de obter a mesma performance que das fontes consultas, o que demonstra que há espaço para otimização em relação ao processo de treinamento da rede. Dado o avanço da performance com o treinamento, podemos inferir que com mais tempo e mais dados poderíamos alcançar resultados melhores.

6.1 TRABALHOS FUTUROS

Devidos as dificuldades descritas no capítulo de resultados, podemos considerar um método de avaliação diferente para este trabalho. Assim como realizado por (ZHANG; ISOLA; EFROS, 2016) podemos conceber um experimento com pessoas para avaliar a qualidade da colorização em comparação com as imagens originais, perguntando-as qual imagem parece ser a correta. Podemos também considerar a avaliação realizada por (IIZUKA; SIMO-SERRA; ISHIKAWA, 2016): perguntar o quão natural a imagem parece ser.

Atualmente os modelos apresentados ficam inacessíveis para os públicos não técnicos. Isso pode ser resolvido com a criação de aplicações web ou mobile que possam aplicar os modelos de forma fácil e prática. Desafios surgem dessas aplicações como a distribuição do modelo e performance ao rodar em aparelhos móveis.

REFERÊNCIAS

ABADI, M. et al. Tensorflow: A system for large-scale machine learning. **ArXiv**, abs/1605.08695, 2016. Disponível em: http://arxiv.org/abs/1605.08695. Acesso em: 30 ago. 2020.

DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of Machine Learning Research**, v. 12, n. Jul, p. 2121–2159, 2011.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. **ArXiv**, 2016. Disponível em: https://arxiv.org/abs/1603.07285. Acesso em: 30 ago. 2020.

GAMA, J. et al. **Inteligência artificial**: uma abordagem de aprendizado de máquina. [S.l.]: Rio de Janeiro: LTC, 2011.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge: MIT Press, 2016. Disponível em: http://www.deeplearningbook.org. Acesso em: 30 ago. 2020.

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**, v. 35, n. 4, 2016.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. **ArXiv**, abs/1502.03167, 2015. Disponível em: http://arxiv.org/abs/1502.03167. Acesso em: 30 ago. 2020.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: **International Conference on Learning Representations, 3.** Conference Track Proceedings, 2015. (ICLR). Disponível em: https://arxiv.org/abs/1412.6980. Acesso em: 30 ago. 2020.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. Acesso em: 30 ago. 2020.

LARSSON, G.; MAIRE, M.; SHAKHNAROVICH, G. Learning representations for automatic colorization. **ArXiv**, abs/1603.06668, 2016. Disponível em: http://arxiv.org/abs/1603.06668. Acesso em: 30 ago. 2020.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural Computation**, MIT Press, Cambridge, MA, USA, v. 1, n. 4, p. 541–551, dez. 1989. ISSN 0899-7667. Disponível em: http://dx.doi.org/10.1162/neco.1989.1.4.541. Acesso em: 30 ago. 2020.

LEVIN, A.; LISCHINSKI, D.; WEISS, Y. Colorization using optimization. **ACM Trans. Graph.**, ACM, New York, NY, USA, v. 23, n. 3, p. 689–694, ago. 2004.

LUAN, Q. et al. Natural image colorization. In: **Proceedings of the 18th Eurographics Conference on Rendering Techniques**. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007. p. 309–320. Disponível em: http://dx.doi.org/10.2312/EGWR/EGSR07/309-320. Acesso em: 30 ago. 2020.

MITCHELL, T. M. **Machine Learning**. [S.l.]: New York: McGraw-Hill Science/Engineering/Math, 1997.

XIAO, Y.; ZHOU, P.; ZHENG, Y. Interactive deep colorization with simultaneous global and local inputs. **ArXiv**, abs/1801.09083, 2018. Disponível em: http://arxiv.org/abs/1801.09083. Acesso em: 30 ago. 2020.

ZHANG, R.; ISOLA, P.; EFROS, A. A. Colorful image colorization. **ArXiv**, abs/1603.08511, 2016. Disponível em: http://arxiv.org/abs/1603.08511. Acesso em: 30 ago. 2020.

ZHOU, B. et al. Learning deep features for scene recognition using places database. In: GHAHRAMANI, Z. et al. (Ed.). **Advances in Neural Information Processing Systems 27**. [S.l.]: Curran Associates, Inc., 2014. p. 487–495. Disponível em: http://papers.nips.cc/paper/5349-learning-deep-features-for-scene-recognition-using-places-database.pdf. Acesso em: 30 ago. 2020.