# Colorização de imagens com Deep Learning

**Giovana de Lucca, Elloá B. Guedes**

[1]Núcleo de Computação
Escola Superior de Tecnologia
Universidade do Estado do Amazonas (UEA)
Manaus – AM – Brasil

gol.eng@uea.edu.br, ebgcosta@uea.edu.br

## 1. Introdução

As técnicas de Aprendizado de Máquina (AM) têm sido aplicadas com sucesso em um grande número de problemas reais em diversos domínios. A principal razão deste sucesso é decorrente da natureza inferencial e da boa capacidade de generalização dos métodos e técnicas desta área, cuja ideia central consiste em utilizar algoritmos capazes de aprender padrões por meio de exemplos, baseando-se apenas em dados previamente disponíveis (FACELI et al., 2011; GULLI; PAL, 2017).

A Visão Computacional (VC), por sua vez, é uma área que procura desenvolver métodos capazes de replicar nos computadores as capacidades da visão humana. O reconhecimento de imagens é uma parte integrante do processo de VC e aplica as técnicas de processamento digital de imagens para extrair as características desejadas (KHAN et al., 2018). Antes da difusão das técnicas de AM, o procedimento de extração de características de imagens envolvia um grande esforço devido ao processamento e à aplicação de uma variedade de métodos matemáticos. Em algumas situações, por exemplo, até mesmo análises de especialistas eram utilizadas para descobrir as regras necessárias para o reconhecimento e extração de certos padrões (FACELI et al., 2011).

Devido ao progresso significativo no campo da VC e da tecnologia de sensores visuais, VC e AM desempenharam juntos papéis decisivos no desenvolvimento de uma variedade de aplicações baseadas em processamento digital de imagens (KHAN et al., 2018). Por exemplo, o reconhecimento de dígitos manuscritos proposto pelo *dataset* MNIST (*Modified National Institute of Standards and Technology*) foi uma das aplicações pioneiras que congregou VC e AM, difundida até os dias atuais para fins didáticos de métodos e técnicas deste domínio (MNIST, 2018). Atualmente, em particular, um dos projetos emergentes é o ImageNet, o qual disponibiliza um grande banco de imagens projetado para tarefas de VC e propõe anualmente um desafio chamado *ImageNet Large Scale Visual Recognition Challenge*, que visa elencar os melhores algoritmos em âmbito mundial para detecção de objetos e classificação de imagens em larga escala (IMAGENET, 2018).

Nas últimas décadas, com a crescente complexidade dos problemas a serem tratados computacionalmente e diante do grande volume de dados constantemente gerados por diferentes setores, tornou-se clara a necessidade de ferramentas, algoritmos, métodos e técnicas computacionais mais sofisticados para endereçar estas questões (FACELI et al., 2011). Embora alguns algoritmos de AM já existissem há bastante tempo, a demanda por aplicar automaticamente cálculos matemáticos complexos em uma grande escala de dados tornou-se uma necessidade particularmente mais recente. Embora tenha havido

um aumento do poder de processamento e armazenamento pelos dispositivos computacionais disponíveis, especialmente via Computação em Nuvem, novas estratégias de AM precisavam ser desenvolvidas ou evoluírem a partir de técnicas já existentes. Neste sentido, as técnicas de *Deep Learning* (DL) emergiram, principalmente compreendendo as redes neurais com uma grande quantidade de camadas ocultas e operações convolucionais (KHAN et al., 2018).

Entende-se por DL, um grande conjunto de camadas de redes neurais artificiais cuja principal característica é aprender representações a partir de dados (BUDUMA, 2017; CHOLLET, 2017). As três principais vantagens oferecidas pelo DL são a simplicidade da construção das redes, a escalabilidade, por lidar com um grande volume de dados, e a transferência de conhecimento (do inglês, *Transfer Learning* – TL), pois um modelo treinado para uma determinada tarefa pode ser adaptado para outras tarefas relacionadas (KHAN et al., 2018).

Considerando este cenário e almejando a aplicação destes conceitos de vanguarda, a proposta do presente trabalho de conclusão de curso visa explorar as arquiteturas canônicas de redes neurais convolucionais, utilizando as técnicas de DL e TL, para endereçar o problema da colorização artificial de imagens. Neste problema, uma imagem em tons de cinza, a exemplo de uma imagem histórica, é apresentada à uma rede neural convolucional que, como resposta, propõe uma versão colorida da mesma. Essa colorização deve ser plausível e realista ao ponto de não possuir inconsistência visual quando analisada por pessoas comuns.

Ao longo desta introdução serão mostrados os demais elementos que compõem este trabalho. A Seção 1.1 contempla os objetivos propostos para o desenvolvimento do projeto. Na Seção 1.2 são apresentadas as justificativas que motivam a realização do trabalho em questão. A metodologia adotada é detalhada na Seção 1.3. Por fim, a Seção 1.4 compreende o cronograma das atividades, seguido da Seção 1.5 que dispõe a organização do restante do documento.

## 1.1. Objetivos

O objetivo geral deste trabalho consiste em explorar estratégias para colorização artificial de imagens utilizando técnicas de *Deep Learning*. Para tanto, faz-se necessário elencar alguns objetivos específicos, descritos a seguir:

1. Consolidar uma base de dados representativa de imagens coloridas para treinamento das redes;
2. Descrever o problema da colorização artificial de imagens segundo uma tarefa de Aprendizado de Máquina;
3. Explorar a utilização das arquiteturas canônicas de redes neurais convolucionais mediante *Transfer Learning* aplicadas ao problema considerado;
4. Propor, treinar e testar diferentes redes neurais convolucionais baseadas nas arquiteturas canônicas elencadas;
5. Analisar os resultados obtidos de maneira quantitativa e qualitativa.

## 1.2. Justificativa

Imagens em escala de cinzas retêm informações que podem ser importantes em diversos aspectos. A colorização de fotos em arquivos antigos pode agregar algum valor aos seus

respectivos contextos históricos e artísticos. Algum detalhe de uma imagem em tons de cinza talvez possua outra interpretação se esta mesma imagem estivesse colorizada. Seguindo o mesmo raciocínio, a coloração das imagens de câmeras de segurança com baixa resolução pode influenciar nas interpretações das filmagens, permitindo, por exemplo, a identificação mais apropriada de indivíduos presentes nestas imagens.s

Na área da Saúde, por exemplo, colorizações podem ser ajustadas e modificadas para restaurar algum tipo de perturbação visual de indivíduos, como é o caso do daltonismo. As técnicas de colorização artificial a serem exploradas neste trabalho podem, por exemplo, possibilitar representações visuais mais adequadas para os portadores deste tipo de agravo.

Além do que foi exposto, a proposta considerada neste trabalho incentiva a prática de conceitos, técnicas e tecnologias de uma área emergente da Computação, contribuindo na formação profissional da aluna concluinte. No mais, esta proposta de trabalho de conclusão de curso está alinhada com as atividades desenvolvidas pelo *Laboratório de Sistemas Inteligentes* (LSI), uma iniciativa promovida por docentes do Núcleo de Computação (NUCOMP) da Escola Superior de Tecnologia (EST) da Universidade do Estado do Amazonas (UEA), motivando o desenvolvimento de soluções inovadoras que utilizam técnicas emergentes do Aprendizado de Máquina.

### 1.3. Metodologia

Para atingir os objetivos propostos no escopo deste trabalho, a condução das atividades obedece à metodologia apresentada a seguir, composta dos seguintes passos:

1. Estudo dos conceitos relacionados à *Machine Learning*, *Deep Learning* e as principais arquiteturas de redes neurais convolucionais;
2. Estudo do ferramental tecnológico para elaboração e execução de projetos de *Deep Learning*, incluindo Python, Keras, Sci-kit Learn, Google Cloud Platform, dentre outros;
3. Elaborar uma base de dados representativa de imagens coloridas e em escalas de cinza para fins de aprendizado dos padrões de coloração pelas redes neurais convolucionais;
4. Elencar um conjunto de arquiteturas canônicas das redes neurais convolucionais aplicáveis ao problema em questão;
5. Propor modificações nas redes neurais identificadas no passo anterior mediante *Transfer Learning*;
6. Treinar as redes modificadas com os exemplos da base de dados;
7. Testas as redes e coletar métricas de desempenho;
8. Analisar os resultados obtidos identificando as redes mais adequadas ao cenário considerado;
9. Escrita da proposta do Trabalho de Conclusão de Curso;
10. Defesa da proposta do Trabalho de Conclusão de Curso;
11. Escrita do Trabalho de Conclusão de Curso;
12. Defesa do Trabalho de Conclusão de Curso.

### 1.4. Cronograma

Uma visão geral do cronograma de atividades a serem desenvolvidas ao longo deste trabalho pode ser vista na Tabela 1. Essas atividades possuem relação com as atividades listadas na Seção 1.3, a qual detalha a metodologia que este trabalho deverá seguir.

**Tabela 1: Cronograma de atividades levando em consideração os onze meses (02/2018 a 12/2018) para realização do presente trabalho de conclusão de curso.**

| | 2018 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | **10** | **11** | **12** |
| **Estudo dos conceitos teóricos relacionados à *Machine Learning*** | x | x | x | x | | | | | | | |
| **Estudo do ferramental tecnológico para elaboração do projeto** | x | x | x | x | | | | | | | |
| **Consolidação da base de dados** | | | x | x | | | | | | | |
| **Especificação das arquiteturas canônicas de redes neurais convolucionais** | | | x | | | | | | | | |
| **Aplicação das técnidas de *Transfer Learning* nas redes neurais convolucionais identificadas** | | | | x | | | | | | | |
| **Treinamento das redes neurais convolucionais com os exemplos da base de dados** | | | | x | x | x | x | | | | |
| **Testes das redes e comparação de métricas de desempenho** | | | | | x | x | x | x | | | |
| **Análise dos resultados obtidos** | | | | | | | | | x | x | x |
| **Escrita da proposta do trabalho** | x | x | x | x | x | | | | | | |
| **Defesa da proposta do trabalho** | | | | | x | | | | | | |
| **Escrita do trabalho final** | | | | | | x | x | x | x | x | x |
| **Defesa do trabalho final** | | | | | | | | | | | x |

### 1.5. Organização do Documento

Para apresentar a proposta do presente trabalho de conclusão de curso, este documento está organizado como segue. A Seção 2 contempla os fundamentos teóricos necessários para elaboração do projeto, incluindo conceitos de *Machine Learning*, *Deep Learning* e as principais arquiteturas de redes neurais convolucionais. A Seção 3 discorre sobre os trabalhos relacionados. A solução proposta para o tema de colorização de imagens utilizando técnicas de *Deep Learning* é abordada na Seção 4. Por fim, as considerações parciais do trabalho encontram-se na Seção 5.

## 2. Fundamentação Teórica

Nesta seção serão apresentados os fundamentos teóricos que dão suporte à realização deste trabalho. As redes neurais artificias, seus principais conceitos, arquiteturas, métodos de aprendizagem e algumas aplicações são discorridas na Seção 2.1. Os conceitos elementares sobre as técnicas de *Machine Learning* conhecidas como *Deep Learning* encontram-se na Seção 2.2. Na Seção 2.2.1 são descritas as características das redes neurais utilizadas pela técnica de *Deep Learning*, as Redes Neurais Convolucionais. Por fim, as tecnologias utilizadas para a realização deste trabalho são apresentadas na Seção 2.4.

### 2.1. Redes Neurais Artificiais

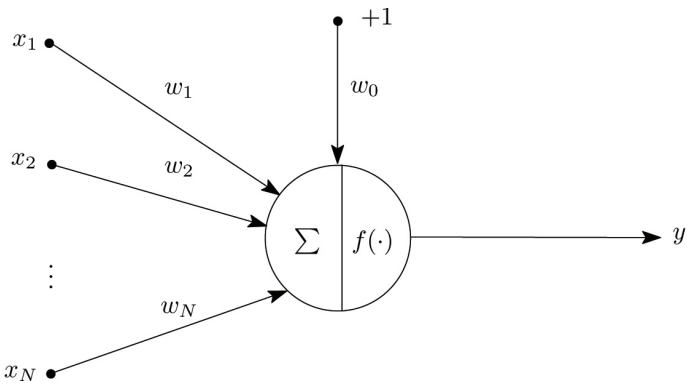
As *Redes Neurais Artificiais* (RNAs) são modelos computacionais inspirados na capacidade de processamento de informações do cérebro humano (ROJAS, 1996). De acordo com esta ideia, as RNAs possuem unidades de processamento simples, denominadas

*neurônios artificiais*, dispostos em camadas interconectadas por ligações associadas a coeficientes numéricos, chamados *pesos* (FACELI et al., 2011). As RNAs são capazes de aprenderem padrões complexos a partir dos dados e prever resultados para exemplos não conhecidos, o que demonstra a sua capacidade de generalização (HAYKIN, 2009).

O neurônio artificial é a unidade fundamental na construção de RNAs, tendo sido inspirado no seu análogo biológico. Segundo Rosenblatt, existe um conjunto de $m$ entradas, equivalentes aos dendritos de um neurônio biológico, por onde os sinais são introduzidos (ROSENBLAT, 1961). Associa-se um peso a cada entrada, representando a relevância referente a uma conexão sináptica. Há também o peso $w_0$, um termo de polarização criado com a intenção de estabelecer um limiar de ativação para cada neurônio. Este peso corresponde à entrada *bias*, cujo valor é sempre unitário. Pode-se então definir um vetor de entradas $X = [+1, x_1, x_2, \ldots, x_m]$ e um vetor de pesos $W = [w_0, w_1, \ldots, w_m]$. As entradas e pesos são combinados por meio de uma função $\phi : \mathbb{R}^{m+1} \to \mathbb{R}$, que é geralmente a soma ponderada das entradas e pesos, conforme Equação 1. Este modelo de neurônio encontra-se ilustrado na Figura 1 (LIMA, 2016).

$$\phi(X, W) = \sum_{i=0}^{m} x_i \cdot w_i. \tag{1}$$

**Figura 1: Neurônio artificial: a combinação linear das entradas *x* ponderadas pelos pesos *w* é transformada pela função de ativação *f* na saída *y*.**

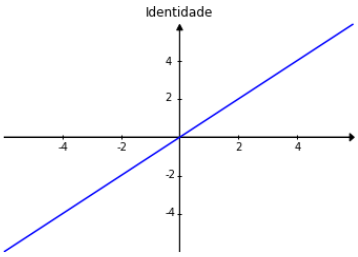



A função $f$ é chamada de *função de ativação* e fornece a resposta de um neurônio para uma dada entrada. Esta função precisa ser monotônica e contínua, podendo comumente ser as funções identidade, sigmóide, tangente hiperbólica, ou a retificada linear (ReLU) (DELICATO; PIRES; SILVEIRA, 2017). Estas funções encontram-se representadas na Figura 2.

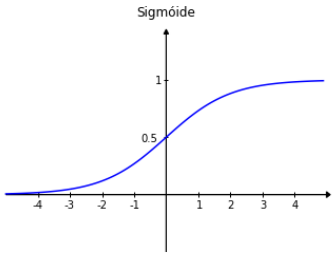
Neurônios artificiais individuais têm uma capacidade computacional limitada, independentemente da função de ativação escolhida, pois resolvem apenas problemas linearmente separáveis. No entanto, um conjunto de neurônios artificiais conectados na forma de uma rede – *rede neural artificial* – adquirem a capacidade de resolver problemas de elevada complexidade (BRAGA; CARVALHO; LUDERMIR, 2007). A alternativa mais utilizada para resolver estes problemas é distribuir os neurônios em uma ou mais camadas conhecidas como camadas ocultas (FACELI et al., 2011). Segundo Cybenko, uma rede com uma camada oculta pode implementar qualquer função contínua e uma rede com

**Figura 2: Exemplos de diferentes funções de ativação.**

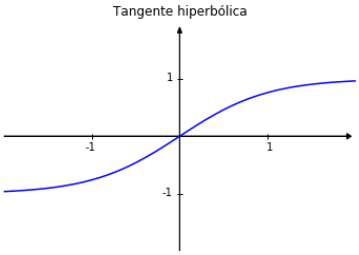
**(a) Função identidade.**



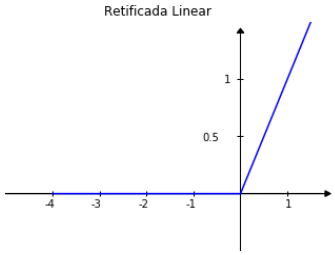

**(b) Função sigmóide.**




**(c) Função tangente hiperbólica.**




**(d) Função retificada linear.**




duas camadas ocultas permite a aproximação de qualquer função (CYBENKO, 1989).

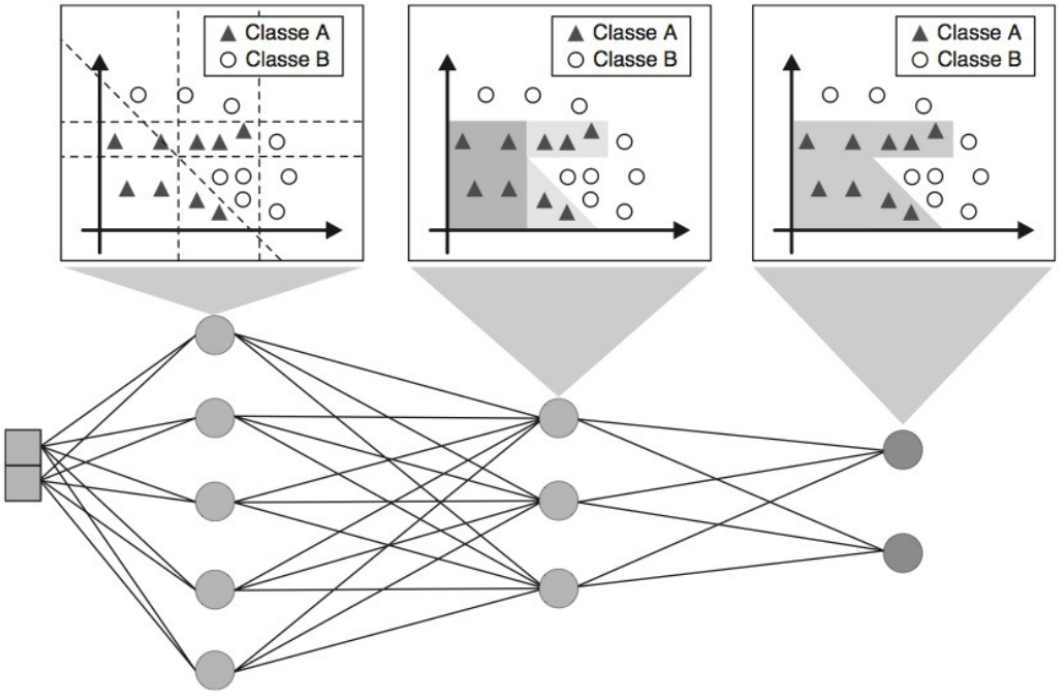
As RNAs do tipo Perceptron Multicamadas (MLP, do inglês *Multilayer Perceptron*) apresentam uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Uma das principais características de uma rede MLP é o seu alto grau de conectividade entre os neurônios, cuja intensidade está associada aos pesos da rede.

Cada neurônio em uma rede MLP atua ponderando as entradas recebidas dos neurônios de uma camada anterior a ele conectados, produzindo como saída um valor, resultante de sua função de ativação, que é propagado às camadas seguintes da rede neural. Conforme exemplificado na Figura 3, a combinação das atuações individuais desempenhadas por cada neurônio da rede que define a atuação associada à RNA como um todo (BRAGA; CARVALHO; LUDERMIR, 2007; FACELI et al., 2011; HAYKIN, 2009).

Uma importante característica das RNAs é a sua capacidade de aprender por meio de exemplos. O processo de *aprendizado* de uma rede neural consiste em sucessivos ajustes de pesos associados aos seus neurônios, de modo a aprimorar seu desempenho de acordo com um critério pré-estabelecido. Tais ajustes são realizados por algoritmos de treinamento formados por um conjunto de regras bem definidas que especificam quando e como deve ser alterado o valor de cada peso. Diversos algoritmos de aprendizado foram propostos, dentre os quais se destacam aqueles que seguem o paradigma de *aprendizado supervisionado* (FACELI et al., 2011; LIMA, 2016).

O aprendizado supervisionado ajusta os pesos aplicando um conjunto de exemplos de treinamento rotulados. Cada exemplo consiste em um sinal de entrada associado à sua resposta alvo desejada. A cada padrão de entrada submetido à rede, compara-se a resposta

**Figura 3: Papel desempenhado pelos neurônios das diferentes camadas de uma rede MLP. Fonte: (FACELI et al., 2011).**




desejada com a resposta calculada, ajustando-se os pesos das conexões para minimizar o erro (HAYKIN, 2009).

O algoritmo mais utilizado para o treinamento de redes MLP é o algoritmo *back-propagation*, também chamado de retropropagação do erro ou ainda regra delta generalizada. Este algoritmo respeita o aprendizado supervisionado em que os pesos são modificados e ajustados para reduzir a distância entre a resposta desejada e a resposta produzida pela rede (HAYKIN, 2009). O treinamento é constituído da iteração de duas fases, uma fase para frente (*forward*) e uma fase para trás (*backwards*) (FACELI et al., 2011). A fase *forward*, que compreende o fluxo da informação a partir da entrada até a saída da rede, é utilizada para produzir uma saída para um dado sinal de entrada. A fase *backwards*, com fluxo da informação da saída da rede em direção à entrada, utiliza a diferença entre as saídas desejada e produzida para atualizar os pesos das conexões entre os neurônios e assim minimizar o erro (BRAGA; CARVALHO; LUDERMIR, 2007). Os ciclos de apresentação dos dados de treinamento e eventuais ajustes de pesos no *backpropagation* são iterados até que seja atingido um critério de parada como, por exemplo, um número máximo de ciclos ou uma taxa máxima de erro (FACELI et al., 2011).

As RNAs são modelos computacionais com ampla aplicação na resolução de problemas de previsão. Algumas aplicações utilizam RNAs para predição de condições climáticas, como em (SOUSA et al., 2017) e (ARAUJO et al., 2017), em que se deseja prever precipitações de chuva de determinados locais. Diversas outras aplicações de RNAs dizem respeito à classificação de padrões. Dentre as aplicações na área de classificação financeira, uma das mais bem sucedidas é a análise de crédito (BRAGA; CARVALHO; LUDERMIR, 2007). Além disso, as RNAs também são muito utilizadas

para diagnósticos médicos, como em (SILVA et al., 2016) e (PEREIRA et al., 2016). Outras aplicações tratam de reconhecimento de caracteres (CARVALHO, 2006), robótica (TINOS, 1999), jogos (CAEXETA, 2008), comunicação (SILVA et al., 2017), dentre outros.

## 2.2. *Deep Learning*

*Deep Learning* (DL), ou Aprendizado Profundo, é uma subárea do AM especialmente baseada na utilização de RNAs com uma grande quantidade de camadas e neurônios para aprender padrões complexos em um vasto volume de dados (CHOLLET, 2017; KHAN et al., 2018; GULLI; PAL, 2017). Por meio do reconhecimento de padrões, os modelos baseados em DL são capazes de reconhecer, traduzir, sintetizar e até prever padrões das mais diferentes naturezas (DELICATO; PIRES; SILVEIRA, 2017).

As técnicas de DL têm sido aplicadas com êxito em muitos problemas, especialmente considerando dados de alta dimensionalidade, a exemplo de imagens e vídeos, e contextos em que há uma grande disponibilidade de exemplos (DELICATO; PIRES; SILVEIRA, 2017; KHAN et al., 2018). Os modelos de DL têm se destacado, por exemplo, em muitas aplicações de saúde, especialmente considerando a detecção automática de padrões em imagens médicas para fins diagnósticos (LU et al., 2017). O desafio *ImageNet Large Scale Visual Recognition Challenge*, de caráter anual realizado desde 2010, também têm promovido a proposição e competição de modelos de vanguada para fins de detecção de objetos e classificação de imagens em larga escala, contribuindo para o desenvolvimento do estado da arte em VC (IMAGENET, 2018).

Os modelos e técnicas de DL têm sido aplicados em tarefas de aprendizado supervisionado e não supervisionado, em que as redes neurais convolucionais têm sido o modelo mais proeminente. A seção a seguir apresenta o detalhamento deste modelo, suas características e conceitos associados.

### 2.2.1. Redes Neurais Convolucionais

As *Redes Neurais Convolucionais* (CNNs, do inglês *Convolutional Neural Networks*) são modelos de redes neurais *feedforward* com muitas camadas ocultas, em que cada camada possui uma finalidade específica e implementa uma determinada funcionalidade básica, como convolução, normalização, *pooling*, etc (GOODFELLOW; BENGIO; COURVILLE, 2016; KHAN et al., 2018).

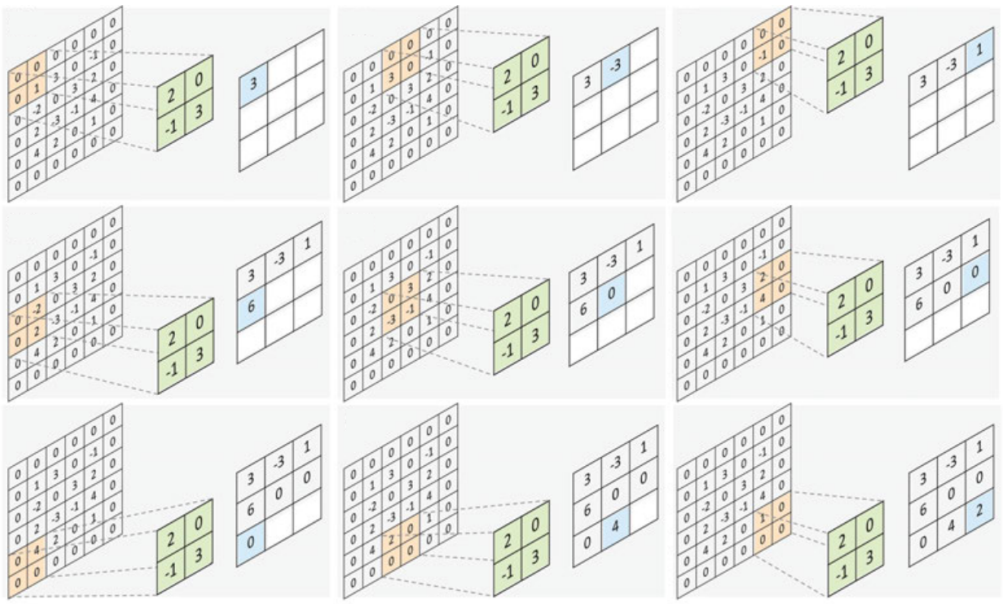
O funcionamento das CNNs reside especificamente na realização de operações de convolução. A *convolução* é uma operação linear que calcula a soma dos produtos de toda a extensão de duas entradas em função de um determinado deslocamento, tendo como principal objetivo a extração de características da entrada por meio de um filtro, produzindo um *mapa de características* (GOODFELLOW; BENGIO; COURVILLE, 2016; SEWAK; KARIM; PUJARI, 2018).

No contexto das CNNS, uma camada convolucional recebe um volume de entrada de $n$ dimensões e pode possuir um preenchimento $p$ de zeros (*zero-padding*), aplicado ao redor da entrada. Essa entrada é processada por $k$ filtros que representam os pesos e as conexões da CNN (KHAN et al., 2018). Cada filtro consiste em uma matriz de números

discretos e possui uma extensão espacial $e$, que é igual ao valor da altura e da largura do filtro, e um *stride* $s$, que é a distância entre as aplicações de convolução consecutivas do filtro no volume de entrada (BUDUMA, 2017).

A Figura 4 exemplifica o processo de convolução aplicado em uma entrada de tamanho $5 \times 5$ e *zero-padding* $p = 1$. O filtro utilizado no exemplo possui extensão espacial $e = 2$ e *stride* $s = 2$ com inicialização de pesos aleatória. É possível observar que o mapa de características é uma saída de tamanho $3 \times 3$ em que cada um dos seus componentes é a da soma das multiplicações dos elementos do filtro com os elementos de um segmento da entrada (KHAN et al., 2018).

**Figura 4: Exemplo de um processo de convolução aplicado em uma entrada de tamanho $5 \times 5$ e um filtro de tamanho $2 \times 2$. Fonte: (KHAN et al., 2018).**

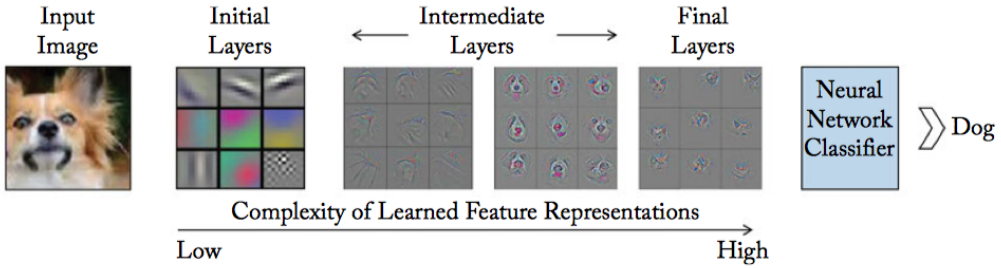


No contexto de VC, em especial, em que as CNNs recebem imagens como entrada, as camadas de convolução atuam reconhecendo características visuais. As primeiras camadas convolucionais iniciam reconhecendo características mais simples, como linhas e bordas. Porém, à medida que a camada convolucional é mais profunda e que a entrada já passou por mapas de características mais elementares, é possível reconhecer características cada vez mais complexas, como texturas, objetos e até rostos, conforme ilustrado na Figura 5.

Considerando a grande dimensionalidade dos dados, é preciso manter um número razoável de parâmetros ajustáveis, evitando uma sobrecarga de processamento. Assim, visando diminuir a dimensão dos mapas de características, tipicametne alternam-se camadas convolucionais e camadas de *pooling*. Uma camada de *pooling* é responsável por dividir o mapa de características recebido como entrada em blocos de tamanhos iguais, processando cada bloco para criar um mapa de características condensado. O processamento dos blocos é definido por uma função de *pool* que pode ser, por exemplo, a função máximo, caracterizando o *max-pooling* (BUDUMA, 2017; KHAN et al., 2018).

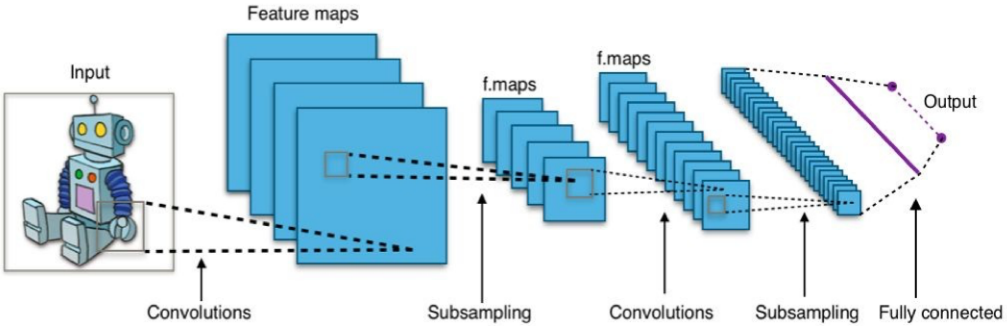
As últimas camadas de uma CNN normalmente são as *Fully Connected Layers*

**Figura 5: Exemplo de mapas de características em uma CNN aplicados a um problema de classificação de imagens. Fonte: (KHAN et al., 2018).**




(FCL), ou camadas completamente conectadas, as quais correspondem essencialmente às camadas de convolução que utilizam filtros de tamanho $1\times$ 1 em que cada elemento é densamente conectado à todos os elementos da camada anterior (KHAN et al., 2018). Nesta última camada, adota-se tipicamente a função de ativação *softmax*, também chamada de função exponencialmente normalizada, a qual escala o vetor de saída em um vetor de probabilidades, recurso muito útil em problemas de classificação (DELICATO; PIRES; SILVEIRA, 2017). O processo de aprendizagem utilizado pelas FCL é baseado nos conceitos das RNAs do tipo MLP e no algoritmo de treinamento *backpropagation* (GULLI; PAL, 2017; KHAN et al., 2018). A Figura 6 sintetiza a ideia da disposição das camadas em uma CNN.

**Figura 6: Disposição de camadas em uma CNN. Fonte: (GULLI; PAL, 2017).**



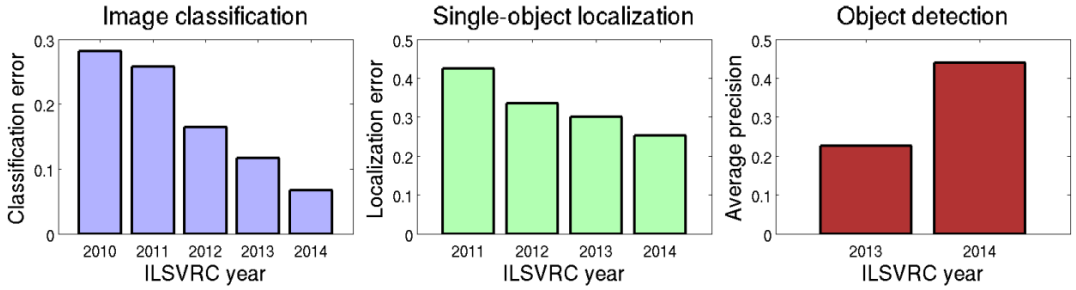

Durante o processo de treinamento, as CNNs utilizam técnicas de regularização para evitar possíveis erros de generalização, tais como o *overfiting* (GOODFELLOW; BENGIO; COURVILLE, 2016). Uma das abordagens mais populares para isto consiste na técnica de *Dropout*, a qual temporariamente evita a propagação de alguns pesos da rede mediante uma probabilidade $p$. Intuitivamente, a existência de *Dropout* força a CNN a ser acurada mesmo na ausência de certa informação (BUDUMA, 2017). A utilização de camadas do tipo *Dropout* impacta positivamente de forma significativa no desempenho das CNNs na fase de testes, prevendo a saída associada a exemplos previamente não conhecidos (KHAN et al., 2018).

### 2.2.2. Arquiteturas Canônicas de Redes Neurais Convolucionais

Como mencionado anteriormente, o desafio anual *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) têm tido um papel protagonista no desenvolvimento de soluções em DL, pois têm promovido um contexto para proposição e comparação de algumas das arquiteturas de CNNs mais bem sucedidas para problemas de detecção de objetos e classificação de imagens em larga escala (RUSSAKOVSKY et al., 2015).

Entre os anos de 2010 e 2014, os resultados alcançados pelo ILSVRC melhoraram significativamente nas três categorias existentes. Os desempenhos alcançados neste cenário são mostrados no gráfico da Figura 7 de acordo com a métrica utilizada em cada categoria (RUSSAKOVSKY et al., 2015).

**Figura 7: Desempenhos alcançados do ILSVRC entre os anos de 2010 e 2014.**

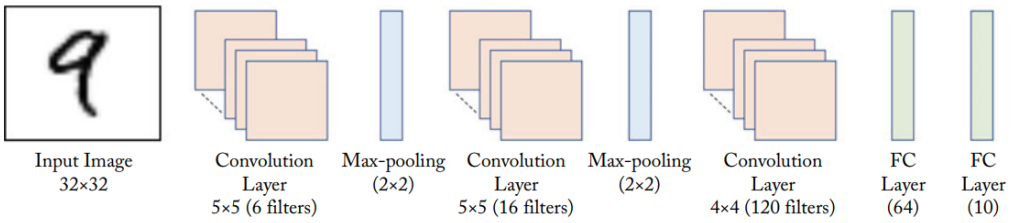



Embora os conceitos das camadas de uma CNN estejam bem estabelecidos e sejam de conhecimento geral, nem sempre é uma tarefa fácil propor uma rede neural deste tipo para um determinado cenário. Assim, uma consequência positiva da realização do ILSVRC é promover a difusão das arquiteturas de destaque na competição, as quais passam ser conhecidas e adaptadas pela comunidade acadêmica e tecnológica na resolução de diversos outros problemas. Considerando esta importância e potencial de aproveitamento de soluções, a seguir são apresentadas algumas destas arquiteturas canônicas.

**LeNet** Yann le Cun desenvolveu, em 1990, uma das primeiras arquiteturas utilizadas para o reconhecimento de dígitos manuscritos, a LeNet. Vencedora do ILSVRC 2010, esta arquitetura é composta por três camadas convolucionais alternadas com camadas de *pooling* seguidas de duas FCLs conforme representado na Figura 8 (SEWAK; KARIM; PUJARI, 2018; KHAN et al., 2018).

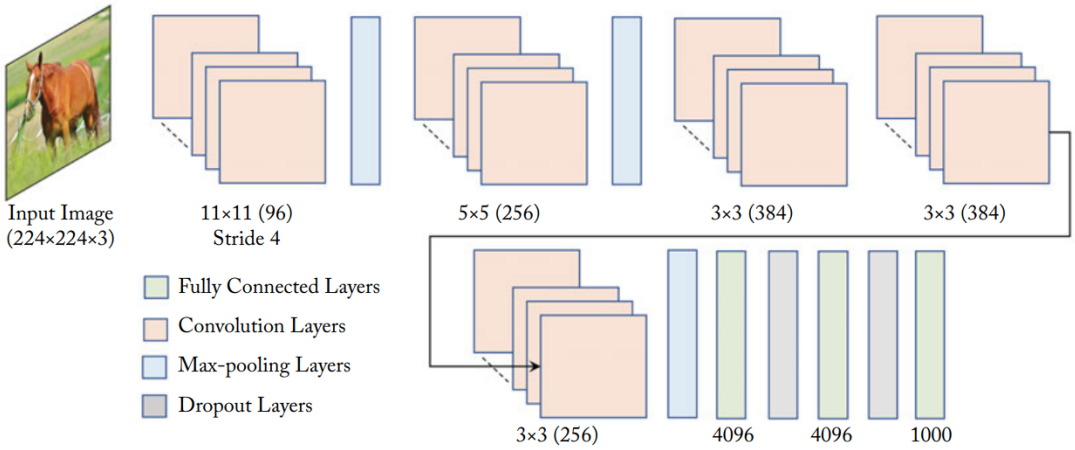
**AlexNet** Em 2012, a vencedora do ILSVRC foi a arquitetura proposta por Alex Krizhevsky, conhecida como AlexNet, ilustrada na Figura 9. A AlexNet é mais profunda e uma versão muito mais ampla da arquitetura LeNet (JOSHI et al., 2017). A principal diferença entre a AlexNet e as CNNs predecessoras é a sua maior profundidade, que lida muito bem com sua grande quantidade de parâmetros, além da utilização de artifícios como *dropout* e *data augmentation*. As cinco primeiras camadas da arquitetura AlexNet são camadas de convolução e *pooling* alternadas de forma similar à LeNet, porém, seguem-se mais três camadas, duas convolucionais e uma de*pooling*. As últimas cama-

**Figura 8: Arquitetura LeNet de CNN. Fonte: (KHAN et al., 2018).**




das são FCL, mas além destas existem camadas *dropout* que ajudam à reduzir *overfiting* (KHAN et al., 2018).

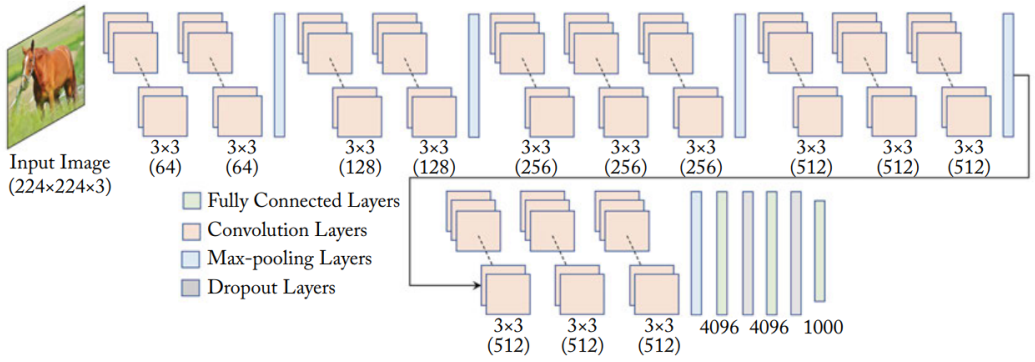
**Figura 9: Arquitetura da AlexNet. Fonte: (KHAN et al., 2018).**




**VGGNet** A arquitetura VGGNet é uma das arquiteturas mais populares desde sua criação em 2014, apesar de não ter sido a vencedora do ILSVRC realizado no respectivo ano. A razão de sua popularidade se dá especialmente em virtude do uso de pequenos filtros de convolução, diminuindo o número de parâmetros ajustáveis e, por conseguinte, aumentando a eficiência do treinamento. A arquitetura VGGNet usa estritamente fitros de convolução de dimensão $3 \times 3$ combinados com camadas de *pooling* para extração de características e um conjunto de três FCLs. Além das camadas de convolução, *pooling* e das camadas conectadas, esta arquitetura também possui as camadas *dropout* como pode ser observado na Figura 10 (KHAN et al., 2018).

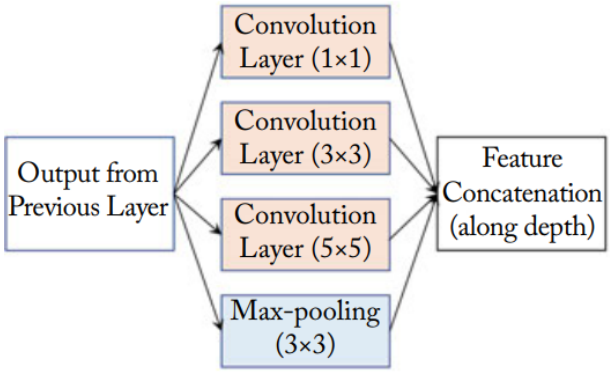
**GoogLeNet** Desenvolvida pela empresa Google e vencedora do ILSVRC 2014, a arquitetura GoogLeNet possui 22 camadas baseadas em um módulo elementar chamado *Inception Module*. O processamento desses módulos ocorre de forma paralela, diferentemente do processamento sequencial das arquiteturas discutidas anteriormente. A ideia

**Figura 10: Arquitetura VGGNet. Fonte: (KHAN et al., 2018).**




central desta arquitetura, também chamada de *Inception*, é paralelizar os módulos e combinar as características da saída sem se preocupar com as funções individuais de cada camada. No entanto, essa abordagem resulta em um mapa de características com muitos elementos, mas para contornar este problema, após a execução do primeiro módulo, a rede realiza uma redução de dimensionalidade utilizando uma FCL antes de continuar o processo de treinamento (KHAN et al., 2018). A representação da arquitetura GoogLeNet encontra-se na Figura 11.

**Figura 11: Arquitetura GoogLeNet. Fonte: (KHAN et al., 2018).**




**ResNet** A *Residual Network* (ResNet) é uma arquitetura desenvolvida pela Microsoft que foi a vencedora do ILSVRC 2015 reduzindo a taxa de erro em 3,6% em cima do desempenho vencedor do ano anterior conquistado pela GoogLeNet. A principal característica da ResNet está nos chamados blocos residuais, em que a entrada original é submetida a uma função de transformação usando uma conexão direta da entrada, chamada *skip identity connection*. Esta transformação que ocorre em um bloco residual é dividida em um termo de identidade (que representa a entrada) e um termo residual, o qual ajuda a evidenciar a transformação dos mapas de características residuais. No ano de 2016, a

arquitetura vencedora do ILSVRC era composta de uma combinação das características das arquiteturas GoogLeNet Inception e ResNet (KHAN et al., 2018).

### 2.2.3. *Transfer Learning*

*Transfer Learning* (TL), ou Transferência de Conhecimento, é uma poderosa técnica de DL a qual possui diversas aplicações em diferentes domínios (GULLI; PAL, 2017). Ao invés de estruturar uma arquitetura de uma CNN e treiná-la por completo, esta técnica permite reutilizar uma rede pré-treinada e adaptá-la a um novo conjunto de dados (SEWAK; KARIM; PUJARI, 2018). Modelos que foram pré-treinados utilizando um vasto e genérico conjunto de dados conseguem capturar características universais, como por exemplo curvas e arestas, em suas primeiras camadas (ZACCONE; KARIM, 2018).

As técnicas de TL podem ser utilizadas de diferentes maneiras, baseando-se nas arquiteturas das CNNs. Existem alguns modelos disponíveis para aplicações que foram pré-treinados utilizando as principais arquiteturas canônicas de CNN e aprenderam as características de grandes conjuntos de dados bastante conhecidos, como o ImageNet e o Places205 (IMAGENET, 2018; PLACES205, 2018). Para diferentes tarefas, esses modelos podem ser alterados modificando a camada de saída e fazendo um retreinamento nas últimas camadas das redes para se obter o aprendizado desejado (KHAN et al., 2018).

## 2.3. Espaços de Cores RGB e CIELab

Um espaço de cores é um modelo matemático utilizado para descrever uma determinada cor representando um *gamut*, isto é, uma paleta de cores que uma certa tecnologia é capaz de reproduzir. Existem diversos modelos de espaços de cores, tais como RGB e CIELab, os quais possuem diferentes formas de representação (GALLETI; SOARES, 2016).

A sigla RGB é uma abreviatura de um espaço de cores formado por três canais: um R, que vem do inglês *red* (vermelho); outro G, do inglês *green* (verde); e o outro B, do inglês *blue* (azul). O RGB forma a síntese aditiva em que, pela sobreposição das três luzes em um ambiente totalmente escuro, obtém-se a luz branca, como ilustrado na Figura 12. Em cada um dos três canais deste espaço de cores, um valor de intensidade é atribuído para cada *pixel* de uma imagem, variando de $0$ a $255$. Essa atribuição tem como finalidade a composição de uma imagem colorida e é muito utilizada na indústria gráfica digital e em mídias digitais, como televisão, câmeras, escâneres, *smartphones*, etc (GALLETI; SOARES, 2016).

A *Commision Internationale L'Eclairage* (CIE), deu nome a um outro espaço de cores bastante utilizado como referência para gerenciamento de cores no tratamento de imagens, o CIELab. A sigla Lab retrata os três canais utilizados por esse sistema os quais estão graficamente representados na Figura 13. O canal simbolizado pela letra $L$ indica luminosidade, variando de preto a branco e os canais das letras $a$ e $b$ são os eixos de cromaticidade, em que $a$ varia de vermelho a verde e $b$ varia de azul a amarelo. Nesse modelo, a separação de cores é feita de maneira a deixar um canal responsável pela formação do desenho da imagem (canal $L$), e os outros dois canais ($a$ e $b$) ficam responsáveis apenas pelas informações das cores (GALLETI; SOARES, 2016).

**Figura 12: Síntese aditiva do espaço de cores RGB. Fonte: (GALLETI; SOARES, 2016).**

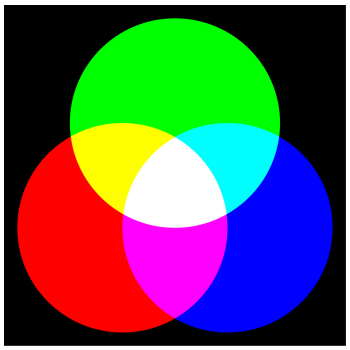
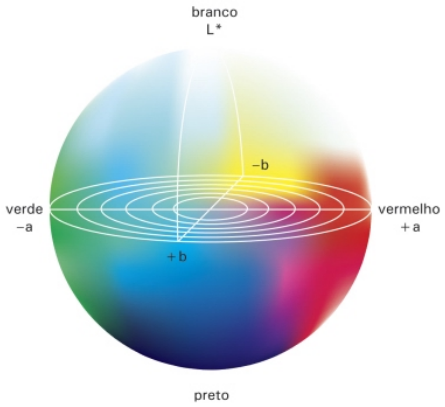


**Figura 13: Representação gráfica do espaço de cores CIELab. Fonte: (GALLETI; SOARES, 2016).**




## 2.4. Tecnologias Utilizadas

As tecnologias e bibliotecas predominantemente utilizadas neste trabalho envolvem e são compatíveis com a linguagem de programação Python, pois esta tem se destacado amplamente em projetos de ML em diversos cenários. Esta é uma linguagem de programação interpretada, interativa, multi-paradigma, de alto nível, multi-plataforma, com uma sintaxe simples e código aberto, idealizada por Guido van Rossum no início da década de 1990 (BORGES, 2010).

Para o processamento da imagens em termos de redimensionamento, persistência, mudança e consulta do espaço de cores, as bibliotecas `PIL` (Pillow) e `colormath` tiveram um papel protagonista (PIL, 2018; COLORMATH, 2018). No tocante à manipulação de arquivos, contemplando abertura, leitura e busca por extensões similares, as bibliotecas `os` e `glob` foram utilizadas (OS, 2018; GLOB, 2018). A manipulação do conjunto de imagens e de suas respectivas representações matriciais ficou por conta da `numpy`, uma biblioteca fundamental para computação científica que é extremamente poderosa para gerenciamento e alterações de matrizes de muitas dimensões (NUMPY, 2018). Ademais, no que diz a respeito do treinamento e testes dos modelos de ML, as bibliotecas `scikit-learn` e `keras` foram consideradas, em que a primeira teve um papel principal nos cálculos automáticos das métricas de desempenho e a segunda nos modelos de DL com parâmetros previamente configurados (LEARN, 2018; KERAS, 2018).

Por fim, a infra-estrutura de computação em nuvem provida pelo Google Cloud

Platform (GCP) totalmente voltada para projetos de ML foi essencial para o desenvolvimento deste projeto. As máquinas virtuais oferecidas pela plataforma aumentaram o poder computacional acessível possibilitando um processamento mais favorável ao cenário em questão. A manipulação e pré-processamento do conjunto íntegro de imagens, o treinamento dos modelos de DL e a fase de análise desses modelos foram executados em instâncias disponíveis pelo GCP tendo em vista os recursos de memória e processamento necessários (GCLOUD, 2018).

## 3. Trabalhos Relacionados

## 4. Solução Proposta

Esta seção apresenta os detalhes da solução proposta pelo presente trabalho de conclusão de curso, descritas segundo o processo de ML (MARSLAND, 2015, vide Seção 1.5). Na Seção 4.1, é apresentada a tarefa de aprendizado considerada, um passo a passo de como a solução proposta está caracterizada e alguns aspectos acerca da avaliação, tais como métrica de desempenho e técnica de validação cruzada. A base de dados experimentais utilizada neste trabalho é detalhada na Seção 4.2 e as tarefas de pré-processamento sobre a mesma são relatadas na Seção 4.3. No tocante aos modelos de DL e seus parâmetros, os mesmos são apresentados na Seção X. Por fim, a Seção Y contempla os resultados obtidos até o momento.

### 4.1. Processo de Aprendizagem de Máquina

No contexto deste trabalho, o problema da colorização de uma imagem será abordado como uma *tarefa de regressão*, cujo objetivo é obter uma estimativa dos parâmetros de coloração dos pixels. De maneira mais detalhada, a Figura 14 ilustra o conjunto de passos a ser realizado.

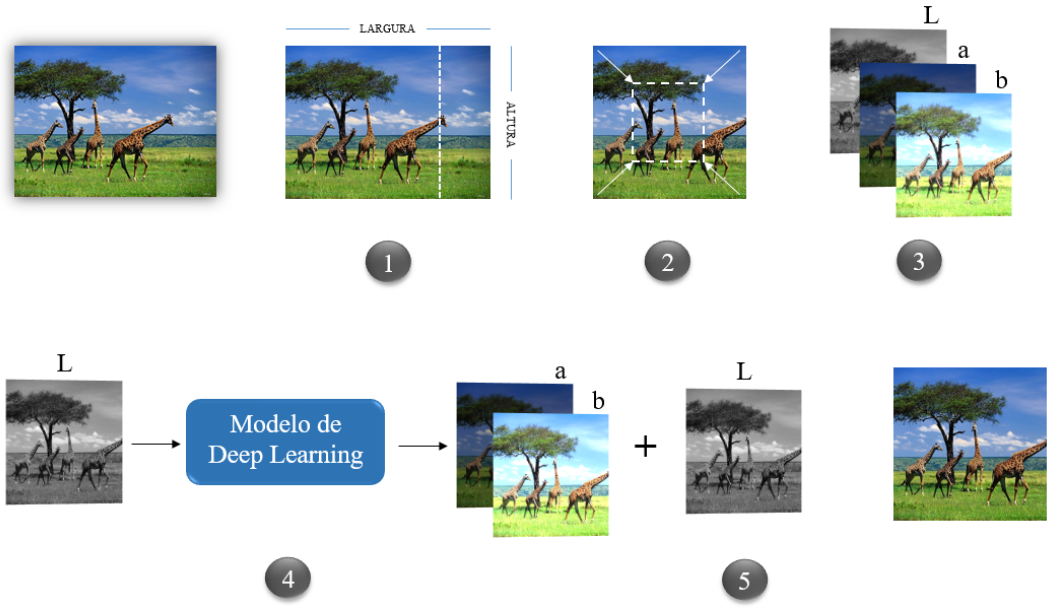



**Figura 14: Detalhamento do processo de aprendizado.**

Dada uma imagem em tons de cinza que se deseja colorir, o primeiro passo a ser realizado (Passo 1) consiste no redimensionamento da imagem de acordo com a dimensão

predominante (altura ou largura), com vistas a obter uma imagem quadrada. Em seguida, para que seja fornecida como entrada para os modelos de DL, a imagem será redimensionada para $128 \times 128$ pixels (Passo 2). Após esta etapa, o espaço de cores da imagem será convertido de RGB para CIELab (Passo 3). Desta conversão, o parâmetro de luminosidade $L$ será fornecido para o modelo de DL, cujo objetivo será produzir duas matrizes com os valores de coloração $a$ e $b$ (Passo 4). Por fim, a luminosidade $L$, já conhecida, será combinada com as matrizes de coloração $a$ e $b$ produzidas, compondo uma versão colorida da imagem inicial (Passo 5).

Para que os modelos propostos para realização da coloração produzam resultados factíveis, é necessária uma etapa de treinamento, em que imagens coloridas serão fornecidas sucessivamente às redes para ajustes dos pesos, produzindo mapas de características mais fidedignos ao domínio do problema. A base de dados a ser utilizada para esta etapa será descrita na seção a seguir.

No escopo deste trabalho serão consideradas as arquiteturas canônicas de CNNs VGGNet, GoogLeNet e ResNet as quais serão ajustadas ao problema em questão mediante TL. Diferentes valores de parâmetros e hiperparâmetros (épocas, taxa de aprendizado, *batch size*, etc.) serão considerados, sempre que possível, buscando um melhor ajuste do modelo ao problema em questão.

Nesta tarefa de regressão, realizada de acordo com o paradigma supervisionado, a métrica de desempenho utilizada será o Erro Quadrático Médio (MSE, do inglês *Mean Squared Error*), definido como:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{2}$$

em que $y_i$ é o valor real esperado para o exemplo $i$, $\hat{y}_i$ é o valor previsto pelo modelo para este exemplo e $n$ é o número de amostras presentes no conjunto de dados (FACELI et al., 2011).

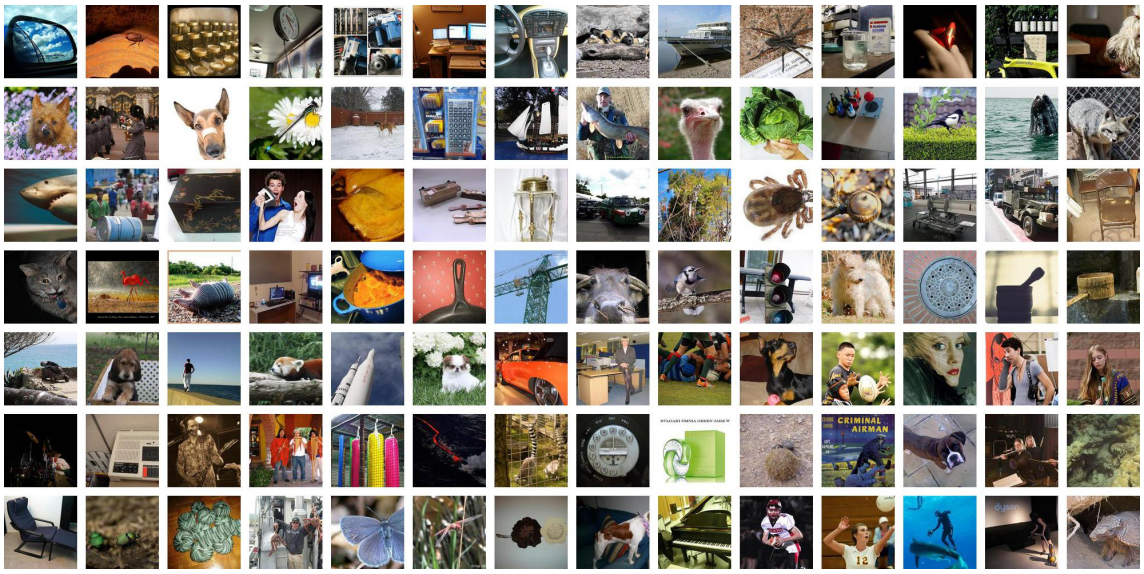
O treinamento e testes das CNNs seguirão a abordagem *Holdout* de validação cruzada, em que $70\%$ dos dados serão utilizados no treino e ajuste de parâmetros e o restante para avaliação, com vista a capturar a qualidade da generalização proposta pelos modelos considerados (BRINK; RICHARDS; FETHEROLF, 2017).

### 4.2. Dados Experimentais

Uma base de dados experimentais se faz necessária para treinar e testar os modelos baseados em DL, fornecendo a experiência sobre o problema e permitindo o aprendizado por meio do ajuste dos parâmetros treináveis. Considerando este objetivo, a base dados considerada no escopo deste trabalho consiste no conjunto de imagens de validação da tarefa *Fine-grained classification* do ILSVRC 2012 (IMAGENET, 2018).

A base de dados experimentais adotada dispõe de um total de $50$ mil imagens de diferentes naturezas, as quais foram coletadas do *website* Flickr e de outras plataformas semelhantes e rotuladas manualmente por ausência ou presença de $1000$ diferentes categorias (RUSSAKOVSKY et al., 2015). Uma amostra das imagens contidas nesta base de dados pode ser vista na Figura 15.

**Figura 15: Visão geral do conjunto de dados.**



Esta base de dados foi escolhida por possuir uma quantidade de imagens favorável para o treinamento dos muitos parâmetros presentes nos modelos de DL e pelas imagens possuirem uma variedade de classes, o que caracteriza um conjunto de exemplos suficientemente representativo para a generalização em um cenário de colorização artificial. Embora haja conjuntos de imagens do ILSVRC de maior tamanho, este conjunto escolhido permite uma manipulação razoável dos dados, considerando os recursos computacionais disponíveis neste projeto.

Segundo a abordagem *Holdout* de validação cruzada a ser adotada, conforme mencionado na Seção 4.1, estas imagens serão particionadas em dois conjuntos: o de treinamento, contendo $35$ mil imagens, para ajuste dos parâmetros dos modelos; e o de testes, contendo as $15$ mil imagens remanescentes, para análise de desempenho.

Entretanto, antes de utilizar diretamente esta base de dados junto aos modelos de DL, as imagens precisaram de ajustes de pré-processamento, conforme detalhado na seção a seguir.

## 4.3. Limpeza e Pré-Processamento

Como se sabe, algoritmos de ML precisam de quantidades significativas de dados que estejam preferencialmente sem muitos ruídos. Entretanto, com o aumento do tamanho do conjunto de dados os custos computacionais também aumentam e faz-se necessário um pré-processamento desses dados para estruturá-los da maneira ideal sem que haja uma excessiva sobrecarga computacional (MARSLAND, 2015).

O conjunto de imagens que será utilizado no processo de colorização deste trabalho será submetido a um pré-processamento que se inicia padronizando a dimensão das imagens. Primeiramente, as imagens são redimensionadas de forma que possuam os mesmos tamanhos de largura e altura, para obtenção de uma imagem quadrada. Em seguida, cada imagem é redimensionada para $128 \times 128$ pixels, padronizando todas as imagens nesta mesma dimensão. Esta padronização visa tornar a tarefa de aprendizado factível diante dos recursos computacionais disponíveis.

Após a etapa de redimensionamento, a etapa seguinte consistiu em selecionar as imagens cujo espaço de cores fosse o RGB. Isto foi efetuado com o intuito de eliminar imagens em tons de cinza e em outros espaços de cores que demandassem um maior processamento posterior. No caso das imagens em tons de cinza, em particular, a eliminação das mesmas simplifica o processo de treinamento dos modelos, visto que estas não fornecem informação relevantes para aprendizado de colorações. Nesta etapa, foram eliminadas cerca de $900$ imagens que se encaixam em um dos critérios mencionados.

Em seguida, todas as imagens foram então convertidas do espaço de cores RGB para o espaço de cores CIELab, tendo em vista a diminuição dos parâmetros de entrada e saída no processo de aprendizagem e também a adequação aos procedimentos descritos na Seção 4.1. Cada imagem passou então a corresponder a uma matriz de dimensões $3 \times 128 \times 128$, a qual foi separada em duas partes: componente $L$, de dimensões $128 \times 128$, a ser fornecida como *entrada* para os modelos, e componentes $a$ e $b$, a serem fornecidos como rótulo de *saída* para o aprendizado supervisionado dos modelos, com dimensões $2 \times 128 \times 128$.

### 4.4. Modelos e Parâmetros Considerados

Para o Passo 4 descrito na Seção 4.1 será considerada a abordagem de *Transfer Learning*, visando aproveitar parâmetros de redes CNNs pré-treinadas com *datasets* massivos, a exemplo do ImageNet. Para tanto, considerando a utilização do *framework* Keras, foi consultada a biblioteca de modelos previamente treinados, disponíveis no módulo `applications`. Como resultado, foram identificadas as arquiteturas canônicas e suas características elencadas na Tabela 2.

**Tabela 2: Arquiteturas canônicas e suas características disponíveis no módulo `keras.applications`.**

| Arquitetura | Profundidade | Parâmetros | Tamanho |
|---|---|---|---|
| Xception | 126 | $22,910,480$ | 88 MB |
| VGG16 | 23 | $138,357,544$ | 528 MB |
| VGG19 | 26 | $143,667,240$ | 549 MB |
| ResNet50 | 168 | $25,636,712$ | 99 MB |
| InceptionV3 | 159 | $23,851,784$ | 92 MB |
| InceptionResNetV2 | 572 | $55,873,736$ | 215 MB |
| MobileNet | 88 | $4,253,864$ | 17 MB |

É interessante notar que todos estes modelos disponíveis foram pré-treinados com a base de dados do ImageNet. Considerando a quantidade de imagens e a variedade desses exemplos, acredita-se que estes pesos podem contribuir positivamente para o problema da coloração dada a transferência de aprendizado.

Levando em conta os recursos computacionais disponíveis em nuvem para este projeto e o tempo de processamento, as CNNs VGG16, ResNet50 e InceptionV3 disponíveis no `keras.applications` serão ajustadas perante *Transfer Learning* para a tarefa de coloração proposta.

Dessas redes pré-treinadas, as últimas camadas serão substituídas por camadas completamente conectadas com saída compatível com as matrizes $a$ e $b$ de coloração.

Eventuais re-treinos serão efetuados para ajuste de interconexão dessa nova camada. Serão consideradas 100 épocas para o treinamento dos modelos propostos e os demais parâmetros terão seus valores padrões preservados.

### 4.5. Resultados Parciais

## 5. Considerações Parciais

## Referências


ARAUJO, N. et al. Previsão do volume mensal de precipitações em manaus, amazonas com redes neurais aritificiais. In: *Anais do IV Encontro Regional de Informática da Região Norte I*. Manaus, Amazonas: ERIN, 2017. p. 108–116.

BORGES, L. E. *Python para Desenvolvedores*. Rio de Janeiro, Rio de Janeiro: Novatec, 2010.

BRAGA, A. de P.; CARVALHO, A. P. de Leon F. de; LUDERMIR, T. B. *Redes Neurais Artificiais: Teoria e Aplicações*. 2. ed. Rio de Janeiro, Rio de Janeiro: LTC, 2007.

BRINK, H.; RICHARDS, J. W.; FETHEROLF, M. *Real-World Machine Learning*. New York, Estados Unidos: Manning Publications, 2017.

BUDUMA, N. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*. 1. ed. Sebastopol, California: O'Reilly Media, 2017.

CAEXETA, G. S. *VisionDraughts - Um Sistema de Aprendizagem de Jogos de Damas em Redes Neurais, Diferenças Temporais, Algoritmos Eficientes de Busca em Árvore e Informações Perfeitas Contidas em Bases de Dados*. Uberlândia, Minas Gerais: Universidade Federal de Uberlândia, 2008.

CARVALHO, C. E. S. de. *Reconhecimento de Caracteres Manuscritos Utilizando Redes Neurais Artificiais*. Brasília, Distrito Federal: Centro Universitário de Brasília, 2006.

CHOLLET, F. *Deep Learning with Python*. 1. ed. Shelter Island, New York: Manning, 2017.

COLORMATH. *python-colormath*. 2018. Disponível em: ⟨https://python-colormath.readthedocs.io/en/latest/⟩.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, v. 2, p. 303–314, December 1989.

DELICATO, F.; PIRES, P.; SILVEIRA, I. Jornada de Atualização em Informática 2017. In: CSBC. *Congresso da Sociedade Brasileira de Computação*. Porto Alegre, Rio Grande do Sul: SBC, 2017. cap. 6, p. 212–260.

FACELI, K. et al. *Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina*. 1. ed. Rio de Janeiro, Rio de Janeiro: LTC, 2011.

GALLETI, L. S.; SOARES, R. V. *Captura e tratamento de imagens*. São Paulo, São Paulo: SENAI-SP, 2016.

GCLOUD. *Google Cloud*. 2018. Disponível em: ⟨https://cloud.google.com/⟩.

GLOB. *Unix style pathname pattern expansion*. 2018. Disponível em: ⟨https://docs.python.org/3/library/glob.html⟩.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. 1. ed. Cambridge, Massachsetts: The MIT Press, 2016. v. 1. (Adaptive computation and machine learning series, v. 1).

GULLI, A.; PAL, S. *Deep Learning with Keras*. 1. ed. Birmingham, West Midlands: Packt, 2017.

HAYKIN, S. S. *Neural Networks and Learning Machines*. 3. ed. Upper Saddle River, New Jersey: Perason, 2009.

IMAGENET. *ImageNet*. 2018. Disponível em: ⟨http://www.image-net.org/⟩.

JOSHI et al. *Information and Communication Technology for Intelligent Systems (ICTIS 2017) Volume 2*. 1. ed. [S.l.]: Springer, 2017. v. 2. (Smart innovation systems and technologies 84, v. 2).

KERAS. *Keras: The Python Deep Learning library*. 2018. Disponível em: ⟨https://faroit.github.io/keras-docs/2.1.2/⟩.

KHAN, S. et al. *A Guide to Convolutional Neural Networks for Computer Vision*. 1. ed. Australia: Morgan and Claypool, 2018. v. 1. (Synthesis Lectures on Computer Vision, v. 1).

LEARN scikit. *scikit-learn*. 2018. Disponível em: ⟨http://scikit-learn.org/stable/index.html⟩.

LIMA, P. M. de. *Redes Neurais para Predição de Séries Temporais de Precipitação em Manaus, Amazonas*. 2016. Trabalho de Conclusão de Curso da Universidade do Estado do Amazonas. Universidade do Estado do Amazonas.

LU, L. et al. *Deep Learning and Convolutional Neural Networks for Medical Image Computing*. Cham, Switzerland: Springer, 2017. (Advances in Computer Vision and Pattern Recognition).

MARSLAND, S. *Machine Learning An Algorithmic Perspective*. Florida, Estados Unidos: Taylor and Francis, 2015.

MNIST. *THE MNIST DATABASE of handwritten digits*. 2018. Disponível em: ⟨http://yann.lecun.com/exdb/mnist/⟩.

NUMPY. *NumPy*. 2018. Disponível em: ⟨http://www.numpy.org/⟩.

OS. *Miscellaneous operating system interfaces*. 2018. Disponível em: ⟨https://docs.python.org/3/library/os.html⟩.

PEREIRA, R. et al. Abordagem deep learning para classificação de lesões mamárias. In: *Anais do XXXVI Congresso da Sociedade Brasileira de Computação*. Porto Alegre, Rio Grande do Sul: CSBC, 2016. p. 2597–2600.

PIL. *Pillow*. Rio de Janeiro, Rio de Janeiro: [s.n.], 2018. Disponível em: ⟨https://pillow.readthedocs.io/en/5.1.x/handbook/overview.html⟩.

PLACES205. *Places: the scene recognition database*. 2018. Disponível em: ⟨http://places.csail.mit.edu/⟩.

ROJAS, R. *Neural Networks: A Systematic Introduction*. 1. ed. Heidelberg, Berlin: Springer, 1996.

ROSENBLAT, F. *Principles of Neurodynamics*: Perceptrons and the theory of brain mechanisms. Washington, DC, 1961. 626 p.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

SEWAK, M.; KARIM, R.; PUJARI, P. *Pratical Convolutional Neural Networks*. 1. ed. Birmingham, West Midlands: Packt, 2018.

SILVA, A. G. et al. Classificação de expressões faciais negativas na língua brasileira de sinais. In: *Anais do IV Encontro Regional de Informática da Região Norte I*. Manaus, Amazonas: ERIN, 2017. p. 80–88.

SILVA, G. da et al. Classification of malignancy of lung nodules in ct images using convolutional neural network. In: *Anais do XXXVI Congresso da Sociedade Brasileira de Computação*. Porto Alegre, Rio Grande do Sul: CSBC, 2016. p. 2481–2489.

SOUSA, R. et al. Redes neurais artificais aplicadas à previsão antecipada de precipitações na região central de manaus. In: *Anais do IV Encontro Regional de Informática da Região Norte I*. Manaus, Amazonas: ERIN, 2017. p. 89–97.

TINOS, R. *Detecção e Diagnóstico de Falhas em Robôs Manipuladores via Redes Neurais Artificiais*. São Carlos, São Paulo: Universidade de São Paulo, 1999.

ZACCONE, G.; KARIM, M. R. *Deep Learning with TensorFlow*. Birmingham, West Midlands: Packt, 2018.