



Universidade de Brasília

Instituto de Ciências Exatas  
Departamento de Ciência da Computação

**Estudo comparativo de redes neurais convolucionais  
para a classificação da qualidade de imagens de  
documentos de identidade**

Leonardo de Almeida e Marcus Vinícius da Silva Borges

Monografia apresentada como requisito parcial  
para conclusão do Bacharelado em Ciência da Computação

Orientadora  
Prof<sup>a</sup>. Dr<sup>a</sup>. Roberta Barbosa Oliveira

Brasília  
2021



Universidade de Brasília

Instituto de Ciências Exatas  
Departamento de Ciência da Computação

**Estudo comparativo de redes neurais convolucionais  
para a classificação da qualidade de imagens de  
documentos de identidade**

Leonardo de Almeida e Marcus Vinícius da Silva Borges

Monografia apresentada como requisito parcial  
para conclusão do Bacharelado em Ciência da Computação

Prof<sup>a</sup>. Dr<sup>a</sup>. Roberta Barbosa Oliveira (Orientadora)  
CIC/UnB

Prof. Dr. Camilo Chang Dórea Prof. Dr. Vinicius Ruela Pereira Borges  
CIC/UnB CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli  
Coordenador do Bacharelado em Ciência da Computação

Brasília, 26 de maio de 2021

# **Dedicatória**

Dedicamos esse trabalho a todos pessoas que acreditaram na nossa jornada e sempre nos incentivaram a persistir.

# Agradecimentos

Agradecemos ao Carlos Luz por ter nos ajudado bastante durante o desenvolvimento deste trabalho e também a todas nossas amizades que começaram na CJR e na Central. Sem a ajuda de todas essas pessoas essa jornada teria sido muito mais difícil. Também agradecemos ao Sidoka por ter tornado os dias que investimos nesse trabalho mais jubilantes ainda que estivéssemos apoquentados. Por fim agradecemos a toda a comunidade do *Stack Overflow* e a Profa. Dra. Roberta Barbosa Oliveira que contribuiram bastante com a nossa formação.

# Resumo

O uso de documentos de identidade como Registro Geral e Carteira Nacional de Habilitação é uma das formas de verificação de clientes em sistemas *onlines* como bancos e *e-commerce*s. Nesses sistemas, uma imagem do documento é enviada para que possa ser feita a identificação do cliente. Devido à grande quantidade de dispositivos que podem capturar essas imagens, um desafio existente é saber se a qualidade da imagem submetida está boa ou não. Uma das possíveis técnicas que podem ser utilizadas para fazer a classificação dessa imagem quanto a sua qualidade é a de Redes Neurais Convolucionais devido a capacidade de extração automática de características. Nesse sentido, o presente trabalho apresenta um estudo comparativo do desempenho de algumas arquiteturas de redes neurais convolucionais, tais como, ResNet50, VGG16, Xception, DenseNet e MobileNet, na classificação da qualidade das imagens de documentos de identificação. Nos experimentos que foram realizados, as arquiteturas foram combinadas com as técnicas de *Transfer Learning*, *Dropout*, *Data Augmentation* e *Fine-Tuning* e as métricas de acurácia, revocação, precisão, F1-Score e tempo médio de classificação, foram utilizadas para fazer as comparações. As arquiteturas que apresentaram os melhores resultados foram a ResNet50, DenseNet e Xception. Por fim, é possível concluir que as arquiteturas VGG16 e Xception apresentaram melhora nos resultados com a adição das técnicas de *Dropout*, *Data Augmentation* e *Fine-Tuning* enquanto a arquitetura ResNet50 apresentou queda no desempenho com a adição das mesmas técnicas.

**Palavras-chave:** Redes Neurais Convolucionais, Classificação de imagens, Documentos de Identidade

# Abstract

The use of identity documents such as *Registro Geral* and *Carteira Nacional de Habilitação* is one of the ways to verify customers in online systems such as banks and e-commerce. In these systems, an image of the document is sent so that the customer's identification can be made. Due to the large number of devices that can capture these images, an existing challenge is whether the quality of the submitted image is good or not. One of the possible techniques that can be used to classify this image according to its quality is Convolutional Neural Networks. In this sense, the present work describes a comparative study of the performance of the architectures ResNet, VGG, Xception, DenseNet and MobileNet in the classification of the quality of images of identification documents. In the experiments that were carried out, the architectures were combined with Transfer Learning, Dropout, Data Augmentation and Fine-Tuning techniques and the metrics of accuracy, recall, precision, F1-Score and average time of evaluation of the test dataset, were used to make the comparisons. Finally, it's possible to conclude that the VGG16 and Xception architectures superior in results with the addition of Dropout, Data Augmentation and Fine-Tuning techniques, while the ResNet50 architecture presented a drop in performance with the addition of the techniques.

**Keywords:** Convolutional Neural Networks, Image Classification, Identity Documents

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Motivação . . . . .	1
1.2	Objetivo . . . . .	2
1.3	Estrutura do Documento . . . . .	2
<b>2</b>	<b>Fundamentação Teórica</b>	<b>4</b>
2.1	Aprendizado de Máquina . . . . .	4
2.2	Redes Neurais Artificiais . . . . .	5
2.2.1	Modelo de neurônio . . . . .	5
2.2.2	Arquiteturas de Redes Neurais Artificiais . . . . .	7
2.3	Redes Neurais Convolucionais . . . . .	8
2.3.1	Operação de convolução em imagens . . . . .	9
2.3.2	<i>Kernels</i> de convolução . . . . .	9
2.3.3	Camada de Convolução . . . . .	9
2.3.4	Camada de Ativação . . . . .	9
2.3.5	Camada de <i>Pooling</i> . . . . .	10
2.4	Arquiteturas de Redes Neurais Convolucionais . . . . .	11
2.4.1	AlexNet . . . . .	12
2.4.2	VGG . . . . .	12
2.4.3	Inception . . . . .	13
2.4.4	ResNet . . . . .	14
2.5	<i>Transfer Learning</i> . . . . .	14
2.6	Avaliação de desempenho de modelos . . . . .	15
2.6.1	Matriz de confusão . . . . .	15
<b>3</b>	<b>Revisão da Literatura</b>	<b>17</b>
3.1	Técnicas de classificação de imagens de documentos . . . . .	17
3.2	Considerações Finais . . . . .	20

<b>4 Metodologia Proposta</b>	<b>21</b>
4.1 Descrição da base de dados . . . . .	21
4.2 Redes Neurais Convolucionais . . . . .	25
4.3 <i>Transfer Learning</i> . . . . .	26
4.4 <i>Data Augmentation</i> . . . . .	26
4.5 <i>Dropout</i> . . . . .	28
4.6 <i>Fine-tuning</i> . . . . .	28
<b>5 Experimentos e Resultados</b>	<b>29</b>
5.1 Ambiente de Desenvolvimento Utilizado . . . . .	29
5.2 Reprodutibilidade dos experimentos . . . . .	30
5.3 Descrição do <i>Hardware</i> . . . . .	30
5.4 <i>Callbacks</i> . . . . .	31
5.5 Hiperparâmetros . . . . .	31
5.6 Visão geral dos Experimentos . . . . .	31
5.7 Resultados . . . . .	33
<b>6 Conclusão</b>	<b>39</b>
6.1 Considerações finais . . . . .	39
6.2 Limitações . . . . .	40
6.3 Trabalhos Futuros . . . . .	40
<b>Referências</b>	<b>41</b>

# Listas de Figuras

2.1	Representação do neurônio biológico (adaptada de Borges et al. [1]). . . . .	6
2.2	Arquitetura de um neurônio artificial (adaptada de Mitchel [2]). . . . .	6
2.3	Modelo de Perceptrons. . . . .	7
2.4	Modelo de Perceptrons Multicamadas. . . . .	7
2.5	Arquitetura geral de uma Rede Neural Convolucional (adaptada de Haleem et al. [3]). . . . .	8
2.6	Gráfico da função de ativação ReLU. . . . .	10
2.7	Gráfico da função de ativação Sigmoid. . . . .	11
2.8	Representação da arquitetura AlexNet (baseada em Krizhevsky et al. [4]).	12
2.9	Representação da arquitetura VGG16 (adaptada de Roman [5]). . . . .	13
2.10	Módulo incepção baseado em Szegedy et al. [6] (adaptado de Khan et al. [7]). . . . .	14
2.11	Bloco Residual (Adaptado de He et al. [8]). . . . .	15
2.12	Matriz de Confusão. . . . .	16
4.1	Etapa das da metodologia proposta. . . . .	22
4.2	Amostras de documentos classificados como bons. . . . .	23
4.3	Amostras de documentos classificados como ruins. . . . .	24
4.4	Demonstração do processo de <i>Transfer Learning</i> via extração de características. . . . .	27
5.1	Visão geral do primeiro conjunto de experimentos. . . . .	32
5.2	Visão geral do primeiro conjunto de experimentos 2. . . . .	33
5.3	Visão geral do primeiro conjunto de experimentos 3. . . . .	34
5.4	Desempenho das métricas da arquitetura VGG16. . . . .	36
5.5	Desempenho das métricas da arquitetura ResNet50. . . . .	37
5.6	Desempenho das métricas da arquitetura Xception. . . . .	37
5.7	Desempenho das métricas da arquitetura DenseNet. . . . .	38
5.8	Desempenho das métricas da arquitetura MobileNet. . . . .	38

# **Lista de Tabelas**

4.1 Quantidade de imagens por classe. . . . .	25
5.1 Resultados para as combinações de arquiteturas de CNNs e variações de parâmetros especificadas em cada conjunto de experimentos. . . . .	34
5.2 Variância do tempo de cada conjunto de experimento. . . . .	35

# Listas de Abreviaturas e Siglas

**AUC-ROC** *Area Under the ROC Curve.*

**BOW** *Bag of Words.*

**CNH** Carteira Nacional de Habilitação.

**CNN** *Rede Neural Convolucional.*

**CNNs** *Redes Neurais Convolucionais.*

**EAST** *Efficient and Accurate Scene Text.*

**FFT** *Fast Fourier Transform.*

**GPUs** *Graphics Processing Units.*

**HOG** *Histogram of Oriented Gradients.*

**MLPs** Perceptrons Multicamadas.

**MSE** *Mean Squared Error.*

**OCR** *Optical Character Recognition.*

**PHOG** *Pyramid Histogram of Oriented Gradients.*

**ReLU** *Rectified Linear Unit.*

**RG** Registro Geral.

**RMSE** *Root-Mean-Square Error.*

**RNA** Redes Neurais Artificiais.

**SIFT** *Scale Invariant Feature Transform.*

**SURF** *Speeded Up Robust Features.*

**SVM** *Support Vector Machine.*

**VLAD** *Vector of Locally Aggregated Descriptors.*

# Capítulo 1

## Introdução

### 1.1 Motivação

Uma das formas de verificação da identidade de clientes em sistemas *onlines* como bancos e *e-commerce* é baseado em documentos de identidade pessoais [9]. Uma das primeiras etapas do processo de verificação da identidade de clientes consiste em submeter um documento pessoal. Um dos desafios dessa etapa é a diversidade de imagens que podem ser submetidas pelo cliente, devido a qualidade de *smartphones* e câmeras digitais [10]. Nesse sentido, a tarefa de classificação dos documentos com base em critérios preestabelecidos é relevante para automatização de processos de reconhecimento de dados pessoais[11].

Uma abordagem que pode ser utilizada para realizar a classificação de imagens de documentos é utilizando extração de características [12]. Uma outra abordagem é utilizando *Redes Neurais Convolucionais* (CNNs). Uma CNN é uma rede neural do tipo *feedforward* que é capaz de extrair características utilizando estruturas convolucionais [13]. Popreshnyak et al. [11] mostraram que é possível obter bons resultados com o uso de CNNs em tarefas similares, como detecção de múltiplos documentos de identidade em uma imagem. Um dos problemas que podem ocorrer no uso de CNNs é o de *overfitting*, o que dificulta a generalizar as características para classificar novas imagens. Uma forma de evitar o *overfitting* é com o uso de técnicas como *dropout* [14]. A técnica de *Data Augmentation* também pode ser utilizada com a finalidade de mitigar o *overfitting*, uma vez que durante o processo de treinamento da CNN, algumas imagens sofrem transformações, como rotação e *zoom*, para que seja possível expor o modelo a diferentes imagens [13]. Por fim, a técnica de *Fine-Tuning* pode melhorar o desempenho da CNN [15]. Nesse sentido, este trabalho apresenta um estudo comparativo entre diferentes arquiteturas de CNNs combinadas com as técnicas de *Dropout*, *Data Augmentation* e *Fine-Tuning* aplicadas a detecção da qualidade da imagem de documentos de identidade. A base de dados que foi utilizada no presente trabalho foi cedida por uma empresa que possui um processo interno

em que é necessário realizar a classificação de imagens de documentos de identidade dentro duas classes relacionadas com a qualidade: boa e ruim. Atualmente, a classificação é feita manualmente, e a automatização desse processo contribuiria reduzindo o esforço humano.

## 1.2 Objetivo

O presente trabalho possui como objetivo geral realizar um estudo comparativo do desempenho de diferentes arquiteturas de CNNs na classificação da qualidade de imagens de documentos para resolver o problema existente na empresa que forneceu a base de dados. Para alcançar esse objetivo, foram definidos os seguintes objetivos específicos:

- Explorar estratégias e técnicas apresentadas na literatura para resolver o problema de classificação de documentos;
- Analisar quais arquiteturas CNNs apresentam melhores resultados para a classificação de qualidade de documentos;
- Verificar se a combinação das técnicas de *Transfer Learning*, *Dropout*, *Data Augmentation* e *Fine-Tuning* podem melhorar os resultados das arquiteturas CNNs para o problema em questão.

## 1.3 Estrutura do Documento

Este trabalho é composto pelos seguintes capítulos:

- Capítulo 2: Introduz conceitos de redes neurais artificiais tradicionais e profundas, apresentando a teoria de algumas arquiteturas de classificação, assim como as técnicas de avaliação de desempenho da classificação;
- Capítulo 3: Realiza a revisão de literatura, na qual foram selecionados alguns artigos que exploram os temas de segmentação e classificação de imagens de documentos utilizando diversas técnicas diferentes;
- Capítulo 4: Descreve a comparação de modelos baseados em CNNs desde a base de dados até a forma de utilização das arquiteturas e parâmetros das redes através dos modelos de classificação utilizados;
- Capítulo 5: Apresenta os resultados obtidos, utilizando as técnicas de avaliação de desempenho explicitadas;

- Capítulo 6: Discute as conclusões obtidas a partir dos resultados dos experimentos e apresenta sugestões para trabalhos futuros.

# **Capítulo 2**

## **Fundamentação Teórica**

Neste capítulo, serão apresentadas as técnicas utilizadas no desenvolvimento do presente trabalho, assim como os conceitos fundamentais para entendimento dessas técnicas, com o objetivo de familiarizar o leitor.

### **2.1 Aprendizado de Máquina**

Segundo Monard e Baranauskas [16], aprendizado de máquina é uma subárea da inteligência artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento. Os algoritmos de aprendizado de máquina constroem um modelo matemático baseado em uma amostra de dados, conhecida como conjunto de treinamento, a fim de fazer classificações sem ser explicitamente programado para tal. Algoritmos de aprendizado são usados em uma grande variedade de aplicações, como filtragem de spam em emails e visão computacional, em que é difícil ou até inviável desenvolver algoritmos convencionais para realizar as tarefas necessárias.

Uma das diversas divisões de algoritmos de aprendizado de máquina é quanto ao modo de aprendizado [17]: (1) aprendizado supervisionado, (2) aprendizado não supervisionado e (3) aprendizado por reforço. No aprendizado supervisionado o algoritmo é treinado utilizando-se um conjunto de dados que possui as saídas esperadas para aquelas entradas. No aprendizado não supervisionado não se tem as saídas esperadas e, muitas vezes, é inviável se ter pela natureza do problema a ser resolvido, as respostas esperadas no conjunto de treinamento, sendo o algoritmo o próprio responsável por compreender a diferença entre os dados passados no treinamento. Por fim, no aprendizado por reforço, o algoritmo é responsável por aprender a partir de novas situações utilizando, por exemplo, um método de tentativa e erro.

A área de aprendizado de máquina possui diversos tipos de algoritmos estudados que podem ser divididos de acordo com o problema que resolvem. Alguns exemplos desses algoritmos são: Regressão Linear, Regressão Logística, Árvore de Decisão e Redes Neurais. Esse último algoritmo trouxe ótimos resultados comparados aos até então usados na área de visão computacional como pode ser visto em Krizhevsky et al. [4]. Algumas das vantagens das Redes Neurais sobre a Regressão Logística, segundo Tu [18], é a habilidade de detectar todas as possíveis interações entre as variáveis preditoras.

## 2.2 Redes Neurais Artificiais

De acordo com Simon Haykin [19], Redes Neurais Artificiais (RNA) são modelos matemáticos que de alguma forma tentam imitar as estruturas neurais biológicas e possuem capacidade computacional adquirida através do aprendizado e generalização. De maneira resumida, essas redes são capazes de reconhecer padrões para mais tarde utilizar o conhecimento adquirido para classificar novos dados a partir das abstrações construídas.

Na Seção 2.2.1 será introduzido o modelo de neurônio artificial, um dos conceitos bases dentro das Redes Neurais Artificiais. Na Seção 2.2.2 é dada uma breve introdução a possíveis arquiteturas de Redes Neurais Artificiais.

### 2.2.1 Modelo de neurônio

Um neurônio é a unidade fundamental do nosso cérebro. É estimado que o cérebro possui aproximadamente 100 bilhões de neurônios e essa massiva rede biológica nos permite pensar e perceber o mundo ao nosso redor. Basicamente o que um neurônio faz é receber informações de outros neurônios. Um único neurônio recebe alguma informação de entrada que é recebida pelos dendritos, mostrados na Figura 2.1. Essas entradas são sumarizadas no corpo celular e transformados em um sinal que é enviado para outros neurônios através do axônio. O Axônio é conectado com os dendritos de outros neurônios através das sinapses [20]. A sinapse pode agir como um peso e fazer o sinal que passa por ele mais forte ou mais fraco baseado em quão frequente essa conexão é usada.

Esse entendimento biológico do neurônio pode ser traduzido em um modelo matemático [20] como o mostrado na Figura 2.2. O neurônio artificial da Figura 2.2 geralmente recebe um vetor de características ( $x_1, x_2, \dots, x_n$ ), em que cada um dos valores do vetor é multiplicado por um peso específico, ( $w_1, w_2, \dots, w_n$ ). Cada conexão de neurônios tem seu próprio peso e esses são os únicos valores que, normalmente, são modificados durante o processo de aprendizado. Além disso, ao resultado dessas operações pode ser somado um valor constante  $b$  chamado *bias*, ou viés, conforme a Equação 2.1. Os pesos das carac-

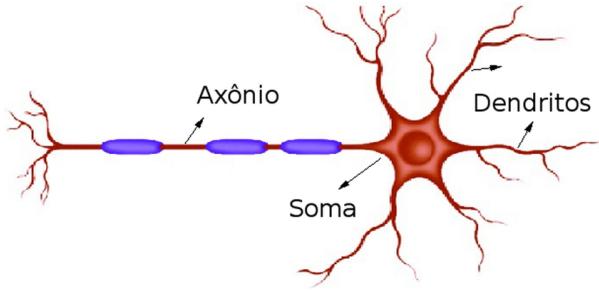


Figura 2.1: Representação do neurônio biológico (adaptada de Borges et al. [1]).

terísticas e do *bias* são ajustados por alguma regra de aprendizagem durante o processo de treinamento da rede neural.

$$z = \sum_{i=0}^n (x_i w_i) + b \quad (2.1)$$

Na Equação 2.1  $n$  representa a quantidade de entradas e  $z$  é a soma dos produtos das entradas e pesos somado ao *bias*. O resultado da Equação 2.1 é passado para a função de ativação para fornecer o valor de saída do neurônio. A função de ativação serve para mapear o valor total calculado anteriormente para números dentro de algum intervalo.

Um exemplo comum desses tipo de função é a sigmoide, mostrada na Equação 2.2, que é uma função não-linear, o que no geral favorece o aprendizado de estruturas mais complexas nos dados, e resulta como saída um valor de 0 a 1, tendo um formato S.

$$f(x) = \frac{1}{(1 + e^{-x})} \quad (2.2)$$

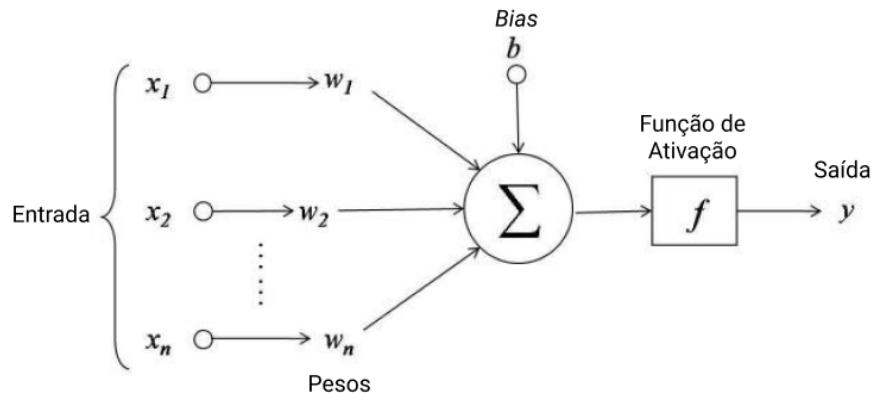


Figura 2.2: Arquitetura de um neurônio artificial (adaptada de Mitchel [2]).

## 2.2.2 Arquiteturas de Redes Neurais Artificiais

O modelo da Seção 2.2.1 também é chamado de Perceptron. Outros modelos mais complexos podem ser criados a partir desse modelo básico, como pode ser visto na Figura 2.3. As redes de múltiplas camadas, como a da Figura 2.3, são chamadas de Perceptrons Multicamadas (MLPs).

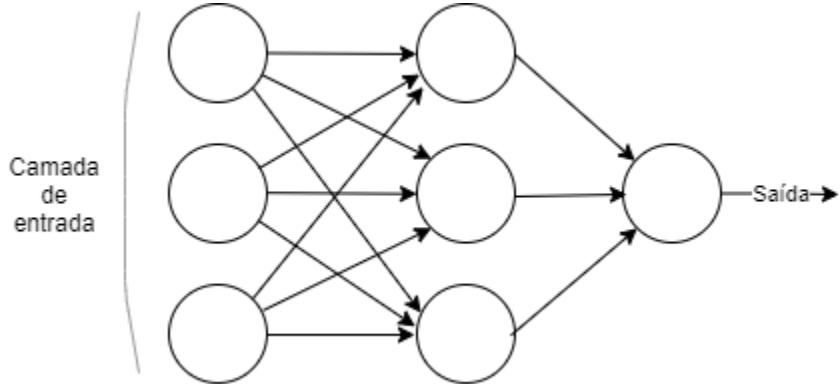


Figura 2.3: Modelo de Perceptrons.

MLPs são redes neurais artificiais nas quais podemos distinguir a camada de entrada, as camadas ocultas e a camada de saída. Como pode ser visto na Figura 2.3, cada camada é formada por neurônios, sendo os neurônios da camada de entrada chamados de neurônios de entrada e os neurônios da camada de saída chamados de neurônios de saída [21]. A rede da Figura 2.3 possui apenas uma camada oculta, mas é possível ter redes com múltiplas camadas ocultas, nomenclatura utilizada para indicar uma camada que não é de entrada ou de saída. Um exemplo de um desses casos é a Figura 2.4.

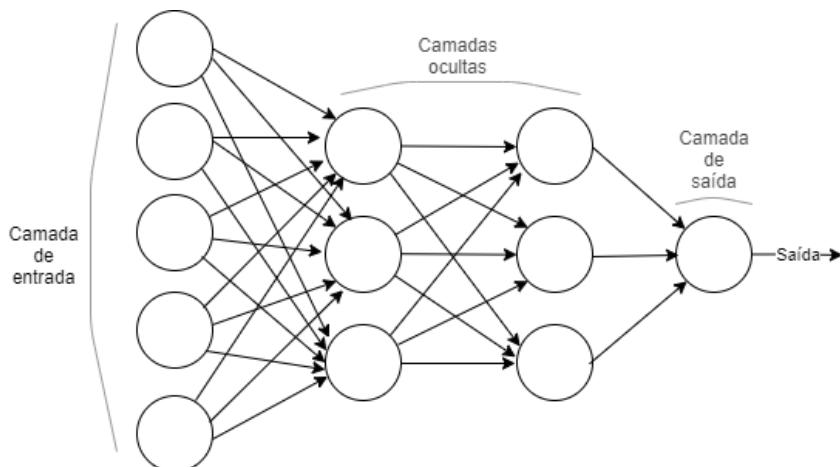


Figura 2.4: Modelo de Perceptrons Multicamadas.

As redes apresentadas até o momento têm camadas totalmente conectadas, uma vez que todos os neurônios estão totalmente conectados a todas as ativações na camada posterior [22]. Além disso, as arquiteturas de redes apresentadas anteriormente são chamadas redes neurais *feedforward*, ou seja, a saída de uma camada é usada como entrada para a próxima camada [23]. Isso quer dizer que não há *loops* na rede e, portanto, as informações sempre são alimentadas na direção da saída e nunca enviadas de volta para as camadas anteriores. Entretanto, existem outros modelos de redes neurais artificiais em que os circuitos de *feedback* são possíveis, como as redes neurais recorrentes. Outro tipo de redes neurais do tipo *feedforward* são as redes neurais convolucionais que serão melhor explicadas nas Seção 2.3.

## 2.3 Redes Neurais Convolucionais

Diferente do modelo de rede neural de MLPs apresentado na Seção 2.2.2, as CNN apresentam uma camada de convolução no lugar de uma camada totalmente conectada em pelo menos uma camada da rede [22]. Em uma CNN, cada camada aplica um conjunto de filtros para extração de características e combina os resultados para as camadas posteriores [22]. As vantagens de utilizar CNN são: (i) a invariância local, que permite classificar uma imagem contendo um determinado objeto, independentemente de onde o objeto aparece na imagem; e (ii) composicionalidade que faz com que seja possível compor filtros com o objetivo de aprender características mais complexas na imagem em camadas mais profundas da rede [22].

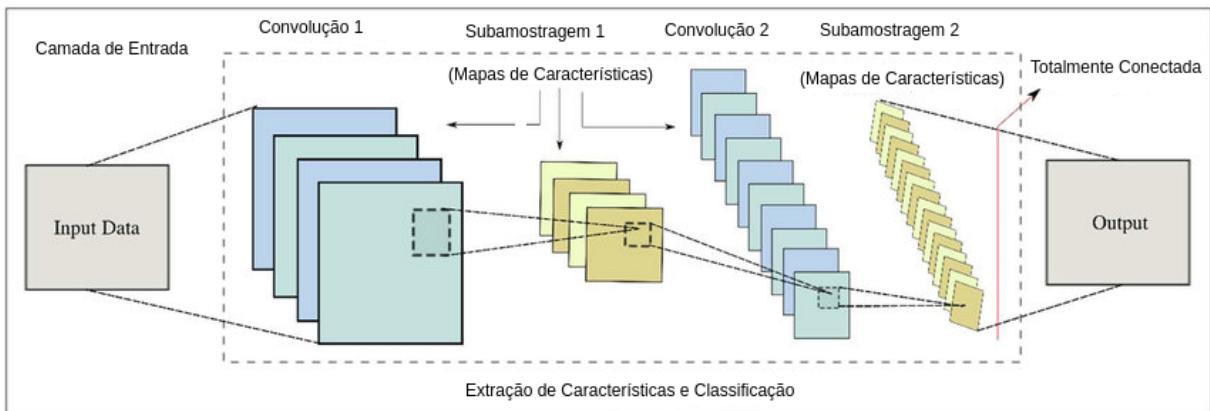


Figura 2.5: Arquitetura geral de uma Rede Neural Convolucional (adaptada de Haleem et al. [3]).

A visão geral de uma CNN pode ser vista na imagem Figura 2.5 onde é possível perceber a existência de camadas de convolução, *pooling*, chamada de *downsampling* (su-

bamostagem), e por fim uma camada totalmente conectada. Nas próximas subseções serão apresentadas as camadas de convolução, ativação, *pooling*, e seus papéis na arquitetura. Todavia, a fim de preparar o leitor, será introduzido antes os conceitos de convolução e *Kernel*, que são necessários para compreender os tipos de camadas que compõem uma CNN.

### 2.3.1 Operação de convolução em imagens

De forma geral, convolução é uma operação sobre duas funções que possuem como argumento um número real [24]. No contexto de CNNs aplicada a imagens, a entrada da CNN, denotada por *Input Data* na Figura 2.5 são *arrays* multidimensionais, dessa forma, a operação de convolução, denotada pelo operador  $*$ , pode ser descrita pela Equação 2.3, onde  $I$  e  $K$  são *arrays* multidimensionais [24].

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \quad (2.3)$$

O resultado da operação, denotado por  $S$ , é denominado mapa de características.

### 2.3.2 *Kernels* de convolução

Na Equação 2.3, o *array*  $I$  representa a imagem de entrada da CNN, enquanto o *array*  $K$ , representa o *Kernel* da operação, também chamados de *kernels* de convolução. Os *kernels* funcionam como filtros que são utilizados para capturar padrões na imagem, como por exemplo, bordas [22].

### 2.3.3 Camada de Convolução

A camada de convolução possui um papel central para a extração de características [13], e é onde ocorre a operação de convolução. Essa camada consiste de um conjunto de *kernels* que serão aplicados ao longo de toda dimensão da imagem de entrada da CNN. Após a aplicação dos filtros, os mapas de ativação obtidos servirão de entrada para a próxima camada da CNN [22].

### 2.3.4 Camada de Ativação

Situadas após a camada de convolução, a camada de ativação é responsável por aplicar uma função de ativação. Também chamada de funções de transferência, as funções de ativação, são funções matemáticas que são aplicadas a cada neurônio da rede, tanto das camadas ocultas, quanto na camada de classificação. O resultado da função é repassado

para o próximo neurônio na rede com o objetivo de medir a relevância do neurônio para a classificação. Essas funções devem possuir algumas propriedades, tais como, serem não-lineares, contínuas, diferenciáveis, centradas no zero, e com baixo custo computacional [25].

Existem diferentes funções de ativação que podem ser usadas em uma rede neural, uma das mais utilizadas é a função *Rectified Linear Unit* (ReLU) [22], definida pela Equação 2.4, que retorna o máximo entre 0 e um valor de entrada  $x$ .

$$f(x) = \max(0, x) \quad (2.4)$$



Figura 2.6: Gráfico da função de ativação ReLU.

A partir do gráfico apresentado na Figura 2.6, é possível concluir que a função ReLU satisfaz as propriedades citadas anteriormente e por isso é utilizada nas camadas ocultas da maioria das CNNs modernas [17].

Outra função de ativação que pode ser utilizada principalmente em problemas de classificação binária é a função Sigmoide, descrita pela Equação 2.5. O seu uso é recomendado como função de ativação na última camada da rede, pois fornece um valor de saída entre 0 e 1 [17].

$$s(x) = \frac{1}{1 + e^{-x}} \quad (2.5)$$

A partir da Equação 2.5, é possível construir o gráfico da função sigmoide, apresentado na Figura 2.7.

### 2.3.5 Camada de *Pooling*

Os mapas de características obtidos na camada de convolução, evidenciam as características filtradas pelo *kernel*. Todavia, os mapas podem causar o problema de *overfitting*,

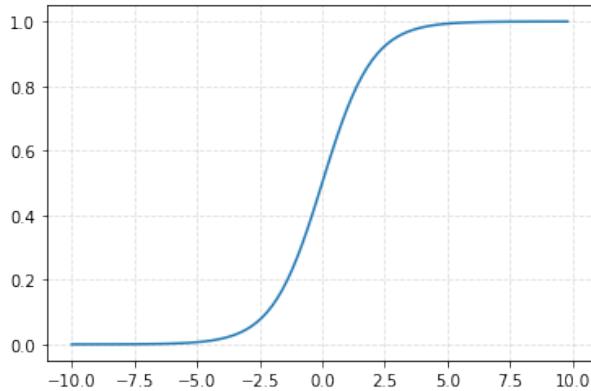


Figura 2.7: Gráfico da função de ativação Sigmoid.

uma vez que as características no mapa são sensíveis a posição dos *pixels* na imagem de entrada [13]. Dessa forma, pequenas variações na imagem produzem um novo mapa, com as características em posições diferentes. Uma forma de evitar esse problema é tornar os mapas mais invariantes a pequenas modificações na imagem. Dessa forma, temos mapas que indicam a existência de uma característica em uma região, e não em *pixels* específicos [24]. Isso é feito na camada de *pooling*, por meio da operação de *pooling*, que consiste em aplicar uma função de *pooling* em uma região do mapa de características obtidos na camada de convolução, e tem como objetivo produzir uma resumo das características da região. Uma possível função de *pooling* é a função *max-pooling*, que retorna o valor máximo em uma vizinhança do mapa de características. Também é possível usar funções que retornam a média, ou a média ponderada da vizinhança [24].

## 2.4 Arquiteturas de Redes Neurais Convolucionais

Desde a primeira arquitetura de RNA multicamada, a ConvNet proposta por LeCun et al. [26], houve um grande aumento na quantidade de arquiteturas existentes. Passou-se por um período de estagnação no início dos anos 2000 e, posteriormente em 2012, ganhou-se bastante força por conta da disponibilidade de grandes quantidades de dados e melhora nas tecnologias de *hardware* [7].

Dentre os principais pontos de aprimoramento com as novas arquiteturas que surgiram ao longo dos anos, pode-se citar a melhora na otimização de parâmetros da rede, regularização e reformulação estrutural. Entretanto, segundo Khan et al. [7], é possível observar que o impulso principal para melhora de performance das RNAs veio da reestruturação das unidades de processamento e do desenho de novos blocos da rede. Nesta seção é explicado um pouco mais sobre algumas arquiteturas específicas.

### 2.4.1 AlexNet

Criada por Krizhevsky et al. [4], a AlexNet foi considerada a primeira arquitetura de CNNs Profundas. Em comparação com a rede anterior mais conhecida chamada LeNet, criada por LeCun et al. [27], a AlexNet estendeu a profundidade de 5 (LeNet) para 8 camadas para fazer a rede ser aplicável para uma categoria diversa de imagens.

Além do aumento da profundidade da rede, uma das principais contribuições da AlexNet foi o uso da função de ativação *ReLU* para melhorar a taxa de convergência amenizando o problema de dissipação do gradiente [28]. Além disso, a arquitetura fez o uso da técnica de *Dropout* e camadas sobrepostas de *max pooling* para reduzir a chance de superajuste dos dados de treinamento, já que uma rede maior favorece essa chance. Na Figura 2.8 é mostrada uma representação da arquitetura AlexNet com cinco camadas convolucionais e três camadas totalmente conectadas.

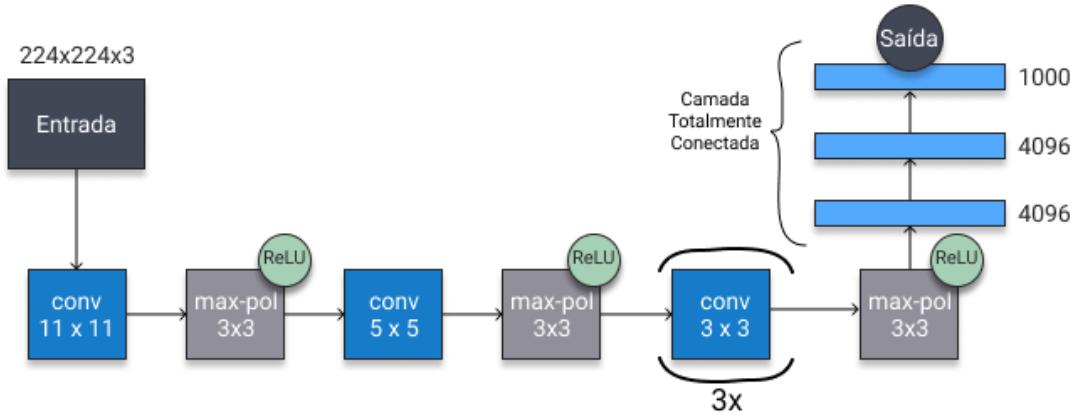


Figura 2.8: Representação da arquitetura AlexNet (baseada em Krizhevsky et al. [4]).

### 2.4.2 VGG

Essa arquitetura foi introduzida por Simonyan e Zisserman [29] e trouxe o desenho de uma rede ainda mais profunda que a AlexNet. O modelo VGG16 possui 13 camadas convolucionais e 3 camadas totalmente conectadas, utilizando entre elas a função de ativação *ReLU*. Também foi desenhada em uma versão variante mais profunda, chamada VGG19.

Essa arquitetura trouxe camadas utilizando o padrão modular, composta por blocos com um número incremental de camadas convolucionais com *kernels* de tamanho  $3 \times 3$ . O uso de *kernels* de tamanho menor do que os normalmente usados até então, de  $11 \times 11$  e  $7 \times 7$ , fornece o benefício adicional de custo computacional de menor complexidade por reduzir o número de parâmetros da rede. Além disso, para reduzir o tamanho dos

mapas de ativação obtidos e regular a complexidade da rede, blocos de *max-pooling* são colocados entre os blocos de convolução, reduzindo o tamanho dos mapas de ativação pela metade. A Figura 2.9 mostra uma representação da arquitetura VGG16 com 13 camadas convolucionais e 3 camadas totalmente conectadas em conjunto com a *ReLU*, seguida da função *softmax* [30] responsável pela classificação.

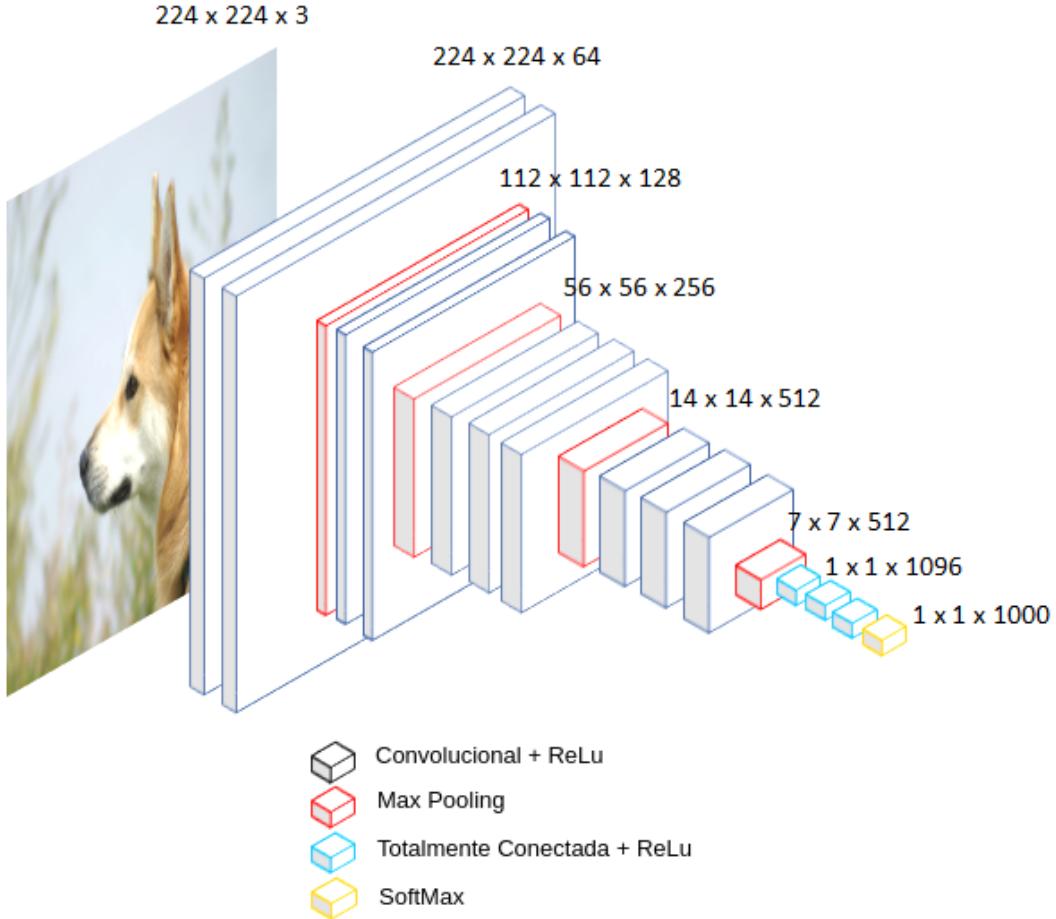


Figura 2.9: Representação da arquitetura VGG16 (adaptada de Roman [5]).

### 2.4.3 Inception

Arquitetura introduzida em Szegedy et al. [6], conhecida como GoogleNet ou Inception V1. Usa blocos com filtros de diferentes tamanhos ( $1 \times 1$ ,  $3 \times 3$  e  $5 \times 5$ ) que são então conectados para extrair características em diferentes escalas. É um modelo de rede inspirado pelo LeNet, mas que implementou um novo conceito: o módulo incepção.

O módulo de incepção é baseado em várias convoluções pequenas com o objetivo de drasticamente reduzir o número de parâmetros e facilitar o treinamento. A Inception V1 possui uma arquitetura de 22 camadas, mas reduziu o número de parâmetros de 60 milhões

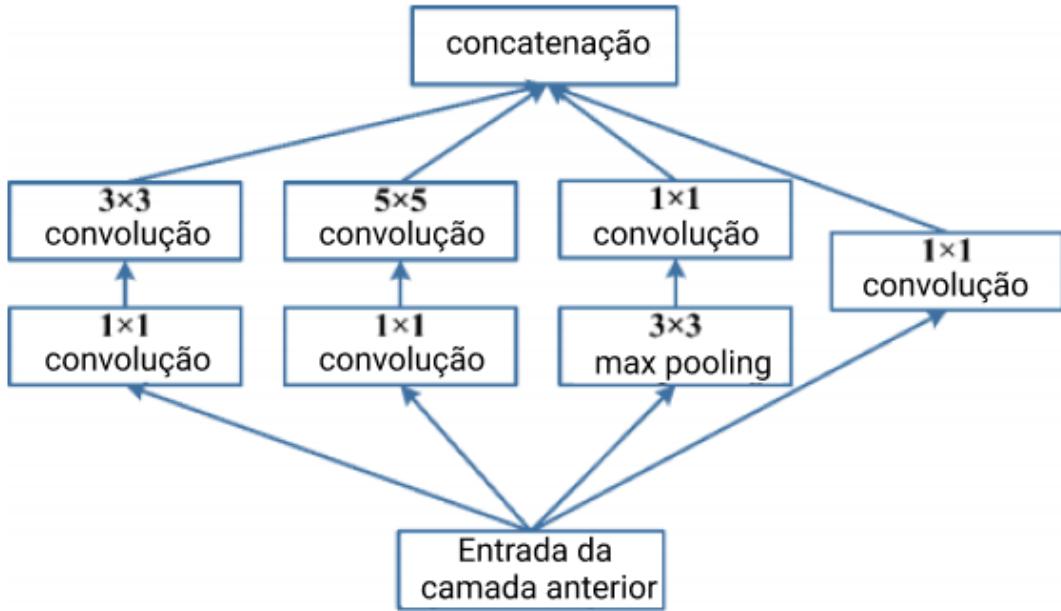


Figura 2.10: Módulo incepção baseado em Szegedy et al. [6] (adaptado de Khan et al. [7]).

(AlexNet) para 4 milhões. Além disso, essa arquitetura requer ainda menos memória que a VGG. Na Figura 2.10 pode ser vista uma representação do módulo incepção descrito anteriormente.

#### 2.4.4 ResNet

A arquitetura ResNet [8], introduziu o conceito de aprendizado residual em CNNs profundas. Através dos caminhos explorados por He et al. [8], foi mostrado que a ResNet obteve menos erros em tarefas de classificação de imagens com até 152 camadas em comparação com uma rede simples de 34 camadas. O problema existente em treinar redes muito profundas foi aliviado com a introdução de uma nova camada da rede chamada de bloco residual, mostrado na Figura 2.11. Além disso, essa rede apresentou um foco pesado em normalização de lotes.

### 2.5 Transfer Learning

*Transfer Learning* é uma técnica de aprendizado de máquina que pode ser utilizada em cenários em que a quantidade de dados disponível para treino é insuficiente, e tem como objetivo, transferir o conhecimento adquirido entre domínios [31].

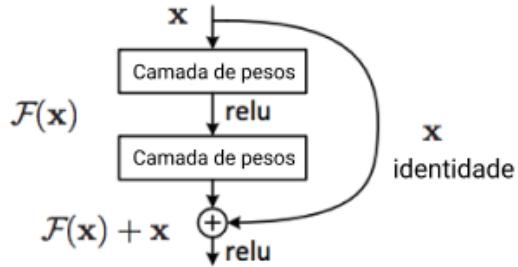


Figura 2.11: Bloco Residual (Adaptado de He et al. [8]).

Essa técnica também pode ser aplicada em redes neurais, no contexto de aprendizagem profunda. Nesse caso, tem como objetivo utilizar uma rede pré treinada em um domínio, e reaproveitar, a estrutura e os parâmetros, como parte de uma nova rede neural para um novo domínio. A construção da nova rede pode ser feita utilizando as camadas frontais de uma rede treinada com uma base de dados de grande escala, e adicionando e treinando somente as camadas finais [32].

## 2.6 Avaliação de desempenho de modelos

Após treinar um modelo, é necessário avaliar o seu desempenho. Sendo assim, será apresentado algumas métricas que podem ser utilizadas para medir a performance em problemas de classificação binária.

### 2.6.1 Matriz de confusão

Seja  $p$  e  $n$  as classes positivas e negativas que um elemento pode assumir, e  $T$  e  $F$  as classes fornecidas por um classificador binário, as seguintes combinações podem acontecer:

1. um elemento é classificado como  $T$ , e pertence a classe  $p$  (Verdadeiro Positivo)
2. um elemento é classificado como  $F$ , e pertence a classe  $p$  (Falso Negativo)
3. um elemento é classificado como  $T$ , e pertence a classe  $n$  (Falso Positivo)
4. um elemento é classificado como  $F$ , e pertence a classe  $n$  (Verdadeiro Negativo)

A matriz de confusão, mostrada na Figura 2.12, é uma tabela construída a partir dessas 4 combinações, e indica a quantidade de ocorrências de cada combinação sobre os elementos de um *dataset* [33]. A partir da matriz de confusão, é possível computar outras métricas para avaliar o modelo, como acurácia, precisão, revocação e *F1-Score*, como descrito abaixo.

		Predição		
		P	n	total
Rótulo verdadeiro	p'	TP	FN	P'
	n'	FP	TN	N'
total		P	N	

Figura 2.12: Matriz de Confusão.

## Acurácia

Definida pela Equação 2.6, a acurácia mede a taxa de classificações corretas pelo total de classificações realizadas. Um dos problemas da acurácia é quando o *dataset* está desbalanceado Vakili et al. [34].

$$\text{Acurácia} = \frac{TP + TN}{P + N} \quad (2.6)$$

## Revocação

A revocação, ou *Recall*, é definida pela Equação 2.7, e indica a capacidade do modelo de classificar elementos da classe positiva de forma correta.

$$\text{Revocação} = \frac{TP}{TP + FN} \quad (2.7)$$

## Precisão

A precisão, definida pela Equação 2.8 indica a taxa de verdadeiros positivos, pelo total de positivos classificados pelo modelo.

$$\text{Precisão} = \frac{TP}{TP + FP} \quad (2.8)$$

## F1-Score

Também conhecida como F-Score, essa métrica é definida pela Equação 2.9, e apresenta a média harmônica entre a precisão e a revocação. O F-Score é útil pois leva em consideração tanto a revocação quanto a precisão, dando uma visão mais robusta do desempenho[35].

$$F1 = 2 * \frac{\text{precisão} * \text{revocação}}{\text{precisão} + \text{revocação}} \quad (2.9)$$

# Capítulo 3

## Revisão da Literatura

Nos últimos anos, uma grande quantidade de pesquisas foram conduzidas para problemas de detecção de tipos de documentos, assim como do conteúdo dos mesmos, resultando na proposta de promissores métodos de reconhecimento de documentos de identidade. Portanto, neste capítulo é apresentada uma breve revisão da literatura sobre reconhecimento de classes de documentos de identidade utilizando desde técnicas mais clássicas de segmentação de imagem até os que utilizam Redes Neurais Convolucionais, *Rede Neural Convolucional* (CNN). A Seção 3.1 trata de artigos que mostram algumas técnicas que foram aplicadas na classificação de imagens de documentos e a Seção 3.2 explica a relevância desse trabalho.

### 3.1 Técnicas de classificação de imagens de documentos

Aplicações baseadas em segmentação de imagens são essenciais porque elas servem como técnicas de pré-processamento para serem utilizadas por algoritmos de ROC e outras aplicações de imagens. No trabalho de Neves Junior et al. [10] é proposta uma rede neural convolucionar rápida e totalmente oitava *octave* completa chamada *OctHU-PageScan*, baseada na arquitetura *U-Net*, com o objetivo de segmentar imagens de documentos de identidade para detectar as bordas do documento e texto. Os dois conjuntos de dados utilizados possuíam juntos mais de 20.000 documentos de identidade brasileiros com diferentes fundos de ambientes do mundo real, com padrões não uniformes, consistindo em diferentes cores, texturas, condições de luz e diferentes resoluções de imagens. Para cada imagem do conjunto de dados foi feito uma máscara, utilizada como base da verdade, que indicava os pontos do documentos em que havia texto. Através das convoluções *octave* a arquitetura proposta conseguiu diminuir o tempo de inferência em 4,6 pontos percentuais,

o armazenamento de espaço em 93,49% e o número de parâmetros consideravelmente em relação a arquitetura U-Net, ainda que sua acurácia tenha sido reduzida aproximadamente 4,6 pontos percentuais.

Tavakolian et al. [36] conduziram um estudo para melhorar o tempo de processamento e a acurácia, no processo de recuperação de informação em imagens de documentos pessoais, em particular, documento de identidade e passaporte. De acordo com os autores, a recuperação de informação baseada em Reconhecimento Óptico de Caracteres, *Optical Character Recognition* (OCR), apresenta relação com a precisão da segmentação de texto do documento. Dessa forma, os autores compararam as métricas de acurácia e tempo de processamento, nos seguintes estágios que antecedem o OCR: verificação de rotação na imagem, detecção e extração de face, *auto-cropping* e segmentação de texto. De acordo com os autores, a fase de verificação de rotação na imagem é necessária, pois a maioria dos documentos não estão na posição horizontal e isso apresenta impactos negativos no resultado final. O método proposto pelos autores para tratar do problema da detecção de rotação, utiliza as técnicas *Fast Fourier Transform* (FFT), *Mean Squared Error* (MSE), transformada de Hough e Detector de bordas de Canny, obtendo 96% de resultados positivos em relação a correção da rotação da imagem. Na etapa de extração de face foi comparado métodos usando HAAR/*Histogram of Oriented Gradients* (HOG) e aprendizagem profunda com arquitetura Resnet-10, e concluem que a abordagem usando aprendizagem profunda é mais rápida e apresenta acurácia de 96,3%, para reconhecimento da face. Na etapa de *auto-cropping* foram comparados dois métodos. Um proposto pelos autores, e que consiste na detecção de regiões com maiores detalhes, por meio da identificação de uma região ótima, e um segundo método baseado na identificação de propriedades de contorno. Os autores apenas dizem que obtiveram um tempo de processamento de 0,7 segundos usando o segundo método. Por fim, na fase de segmentação de texto é comparado os métodos *Efficient and Accurate Scene Text* (EAST) e *Root-Mean-Square Error* (RMSE). Os autores concluem que o método EAST, que é baseado em aprendizagem profunda, é mais rápido.

No trabalho de Mothes et al. [37] foi feito o uso de CNN para o reconhecimento de caracteres, por meio de segmentação e classificação, por meio de imagens de documentos de identidades, ou seja, foi criado um *Optical Character Recognition* (OCR) de documentos de identidade. O experimento foi conduzido utilizando 20000 imagens sintéticas para treino, e 2000 para testes para cada uma das 74 classes. Os resultado da acurácia foi comparado para as seguintes arquiteturas: LeNet, com 84%, CifarNet com 94%, Resnet-10 com 97% e por fim, 98% usando a arquitetura Resnet-20, além de contar com um sistema iterativo auto-supervisionado de dados que usa os dados dos documentos de identidade para melhorar a acurácia.

O uso de documentos pessoais no formato digital por serviços como bancos, e companhias de transportes, motivou Popreshnyak et al. [11] a desenvolverem um sistema para identificação de documentos pessoais com o objetivo de automatizar reconhecimento de informações pessoais. Usando técnicas de aprendizagem profunda, os autores desenvolveram uma rede neural com capacidade de classificar uma imagem passada como entrada em uma das quatro classes: passaporte, documento de identidade, certidão de nascimento e carteira de habilitação. Para isso, os autores usaram uma CNN com função de ativação ReLU. De acordo com os autores, o uso de CNN para esse caso é mais indicado do que outros métodos disponíveis na literatura, como Mapas de Kohonen, uma vez que os mapas impõem restrições ao tamanho da imagem, o que nem sempre é possível, pois as imagens são, em sua maioria, adquiridas por possíveis clientes, e podem conter erros, como baixo contraste. Para treinar a rede e evitar o problema de *overfitting*, os autores dividiram o conjunto de treino, contendo 30 imagens em cada classe, em outros dois subconjuntos, treino e validação, na proporção de 80/20, respectivamente. Após o treinamento, os autores conseguiram um acurácia de 85%.

Simon et al. [38] conduziram um estudo com o objetivo realizar extração automática das seguintes informações de documentos pessoais: país emissor, tipo do documento e a versão do documento. De acordo com os autores, o uso de um OCR não é uma opção viável, pois os documentos apresentam uma grande diversidade de *layouts*, texturas de fundo, e até mesmo idiomas que impactam negativamente na extração das informações, uma vez que apresentam fontes diferentes. Dentre os documentos utilizados pelos autores estão: documentos de identidade, passaporte e carteira de habilitação, tanto a parte frontal quanto verso. A motivação é que, com essas informações, é possível consultar em um banco de dados o *layout* do documento, e consequentemente, extrair as demais informações do mesmo. Os autores analisam algumas técnicas de classificação, já disponíveis na literatura, e suas respectivas acurárias, tais como: *Pyramid Histogram of Oriented Gradients* (PHOG) com 91,6%, *Colormame* com 91,6%, e HOG com 92,7%. Além disso, os autores também fazem um experimento usando CNN com arquitetura *ImageNet*, com apenas 1 imagem de cada uma das 74 categorias de 35 países para treino, e as demais para teste. Além disso, a rede não foi totalmente treinada, devido a falta de imagens, uma vez que as imagens possuem informações pessoais. Os resultados da acurácia obtida com a rede neural foi de 96,5%, e da *Area Under the ROC Curve* (AUC-ROC) de 0,986. Os melhores resultados para a acurácia obtidos na pesquisa, entretanto, foi de 97,7%, por meio da combinação das técnicas PHOG, HOG, e *Colormame*.

No trabalho de Sicre et al. [12] é feita a comparação entre duas abordagens para classificar documentos de identidade. Uma das abordagens utiliza três passos para classificar: i) extração de características locais através de *Scale Invariant Feature Transform* (SIFT);

ii) *Encoding* dos descritores de imagem local e *pooling* desses descritores em uma representação global da imagem usando *Bag of Words* (BOW), *Vector of Locally Aggregated Descriptors* (VLAD) ou *Fisher Vectors*; e iii) treinamento e classificação de descritores de imagens para reconhecimento de objetos na imagem através de uma *Support Vector Machine* (SVM). A outra abordagem utiliza CNN para obter as características da imagem e o uso de SVM para classificação. Essas abordagens foram comparadas utilizando imagens de documentos de identidade franceses e belgas adquiridas de diferentes dispositivos com fundos e rotações diferentes. O método utilizando CNN teve resultados superiores com 95.6% de acurácia média por classe e acurácia média de 99.5%.

Os autores Awal et al. [39] propuseram um método para classificação de imagens de documento usando documento de identidade de diversos países em duas etapas. Na primeira etapa é feita a criação de modelos de referência para cada classe de documento. Isso é feito através do mascaramento manual de zonas variáveis do documento, como dados pessoais e fotos, e então é usado o método de extração de descritores locais *Speeded Up Robust Features* (SURF) para extrair os pontos-chave do documento. A segunda etapa consiste na classificação de documentos de identidade fazendo um casamento de padrões entre seus descritores de pontos-chave e o dos modelos de referências. O tipo do documento, é classificado de acordo com o modelo em que tiver a maior quantidade de pontos-chave em comum. O método proposto mostrou acurácia de 95,8% em comparação com um método que usa CNN e SVM.

## 3.2 Considerações Finais

A partir dos trabalhos estudados, foi possível entender diversas técnicas já aplicadas em diferentes contextos de classificação de imagens de documentos de identificação, desde pré-processamento de imagens, técnicas de segmentação de imagens, abordagens de reconhecimento ótico de caracteres, até modelos de classificação que façam uso de redes neurais convolucionais e diversas maneiras para avaliação de performance. Entretanto, a maior parte dos trabalhos estudados explora a classificação de tipos de documentos de identidade ou extração dos dados presentes nesses documentos e não foi encontrado na literatura um trabalho que lide apenas com a etapa de classificação.

Esta pesquisa pretende comparar arquiteturas de redes neurais convolucionais como classificadores em um conjunto de dados de documentos de identidade, disponibilizado por uma empresa. Os classificadores binários avaliam entre boa ou má qualidade da imagem de documento, segundo critérios pré-definidos, sendo assim uma contribuição para estudos futuros.

# Capítulo 4

## Metodologia Proposta

Este trabalho pretende comparar arquiteturas de redes neurais convolucionais como classificadores em um conjunto de dados de documentos de identidade, disponibilizado por uma empresa que não deseja ser identificada. Os classificadores binários avaliam entre boa ou má qualidade da imagem de documento, segundo critérios pré-definidos. Neste capítulo será apresentada a metodologia aplicada no presente trabalho. A Seção 4.1 apresenta a base de dados que foi utilizada para execução dos experimentos. A Seção 4.2 apresenta as arquiteturas que foram consideradas. Por fim, as Seções 4.2, 4.3, 4.4, 4.5 e 4.6 tratam a respeito das variações escolhidas para executar e comparar os resultados.

A Figura 4.1 ilustra o *pipeline* contendo as etapas presentes para os experimentos para cada arquitetura de *Redes Neurais Convolucionais* (CNNs) escolhida. Em um primeiro momento, foi utilizada a base de dados classificadas por uma empresa privada. Então, foram consideradas arquiteturas de redes neurais convolucionais para classificar a qualidade de documentos. Por fim, com objetivo de comparar os resultados das diferentes arquiteturas foram feitos experimentos utilizando-se *Transfer Learning* e variando *Data Augmentation*, *Dropout* e *Fine-Tuning*. Cada etapa do *pipeline* é explicada em detalhes nas seções desse capítulo, enquanto os resultados são discutidos na Seção 5.7.

### 4.1 Descrição da base de dados

A base de dados utilizada é constituída de 2.438 imagens de documentos pessoais obtidas diretamente do ambiente de produção de uma empresa privada e que exigiu anonimato durante a concessão, portanto, não é uma base de acesso público.

A base consiste em imagens de três categorias:

1. Documentos de identidade frente;
2. Documentos de identidade verso;

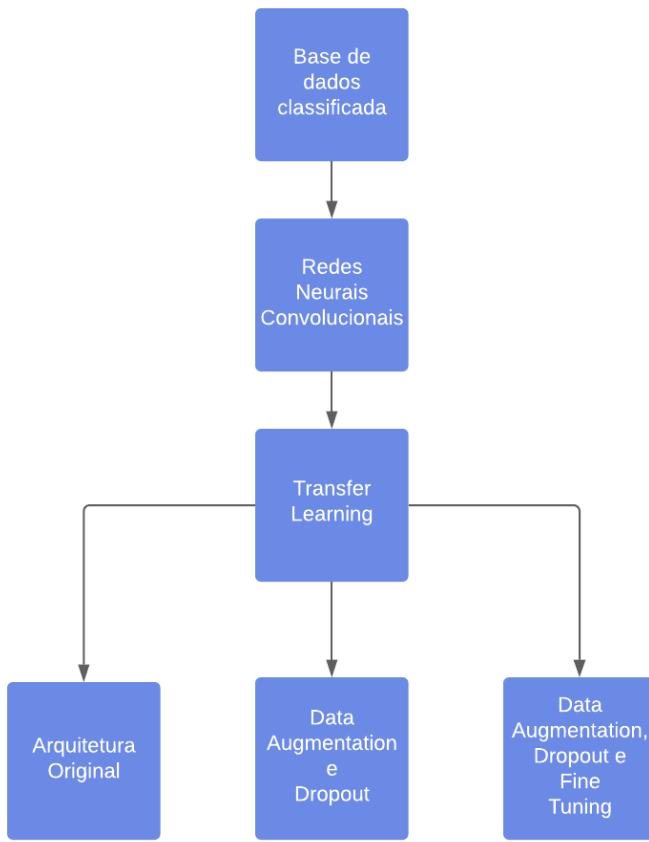


Figura 4.1: Etapa das da metodologia proposta.

### 3. Carteiras Nacional de Habilitação.

As imagens da Categoria 1 consistem da parte frontal de fotos de documentos de identidade, também conhecidos como Registro Geral (RG). As imagens da Categoria 2 consistem da parte traseira do RG, e por fim, as imagens da Categoria 3 consistem de fotos de Carteira Nacional de Habilitação (CNH).

Considera-se como parte frontal do RG a parte relativa a foto do dono do documento, e traseira a parte relativa às informações do dono do documento sem a foto. Devido ao modelo do documento, as imagens relativas a CNH não possuem essa divisão uma vez que é possível capturar todo o documento em uma única foto. A Figura 4.2 apresenta quatro amostras da base que foram definidas como boas. Amostras de imagens de RG e CNH classificadas como ruim também são apresentadas na Figura 4.3. Para que pudesse ser incluídas nesse trabalho, em todas imagens, as informações pessoais dos donos foram preservadas.

As imagens são capturadas por meio de diversos dispositivos, em sua maioria, dispositivos móveis, como celulares e tablets, assim como imagens de documentos escaneados.



(a) Foto boa de RG frontal

(b) Foto boa de RG traseira



(c) Foto boa de CNH

(d) Foto boa de CNH

Figura 4.2: Amostras de documentos classificados como bons.

Em todas as situações as imagens possuem qualidades, rotações, fundos e dimensões variáveis sendo todas essas determinadas pelos diversos contextos em que os usuários da empresa escolheram no momento de captura. Todas as imagens estão no formato JPEG.

A classificação da qualidade da imagem será feita entre duas classes: boa e ruim. Neste contexto, é definido qualidade ruim como imagens em que a leitura das informações



(a) Foto ruim de RG frontal



(b) Foto ruim de RG traseira



(c) Foto ruim de CNH



(d) Foto ruim de CNH

Figura 4.3: Amostras de documentos classificados como ruins.

do documento sejam prejudicadas por qualquer motivo. Alguns exemplos comuns são: imagem tremida, falta de iluminação e *flashes*, como pode ser visto imagens das Figuras 4.3a, 4.3b e 4.3c, respectivamente. Além disso, fotos que não mostrem o documento inteiro e documento ilegível pela distância da foto, como pode ser observado nas Figuras 4.3c e 4.3d, respectivamente. Pode-se observar que uma imagem pode ter mais de um quesito que a define como ruim, como no caso da Figura 4.3c em que o documento se encontra distante

Tabela 4.1: Quantidade de imagens por classe.

	<b>Treino</b>	<b>Validação</b>	<b>Teste</b>	<b>Total</b>
<b>Bom</b>	880	220	122	1222
<b>Ruim</b>	876	219	121	1216

e também possui *flash*. A imagem de qualidade boa possui um documento inteiro em que todas as informações estão legíveis sem nenhum efeito de foto ou iluminação impedindo a leitura delas, similar às amostras da Figura 4.2.

A Tabela 4.1 apresenta a divisão da base para cada categoria utilizada durante o treino, validação e teste para cada arquitetura de rede neural convolucional manuseada. Foi destinado 10% da base, totalizando 243 imagens, para teste. O restante, foi dividido na proporção de 80/20 para separar o *dataset* de treino e validação, totalizando 1.756 imagens para o conjunto de treino e 439 para validação.

Por fim, o rótulo de todas as imagens foram verificado e alterados de acordo com a necessidade, com o intuito de seguir os padrões definidos como bons e ruins anteriormente apresentados. Isso foi necessário pois a base original não estava classificada de acordo com o padrão estabelecido pela empresa. A base de dados foi definida de acordo com a seleção de imagens aleatórias da base original e analisando em qual classe ela se encaixa, de acordo os padrões apresentados anteriormente nesta seção.

## 4.2 Redes Neurais Convolucionais

Para realizar a classificação dos documentos, cinco modelos foram treinados, cada um utilizando uma arquitetura distinta a fim de comparar seus desempenhos. Em todos os modelos, foi utilizado a técnica de *transfer learning*. Arquiteturas de *Redes Neurais Convolucionais* (CNNs) foram consideradas no presente trabalho com base nos trabalhos de Popreshnyak et al. [11], Sicre et al. [12] e Awal et al. [39] onde os autores obtiveram resultados promissores na criação de modelos para de classificação de documentos com o uso de CNNs, embora nenhum deles tenham classificado imagens de documentos de identidade quanto a sua qualidade com os critérios iguais ou similares aos definidos na seção 4.1.

As arquiteturas comparadas foram: VGG16 [29], ResNet50 [8], Xception [40], InceptionV3 [41], MobileNetV2 [42] e DenseNet121 [43]. Esse conjunto de arquiteturas foi escolhido por terem trazido inovações arquiteturais que poderiam trazer diferentes impactos nos experimentos executados. Um resumo das contribuições de cada arquitetura pode ser encontrado na lista abaixo.

- VGG: topologia homogênea e uso de *kernels* menores;

- Inception: resolve o problema de um gargalo de representação e substitui filtros grandes por filtros menores;
- ResNet: aprendizagem residual e conexões de salto baseadas em mapeamento de identidade;
- DenseNet: fluxo de informações entre camadas;
- MobileNet: arquitetura leve e endereço de eficiência energética e portabilidade para plataformas embarcadas.

### **4.3 Transfer Learning**

Conforme apresentado na Seção 2.5, o uso *transfer learning* pode ser útil em cenários com pouca quantidade de imagens disponíveis na base de dados como no caso deste trabalho. Uma abordagem para utilizar *transfer learning* em CNNs é por meio de uma rede pré-treinada, reutilizando as representações aprendidas para treinar um novo classificador [17].

Dessa forma, cada arquitetura citada na Seção 4.2 foi carregada com os pesos pré-treinados na base de dados do ImageNet [44] e sem o classificador no topo da rede, funcionando assim como um extrator de características. Em sequência, um novo classificador foi adicionado ao topo de cada arquitetura e treinado com a base de dados da Seção 4.1. O ImageNet é um grande conjunto de dados de fotografias anotadas destinadas à pesquisas de visão computacional, composto por pouco mais de 14 milhões de imagens no conjunto de dados e pouco mais de 21 mil grupos de classes e foi escolhido para aplicar a técnica de *transfer learning* pela sua completude.

A Figura 4.4 apresenta uma visão geral desse processo, onde o classificador original foi treinado utilizando a base de dados Imagenet e a partir do processo de *transfer learning* é criado um classificador para classes específicas, bom e ruim, mantendo-se os pesos do classificador original nas camadas iniciais da rede neural convolucional e utilizando-se uma base relacionada as classes específicas para alterar os pesos das camadas finais da rede. Para treinar o novo classificador, as camadas internas da rede foram congeladas, para que as representações aprendidas no treinamento na base do ImageNet não fossem destruídas.

### **4.4 Data Augmentation**

*Data augmentation* é uma técnica amplamente utilizada para evitar o *overfitting* de modelos, sendo uma técnica particularmente interessante quando se tem uma baixa quantidade de imagens disponíveis na etapa de treinamento. O uso de *data augmentation* mostra

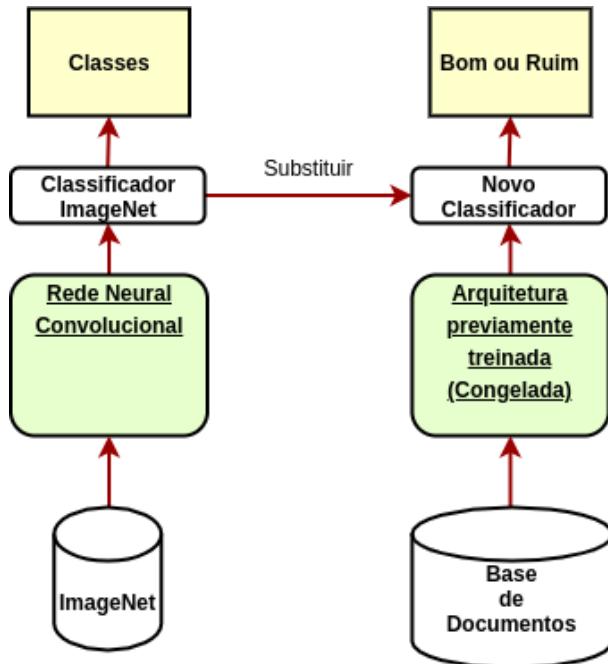


Figura 4.4: Demonstração do processo de *Transfer Learning* via extração de características.

aumentos de acurácia em tarefas de classificação de imagens usando CNNs segundo Perez [45], Mikolajczyk [46] e Gu [47]. Uma das preocupações no presente trabalho era de que a quantidade de imagens nos conjuntos de treino não fossem grandes os suficientes para conseguir um bom desempenho e, portanto, espera-se que o uso de *data augmentation* contribua para o melhor desempenho dos modelos deste trabalho que a utilizarem.

Dentre as várias possíveis combinações de métodos de *data augmentation*, foram considerados o giro e a rotação de imagens do conjunto de dados de treinamento, já que essas variações mantém a classe original de uma imagem, ao contrário por exemplo da técnica de redução de zoom que poderia fazer uma imagem da classe boa passar para a classe ruim por impossibilitar a legibilidade. Utilizando as implementações da biblioteca Keras<sup>1</sup>, durante cada época de treinamento, uma quantidade aleatória de imagens do conjunto de treinamento foram selecionadas e tiveram as transformações de giro e rotação aplicadas a essas, sendo feitos os giros de imagens no sentido horizontal e as rotações até 20% tanto no sentido horário quanto anti-horário em relação a imagem original.

---

<sup>1</sup><https://keras.io/>

## 4.5 Dropout

O *dropout* introduz regularização dentro da rede, o que acaba melhorando a generalização ao pular aleatoriamente algumas unidades ou conexões com uma certa probabilidade. Em CNNs, várias conexões que aprendem uma relação não linear às vezes são co-adaptadas, o que causa *overfitting* [48]. Esta dispensa aleatória de algumas conexões ou unidades produz várias arquiteturas de redes diluídas e, finalmente, uma rede representativa é selecionada com pequenos pesos. Essa arquitetura selecionada é então considerada uma aproximação de todas as redes propostas [49]. Essa técnica foi escolhida para o presente trabalho com o objetivo de trazer um possível ganho de desempenho nos modelos que a utilizarem, uma vez que a técnica pode ajudar a evitar *overfitting* e a quantidade de documentos na base de dados poderia não ser suficiente para o treinamento dos modelos escolhidos.

No presente trabalho a implementação de *dropout* utilizada foi a disponibilizada pela biblioteca Keras<sup>2</sup>. Os experimentos no presente trabalho que fizeram uso dessa técnica possuíam uma CNN com uma camada extra redefinindo entradas aleatoriamente para 0 com uma taxa de frequência de 20% a cada passo durante o treinamento. As entradas que não foram redefinidas para 0 foram aumentadas em  $1 / (1 - 0,2)$ , de forma que a soma de todas as entradas permaneceram inalterada, segundo a implementação da biblioteca Keras.

## 4.6 Fine-tuning

A técnica de *Fine-tuning* foi utilizada juntamente com as técnicas de *Data Augmentation* e *Dropout* durante o treinamento do modelo utilizando cada uma das cinco arquiteturas citadas anteriormente. Nesses cenários o uso dessa técnica permite adaptar as representações aprendidas por um modelo para um novo contexto, com a possibilidade de melhorar um pouco mais o desempenho do modelo [17]. No presente trabalho, a técnica de *Fine-tuning* se deu devido a pouca quantidade de imagens disponíveis na base e é esperado que seu uso possa trazer uma melhora no desempenho dos modelos utilizados.

---

<sup>2</sup><https://keras.io/>

# Capítulo 5

## Experimentos e Resultados

Neste capítulo são detalhados os experimentos e os resultados. Na Seção 5.1 é apresentado a linguagem de programação que foi utilizada, bem como as bibliotecas e demais ferramentas utilizadas para treinar os modelos. A Seção 5.2 apresenta os detalhes de implementação para garantir a redutibilidade dos experimentos. A Seção 5.3 apresenta os detalhes do *Hardware* utilizado para realização dos experimentos do presente trabalho. A Seção 5.4 apresenta o *Callback* utilizado durante o treinamento dos modelos. A Seção 5.6 apresenta os experimentos que foram realizados. Por fim, a Seção 5.7 apresenta os resultados experimentais.

### 5.1 Ambiente de Desenvolvimento Utilizado

O Google Colaboratory, ou Colab<sup>1</sup>, foi utilizado para o desenvolvimento do presente trabalho. É uma ferramenta do Google que permite que qualquer usuário escreva e execute código arbitrariamente em python atrás do navegador e é especialmente usado para aprendizado de máquina, e análise de dados. De maneira um pouco mais técnica, o Colab é um serviço de Jupyter Notebook<sup>2</sup>, hospedado em nuvem que não requer instalações prévias, enquanto provê acesso gratuito a recursos computacionais, incluindo *Graphics Processing Units* (GPUs).

O Colab foi utilizado para que não houvesse dependência de acesso físico a um computador para fazer os experimentos, mantendo-se assim sua reproduzibilidade, além da não necessidade de um computador com *hardware* específico para o bom desempenho dos experimentos. Outras vantagens decorrentes dessa escolha foram a facilidade de acesso a mesma versão do código entre os autores, além da facilidade em executá-lo sem a necessidade de preparar um ambiente de desenvolvimento local.

---

<sup>1</sup><https://colab.research.google.com/>

<sup>2</sup><https://jupyter.org/>

A principal biblioteca utilizada para guiar os experimentos foi o Keras<sup>3</sup>. O Keras foi desenhado para facilitar e abstrair a utilização de modelos de aprendizado profundo de maneira rápida, eficiente e através de uma interface simples e intuitiva, além de possuir uma boa documentação e diversos guias de desenvolvimento. Por fim, no início de 2019 foi feita uma enquete com times que ficaram entre os cinco primeiros lugares de qualquer competição Kaggle nos dois anos anteriores e o Keras foi ranqueado em primeiro lugar como biblioteca utilizada entre esses times dentro todas as bibliotecas e *frameworks* utilizados<sup>4</sup>.

As versões das bibliotecas utilizadas para o desenvolvimento desse trabalho são as mais atuais na configuração padrão do Colab atualmente, sendo a versão da linguagem *Python* 3.6.9, a versão do Keras 2.3.1 e a versão do NumPy 1.18.5. Todos os experimentos foram executados sob essas mesmas especificações.

## 5.2 Reprodutibilidade dos experimentos

As bibliotecas citadas na Seção 5.1 dão suporte para que seja criado um ambiente controlado, evitando-se o uso de valores aleatórios em suas implementações a cada nova rodada de experimentos. O suporte a reproduzibilidade nessas bibliotecas é dado através das sementes ou *seeds*. Esse controle foi feito via código inicializando-se todos os experimentos passando o mesmo valor de *seed* em todos os casos para que todos os experimentos reproduzidos tivessem o mais próximo possível de mesmo ambiente de execução.

## 5.3 Descrição do *Hardware*

Conforme descrito na Seção 5.1, todos os experimentos foram executados no ambiente Google Colaboratory e desta maneira, o *hardware* é definido de acordo com esse serviço. Utilizando a documentação do Google Colab<sup>5</sup> e levando em conta que foi usado o serviço gratuito sem nenhuma customização tem-se abaixo as especificações computacionais nas quais todos os experimentos foram realizados:

- GPU Tesla K80, frequência de clock de 0.82GHz, tendo 2496 CUDA cores, 12GB GDDR5 VRAM;
- Processador Intel(R) Xeon(R) CPU @ 2.20GHz, 2 cores, 2 threads por core;
- 64GB de espaço livre em disco rígido;

---

<sup>3</sup><https://keras.io/>

<sup>4</sup>[https://keras.io/why\\_keras/](https://keras.io/why_keras/)

<sup>5</sup><https://colab.research.google.com/>

- Memória RAM disponível de 13GB.

## 5.4 Callbacks

Em todos os experimentos foi utilizado um *callback* de *early stopping*<sup>6</sup> com o objetivo de evitar *overfitting*. Isso foi feito monitorando a métrica F1-Score e interrompendo o treinamento quando essa métrica parar de melhorar. Essa métrica foi escolhida pois é capaz de indicar quando a distribuição das classes está desbalanceada, conforme explicado na Seção .

O *callback* foi programado para ser executado ao fim de cada época do treinamento. A implementação utiliza 10 épocas de treinamento como intervalo para o modelo melhorar. Se ao fim desse intervalo o valor dessa métrica não aumentar, o treinamento é encerrado e os pesos obtidos na melhor época é restaurado.

## 5.5 Hiperparâmetros

Em todos os experimentos foi utilizado o otimizador Adam [50] juntamente com a função de *loss Binary Cross-Entropy*<sup>7</sup>. O valor da taxa de aprendizagem utilizada foi de  $1 \times 10^{-3}$ , somente sendo modificado para  $1 \times 10^{-5}$  no momento de aplicar a técnica de *Fine-Tuning*, quando é necessário reduzir a taxa de aprendizagem para evitar que o conhecimento da rede pré-treinada seja destruído [22].

## 5.6 Visão geral dos Experimentos

Para atingir os objetivos do presente trabalho, foram realizados três conjuntos de experimentos. Em todos os conjuntos, foi utilizado as arquiteturas *Xception*, *VGG16*, *ResNet50*, *DenseNet* e *MobileNet* com diferentes técnicas com o objetivo de comparar os desempenhos. A lista abaixo mostra as técnicas que foram utilizadas em cada conjunto:

- Conjunto 1: *Transfer Learning*;
- Conjunto 2: *Transfer Learning*, *Dropout* e *Data Augmentation*;
- Conjunto 3: *Transfer Learning*, *Dropout*, *Data Augmentation* e *Fine-Tuning*.

Em cada experimento, foi comparado as seguintes métricas: acurácia, precisão, revo-cação e F1-Score, apresentadas na Seção 5.4.

---

<sup>6</sup>[https://keras.io/api/callbacks/early\\_stopping/](https://keras.io/api/callbacks/early_stopping/)

<sup>7</sup>[https://keras.io/api/losses/probabilistic\\_losses/#binarycrossentropy-class](https://keras.io/api/losses/probabilistic_losses/#binarycrossentropy-class)

## Conjunto de experimentos 1

Esse primeiro conjunto de experimentos foi aplicada a técnica de *transfer learning* a cada uma das cinco arquiteturas citadas anteriormente. A Figura 5.1 exemplifica o fluxo geral do experimento, onde cada uma das arquiteturas da lista foi importada no Colab com os pesos da base do ImageNet e sem a camada densa no topo da rede, uma vez que essa camada funciona como o classificador da rede e a base do ImageNet não possui as classes boa e ruim apresentadas na Seção 4.1. Em seguida uma nova camada densa foi adicionada a essa rede pré-treinada, para iniciar um novo treinamento com a base da Seção 4.1. A função de ativação utilizada no novo classificador é a função Sigmoide.

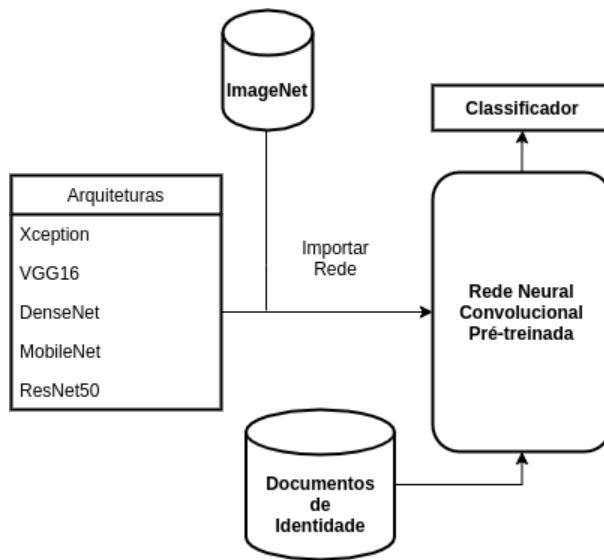


Figura 5.1: Visão geral do primeiro conjunto de experimentos.

Todas as arquiteturas foram programadas para serem treinadas por 50 épocas, todavia, devido ao uso do *callback Early Stopping* nem todos chegaram ao final das 50 épocas. Inicialmente nos testes que foram feitos, foi utilizado 999 épocas, junto com o *callback* de *early stopping*, com o objetivo de permitir o modelo treinar o máximo possível, todavia, todas as arquiteturas encerraram o treinamento entre 30 e 40 épocas. Dessa forma, foi utilizado 50 épocas em todos os experimentos realizados.

## Conjunto de experimentos 2

O segundo conjunto de experimentos consiste no uso da técnica de *transfer learning* combinada com *dropout* e *data augmentation*. A Figura 5.2 exemplifica o fluxo geral, onde cada uma das arquiteturas da lista foi importada no Colab com os pesos da base do ImageNet e sem a camada densa no topo da rede, uma vez que essa camada funciona como o classificador da rede e a base do ImageNet não possui as classes boa e ruim apresentadas

na Seção 4.1. Em seguida uma nova camada densa foi adicionada a essa rede pré-treinada, junto com uma camada de *Dropout* anterior a camada densa, para então iniciar um novo treinamento com a base da Seção 4.1. A função de ativação utilizada no novo classificador é a função Sigmoide.

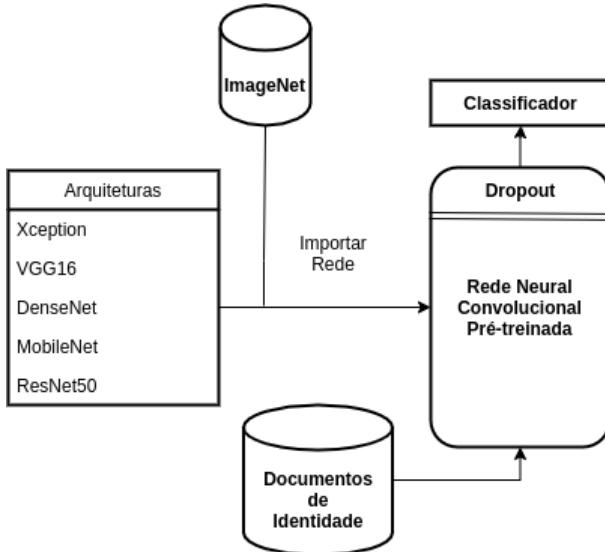


Figura 5.2: Visão geral do primeiro conjunto de experimentos 2.

### Conjunto de experimentos 3

O último conjunto de experimentos consiste no uso da técnica de *transfer learning* junto com *dropout*, *data augmentation* e *fine tuning*. A Figura 5.3 exemplifica o fluxo geral, onde cada uma das arquiteturas da lista foi importada no Colab com os pesos da base do ImageNet e sem a camada densa no topo da rede, uma vez que essa camada funciona como o classificador da rede e a base do ImageNet não possui as classes boa e ruim apresentadas na Seção 4.1. Em seguida uma nova camada densa foi adicionada a essa rede pré-treinada, junto com uma camada de *Dropout* anterior a camada densa, para então iniciar um novo treinamento com a base da Seção 4.1. Após o treinamento ser concluído, a técnica de *Fine-tuning* foi aplicada, resultando no treinamento de mais 10 épocas. A função de ativação do novo classificador é a função Sigmoide.

## 5.7 Resultados

A Tabela 5.1 apresenta os resultados da acurácia, revocação, F1-Score obtidos nos três conjuntos de experimentos. O tempo também foi adicionado, e indica em segundos o tempo que o modelo levou para classificar todas as 243 imagens do conjunto de teste.

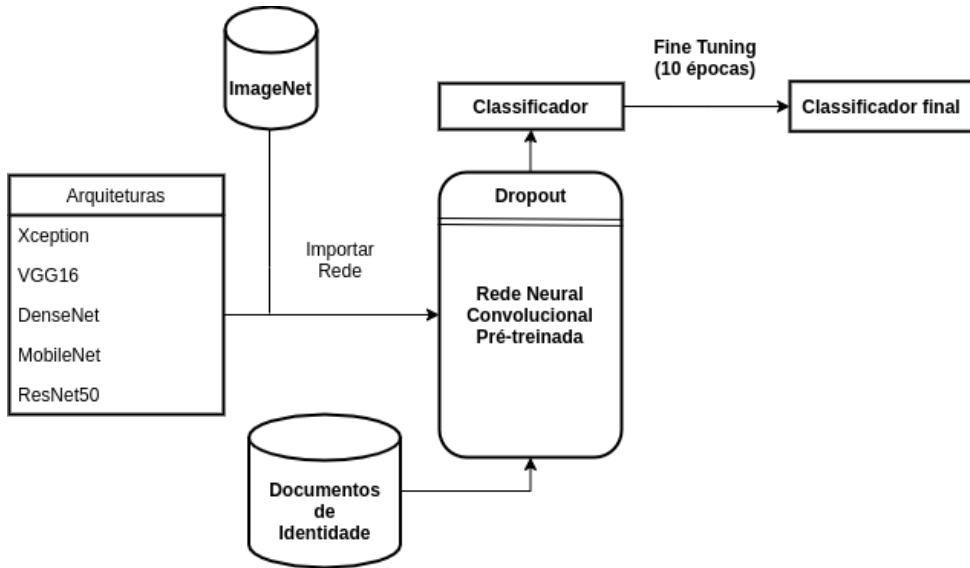


Figura 5.3: Visão geral do primeiro conjunto de experimentos 3.

Como apresentado na Seção 2.6, a revocação indica a capacidade do modelo de identificar elementos da classe positiva. No contexto desse trabalho, definimos como classe positiva as imagens ruins de documento, uma vez que apresenta características indesejáveis.

Tabela 5.1: Resultados para as combinações de arquiteturas de CNNs e variações de parâmetros especificadas em cada conjunto de experimentos.

Experimentos	Arquitetura	Acurácia	Revocação	Precisão	F1-Score	Tempo (s)
Conjunto de experimentos 1	VGG16	74,49%	82,49%	71,19%	76,18%	28,62
	ResNet50	<b>81,48%</b>	<b>87,61%</b>	<b>79,18%</b>	<b>82,92%</b>	33,17
	Xception	76,95%	83,81%	74,19%	78,44%	<b>27,56</b>
	DenseNet	78,60%	79,52%	75,39%	77,10%	29,01
	MobileNet	77,37%	81,86%	75,98%	78,28%	28,52
Conjunto de experimentos 2	VGG16	74,07%	87,19%	68,81%	76,70%	28,08
	ResNet50	74,90%	<b>95,64%</b>	66,74%	78,35%	28,07
	Xception	<b>75,72%</b>	90,09%	<b>70,84%</b>	<b>78,60%</b>	28,35
	DenseNet	74,90%	<b>95,64%</b>	66,20%	78,35%	27,76
	MobileNet	73,12%	93,61%	66,51%	77,48%	<b>27,03</b>
Conjunto de experimentos 3	VGG16	74,90%	88,98%	68,35%	77,19%	29,81
	ResNet50	76,13%	95,27%	70,01%	80,87%	28,17
	Xception	<b>80,66%</b>	89,72%	<b>74,84%</b>	<b>81,24%</b>	28,21
	DenseNet	74,03%	96,78%	66,52%	78,51%	28,27
	MobileNet	73,58%	<b>98,76%</b>	65,96%	78,76%	<b>26,93</b>

Nos resultados do primeiro conjunto de experimentos, descrito na Seção 5.6, é possível visualizar que a arquitetura ResNet50 possui a maior acurácia, com o valor de 81,48% no conjunto de teste, com a diferença de 2,88 pontos percentuais da arquitetura com a

Tabela 5.2: Variância do tempo de cada conjunto de experimento.

Conjunto	Variância
Conjunto 1	4,78
Conjunto 2	0,25
Conjunto 3	1,04

segunda maior acurácia desse conjunto, a DenseNet. Pode-se notar que a arquitetura com melhor resultado em revocação, precisão e F1-Score é a ResNet50, com 87,61%, 79,18% e 82,92%, respectivamente, embora tenha o tido o maior tempo de avaliação para as 243 imagens do conjunto do teste com 33,17s. O resultado de acurácia mais baixo nesse conjunto foi o da arquitetura VGG16 com a acurácia de 74,49%.

Dos resultados descritos na Seção 5.6 (conjunto de experimentos 2), tem-se que o maior valor de acurácia foi alcançado pela arquitetura Xception com 75,72%. Entretanto, o maior resultado de revocação pode ser visualizado na arquitetura DenseNet com o valor de 95,64%. Por fim, o resultado de F1-Score de valor mais alto para a o segundo conjunto de experimentos foi de 78,35%, também pela arquitetura DenseNet.

Nos resultados do conjunto de experimentos 3, também descrito na Seção 5.6, é possível notar que a maior acurácia foi obtida pela arquitetura Xception, assim como no conjunto do experimentos 2, porém dessa vez com o valor de 80,66%. A MobileNet foi a arquitetura com maior valor de revocação no conjunto de experimentos 3, com o valor 98,76%, embora essa arquitetura também tenha tido a acurácia mais baixa desse conjunto. Por fim, a arquitetura Xception teve o valor 81,24% de F1-Score, sendo esse o maior valor dessa métrica no conjunto 3.

Comparando-se os três conjuntos de experimentos, é possível visualizar que o resultado mais alto de acurácia é o da arquitetura ResNet50 utilizando apenas a técnica de *transfer learning*. O maior valor de revocação dentre os conjuntos ficou com a arquitetura MobileNet utilizando *transfer learning*, *dropout*, *data augmentation* e *fine-tuning*, chegando a 98,76%. Ainda no âmbito de todos os conjuntos de experimentos, o maior valor de F1-Score foi obtido pela arquitetura ResNet50 utilizando apenas *transfer learning*.

Os valores do tempo apresentados na Tabela 5.1 indicam o tempo que cada arquitetura levou para avaliar todas as 243 imagens pertencentes ao conjunto de teste. Esse tempo é o valor da média de 10 iterações. No primeiro conjunto de experimentos, a arquitetura que obteve o melhor tempo foi a Xception. No segundo e terceiro conjunto, a MobileNet obteve melhores resultados. A Tabela 5.2 apresenta as variâncias das arquiteturas de cada conjunto de experimentos. É possível concluir que o primeiro conjunto de experimentos apresentou os tempos mais dispersos e que no conjunto 2 houve pouca variações.

A Figura 5.4 apresenta o desempenho das métricas da arquitetura VGG16 em cada conjunto de experimentos. É possível concluir que as técnicas de *Dropout*, *Data Augmen-*

*tation* e *Fine-Tuning* não apresentaram uma melhora significativa nas métricas, pois a acurácia se manteve aproximadamente com 74%. Embora a revocação tenha aumentado, a precisão acabou caindo com a adição das técnicas, fazendo com que o valor da métrica F1-Score não obtivesse melhora expressiva.

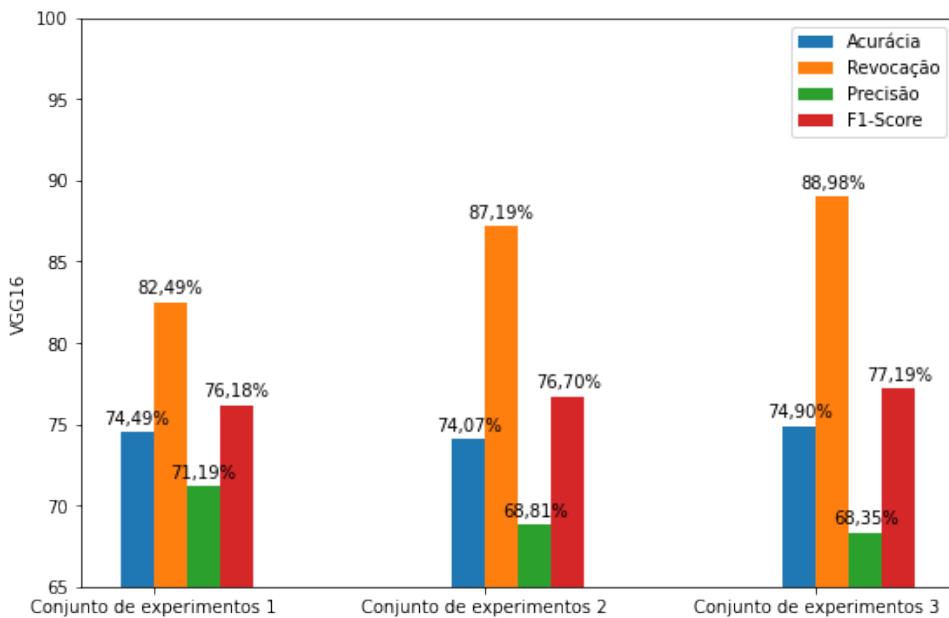


Figura 5.4: Desempenho das métricas da arquitetura VGG16.

A Figura 5.5 compara os resultados das métricas entre os três conjuntos de experimentos para a arquitetura ResNet50. É possível notar que a métrica F1-score apresentou uma melhora com a adição das técnicas em cada conjunto de experimentos.

A Figura 5.6 apresenta a evolução das métricas da arquitetura Xception em cada conjunto de experimentos. É possível notar que tanto a acurácia quanto F1-Score obtiveram aumento nos valores quando foi adicionado as técnicas de *Dropout*, *Data Augmentation* e *Fine-tuning* no terceiro conjunto de experimentos.

A Figura 5.7 mostra a evolução das métricas da arquitetura DenseNet em cada conjunto de experimentos. É possível observar que essa arquitetura, assim como a ResNet50 obteve uma queda na porcentagem da acurácia com a adição das novas técnicas em cada conjunto de experimentos. Em contra partida, as novas técnicas proporcionaram uma melhora de 1.41% da métrica F1-Score.

A partir da Figura 5.7, é possível notar que o uso das técnicas de *transfer learning*, *dropout* e *data augmentation* fizeram com que a rede DenseNet tivesse melhores resultados de revocação e F1-Score em comparação com sua versão do experimento que utiliza apenas *transfer learning*, embora sua acurácia tenha sido menor nesse caso.

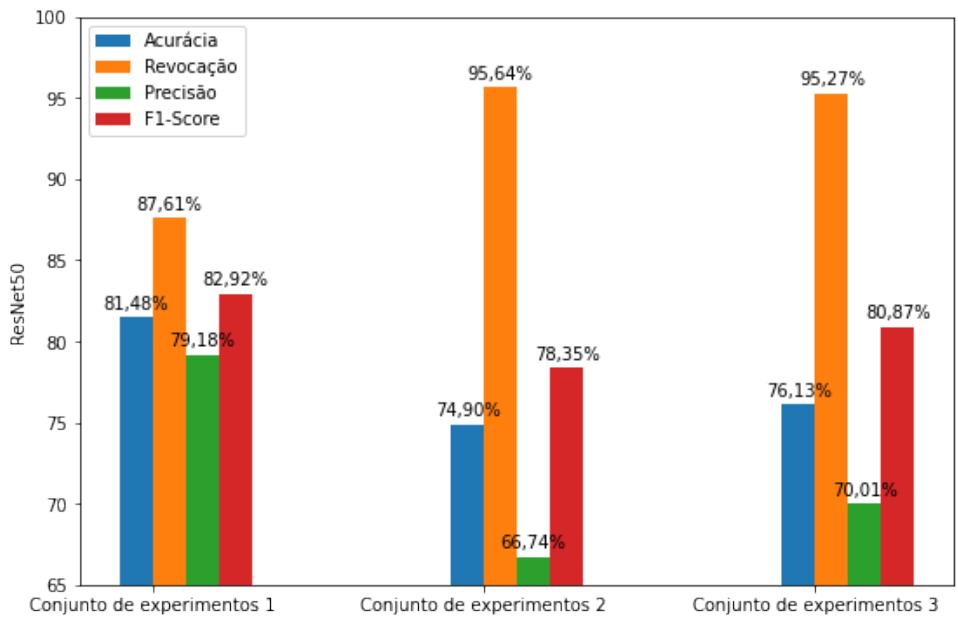


Figura 5.5: Desempenho das métricas da arquitetura ResNet50.

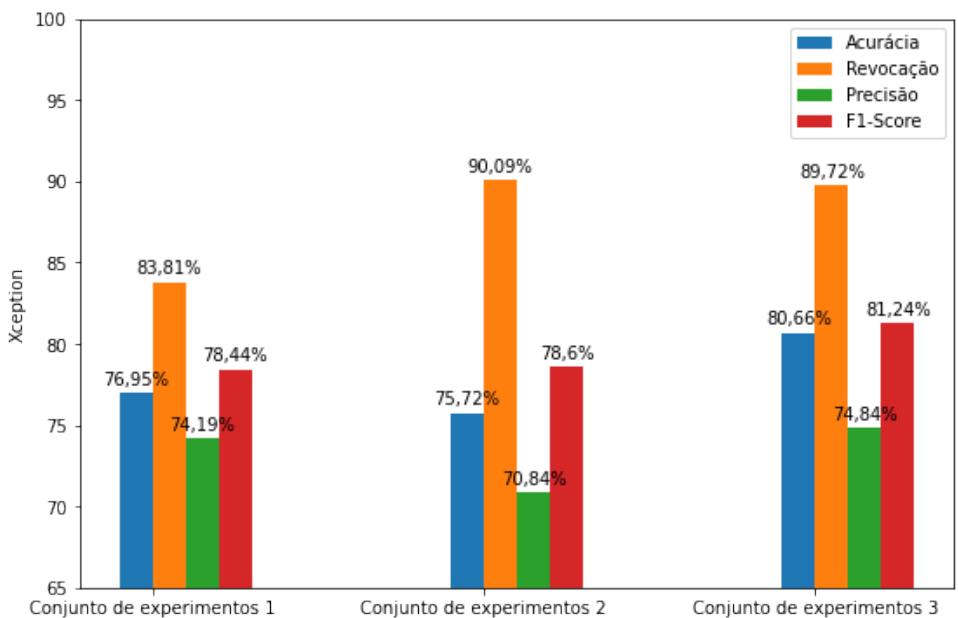


Figura 5.6: Desempenho das métricas da arquitetura Xception.

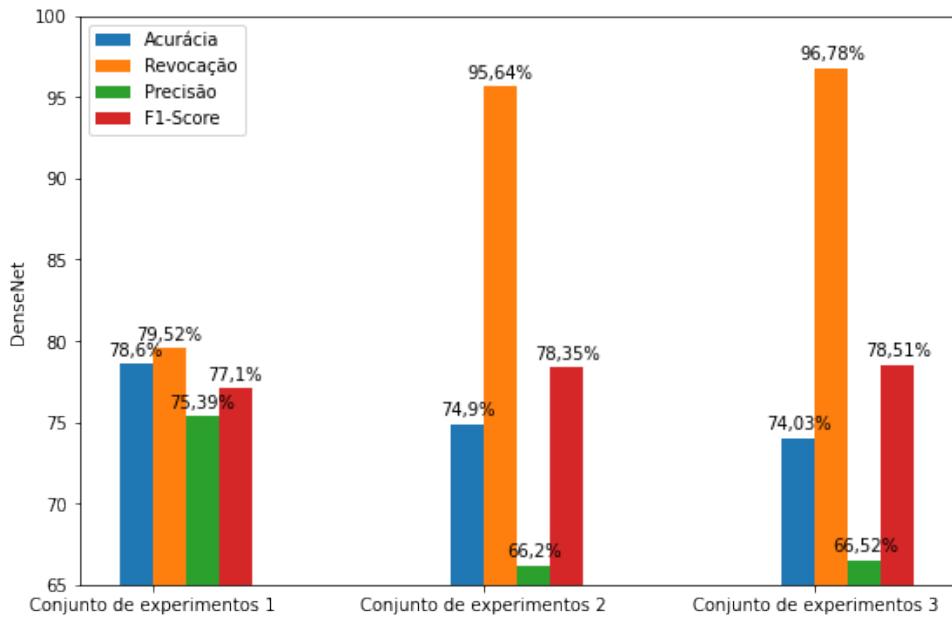


Figura 5.7: Desempenho das métricas da arquitetura DenseNet.

Por fim, a Figura 5.8 mostra a evolução das métricas da arquitetura MobileNet em cada conjunto de experimentos. É possível observar que essa arquitetura obteve um aumento da revocação com a adição das técnicas de *Dropout*, *Data Augmentation* e *Fine-tuning*, todavia, a métrica F1-Score não obteve melhora significativa devido a queda da precisão.

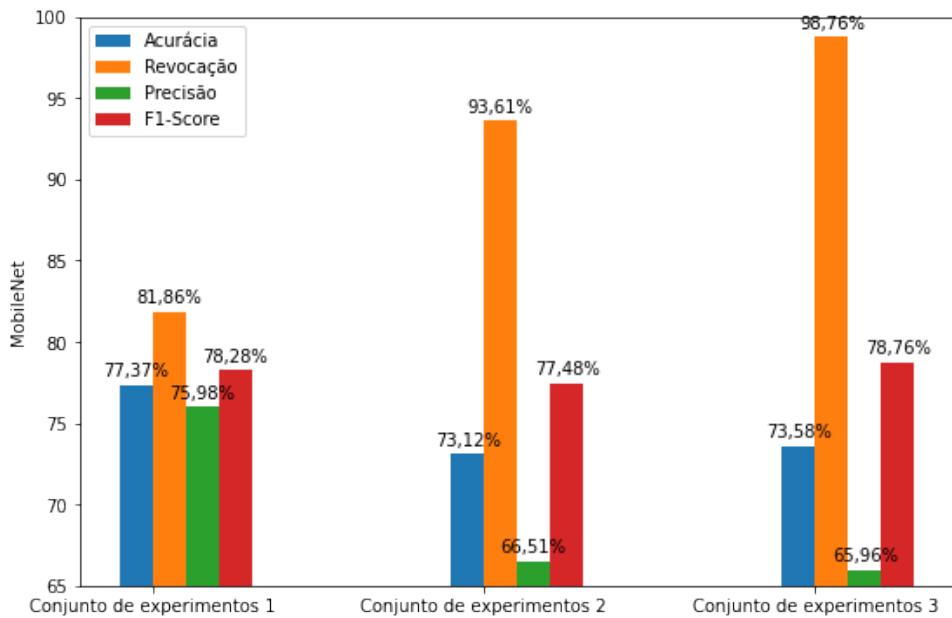


Figura 5.8: Desempenho das métricas da arquitetura MobileNet.

# Capítulo 6

## Conclusão

Neste capítulo serão feitas breves considerações finais na Seção 6.1. A Seção 6.2 aborda algumas das limitações do presente trabalho. Por fim, a Seção 6.3 apresenta algumas possíveis contribuições para trabalhos futuros.

### 6.1 Considerações finais

Utilizando a metodologia proposta no Capítulo 4, o presente trabalho realizou um estudo comparativo do desempenho de diferentes arquiteturas de CNNs na classificação da qualidade de imagens de documentos. Com base nos resultados apresentados na Seção 5.7, o conjunto do experimentos 2 da Seção 5.6 que utiliza as técnicas de *transfer learning*, *data augmentation* e *dropout*, apresentou o melhor resultado segundo as métricas analisadas para a arquitetura Xception com o valor de 78,60% na *F1-Score*. No caso do conjunto de experimentos 3 da Seção 5.6, que utilizou *fine-tuning* em adição as técnicas do conjunto do experimento 2, a arquitetura que apresentou o melhor resultado também foi a Xception com 81,25% de *F1-Score*. Por fim, a arquitetura ResNet50 apresentou os melhores resultados para todas as métricas analisadas no conjunto de experimentos 1, explicado na Seção 5.6, em que é utilizado a técnica de *transfer learning*. Além disso, também apresentou o maior resultado da métrica *F1-Score* dentre todos os conjuntos de experimentos, com o valor de 82,92%.

Apesar da arquitetura ResNet50 ter apresentado o melhor resultado segundo as métricas escolhidas, seu tempo médio de avaliação para as 243 imagens do conjunto de testes foi o pior dentre todos os três conjuntos de experimentos, com o valor de 33,17 segundos nesse caso. Em contraste com a arquitetura que teve o menor tempo médio dentre todos os conjuntos de experimentos, a MobileNet com 26,93 segundos, a diferença entre o maior e o menor tempo pode não ser considerada como não tão relevante dependendo do domínio da aplicação dessa classificação.

## 6.2 Limitações

No presente trabalho, para medir o desempenho dos modelos foi utilizado as imagens do conjunto de teste presentes na base apresentadas na Seção 4.1. Todavia, como a quantidade de imagens é relativamente pequena, os valores das métricas podem ser afetados pela escolha das imagens. Nesses cenários, pode ser utilizada a técnica de *cross-validation* para melhorar a confiabilidade do modelo [17].

## 6.3 Trabalhos Futuros

Embora os resultados obtidos tenham sido satisfatórios para a empresa que concedeu o uso da base, ainda existe espaço para aprimoramentos. Como trabalhos futuros, seria interessante comparar novas arquiteturas.

Um outro ponto que poderia agregar bons resultados a pesquisa é testar o uso de segmentação de imagens antes de iniciar o treinamento do modelo. Conforme apresentado no Capítulo 3, já existem na literatura pesquisas que fazem o uso de segmentação de imagens para detecção de bordas em imagens de documentos [37][10]. Nesse sentido, poderia ser adicionado ao *pipeline* proposto na Figura 4.1 uma camada de pré-processamento para extração de bordas antes de iniciar o treinamento do modelo.

Outra possível contribuição para melhorar ainda mais o *pipeline* proposto na Figura 4.1 é estudar e aplicar também técnicas para a extração do conteúdo do documento ao classificar um documento como bom. No presente trabalho só é feito a classificação da qualidade do documento, todavia, os processos burocráticos dentro da empresa poderiam ser acelerados ao adicionar técnicas de extração de características com o objetivo de capturar as informações do documento.

Por fim, um outro ponto que pode melhorar os resultados dessa pesquisa é a coleta de mais imagens para realização de novos experimentos. A base atualmente conta com 2438 imagens. Todavia, com mais imagens seria possível realizar novos experimentos com o objetivo de validar ainda mais os resultados obtidos aplicando a técnica de *cross-validation*. Isso pode ser feito gerando novos *datasets* para expor o modelo a essas novas variantes, com o objetivo de calcular alguma medida estatística, por exemplo, a média das métricas obtidas nesses diferentes *datasets*.

# Referências

- [1] Borges, Rafael, Kelly Iarosz, Antônio Batista, Iberê Caldas, Fernando Borges e Ewandson Lameu1: *Sincronização de disparos em redes neurais com plasticidade sináptica*. Revista Brasileira de Ensino de Física, 37:2310–1, junho 2015. ix, 6
- [2] Mitchell, Tom M.: *Machine Learning*. McGraw-Hill, New York, 1997, ISBN 978-0-07-042807-2. ix, 6
- [3] Jan, Bilal, Haleem Farman, Muhammad Imran, Ihtesham Islam, Awais Ahmad, Shaukat Ali e Gwanggil Jeon: *Deep learning in big data analytics: A comparative study*. Computers Electrical Engineering, 75, dezembro 2017. ix, 8
- [4] Krizhevsky, Alex, Ilya Sutskever e Geoffrey E Hinton: *Imagenet classification with deep convolutional neural networks*. Communications of the ACM, 60(6):84–90, 2012. ix, 5, 12
- [5] Roman, Victor: *Most popular convolutional neural networks architectures*, 2020. <https://towardsdatascience.com/convolutional-neural-networks-most-common-architectures-6a2b5d22479d>, Último acesso: 10 de fevereiro 2021. ix, 13
- [6] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke e Andrew Rabinovich: *Going deeper with convolutions*. CoRR, abs/1409.4842:1–9, 2014. ix, 13, 14
- [7] Khan, Asifullah, Anabia Sohail, Umme Zahoor e Aqsa Saeed Qureshi: *A survey of the recent architectures of deep convolutional neural networks*. Artificial Intelligence Review, 53(8):5455–5516, 2020. ix, 11, 14
- [8] He, Kaiming, Xiangyu Zhang, Shaoqing Ren e Jian Sun: *Deep residual learning for image recognition*. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. ix, 14, 15, 25
- [9] Mullins, Ryan, Michael Ahearne, Son Lam, Zachary Hall e Jeffrey Boichuk: *Know your customer: How salesperson perceptions of customer relationship quality form and influence account profitability*. Journal of Marketing, 78:22–2429, novembro 2014. 1
- [10] Neves Junior, Ricardo Batista das, Luiz Felipe Verçosa, David Macêdo, Byron Leite Dantas Bezerra e Cleber Zanchettin: *A fast fully octave convolutional neural network for document image segmentation*, 2020. 1, 17, 40

- [11] Poporeshnyak, Svitlana, Olha Suprun, Oleh Suprun e Tadeusz Wieckowski: *Personal documents identification system development using neural network*. Em *13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, páginas 129–134, Lviv, Ukraine, 2018. IEEE. 1, 19, 25
- [12] Sicre, Ronan, Ahmad Montaser Awal e Teddy Furon: *Identity documents classification as an image classification problem*. Em *Identity Documents Classification as an Image Classification Problem*, páginas 602–613. Catania, Italy, 2017. 1, 19, 25
- [13] Li, Zewen, Wenjie Yang, Shouheng Peng e Fan Liu: *A survey of convolutional neural networks: Analysis, applications, and prospects*, 2020. 1, 9, 11
- [14] Labach, Alex, Hojjat Salehinejad e Shahrokh Valaei: *Survey of dropout methods for deep neural networks*, 2019. 1
- [15] Jeddi, Ahmadreza, Mohammad Javad Shafiee e Alexander Wong: *A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning*, 2020. 1
- [16] Monard, Maria Carolina e José Augusto Baranauskas: *Sistemas Inteligentes Fundamentos e Aplicações*, volume 1 de 1. Manole Ltda, Barueri-SP, 1<sup>a</sup> edição, 2003. 4
- [17] Chollet, François: *Deep Learning with Python*. Manning Publications, 1<sup>a</sup> edição, 2017, ISBN 9781617294433. 4, 10, 26, 28, 40
- [18] Tu, Jack Ven: *Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes*. 49(11):1225–1231, 1996. 5
- [19] Haykin, Simon: *Redes Neurais - 2ed.* Bookman, 2<sup>a</sup> edição, 2001, ISBN 9788573077186. 5
- [20] Lawrence, Jeannette: *Introduction to Neural Networks*. California Scientific Software, USA, 1993, ISBN 1883157005. 5
- [21] Gardner, Matthew William e S. R. Dorling: *Artificial neural networks (the multi-layer perceptron)—a review of applications in the atmospheric sciences*. Atmospheric environment, 32(14-15):2627–2636, 1998. 7
- [22] Rosebrock, Adrian: *Deep Learning for Computer Vision with Python*. PyImage-Search, 2017. 8, 9, 10, 31
- [23] Svozil, Daniel, Vladimir Kvasnicka e Jiri Pospichal: *Introduction to multi-layer feed-forward neural networks*. 39(1):43–62, 1997. 8
- [24] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep Learning (Adaptive Computation and Machine Learning series)*. ISBN 978-0262035613. 9, 11
- [25] Datta, Leonid: *A survey on activation functions and their relation with xavier and he normal initialization*, 2020. 10

- [26] LeCun, Yann, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard e Lawrence Jackel: *Backpropagation applied to handwritten zip code recognition*. Neural computation, 1(4):541–551, 1989. 11
- [27] LeCun, Yann, Léon Bottou, Yoshua Bengio e Patrick Haffner: *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, 1998. 12
- [28] Hochreiter, Sepp: *The vanishing gradient problem during learning recurrent neural nets and problem solutions*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02):107–116, 1998. 12
- [29] Simonyan, Karen e Andrew Zisserman: *Very deep convolutional networks for large-scale image recognition*. arXiv preprint arXiv:1409.1556, setembro 2014. 12, 25
- [30] Nwankpa, Chigozie, Winifred Ijomah, Anthony Gachagan e Stephen Marshall: *Activation functions: Comparison of trends in practice and research for deep learning*, 2018. 13
- [31] Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong e Qing He: *A comprehensive survey on transfer learning*. CoRR, abs/1911.02685, 2019. <http://arxiv.org/abs/1911.02685>. 14
- [32] Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang e Chunfang Liu: *A survey on deep transfer learning*. CoRR, abs/1808.01974, 2018. <http://arxiv.org/abs/1808.01974>. 15
- [33] Fawcett, Tom: *Roc graphs: Notes and practical considerations for researchers*. Machine Learning, 31:1–38, janeiro 2004. 15
- [34] Vakili, Meysam, Mohammad Ghamsari e Masoumeh Rezaei: *Performance analysis and comparison of machine and deep learning algorithms for iot data classification*, 2020. 16
- [35] Powers, David M. W.: *What the f-measure doesn't measure: Features, flaws, fallacies and fixes*, 2019. 16
- [36] Tavakolian, Niloofar, Azadeh Nazemi e Donal Fitzpatrick: *Real-time information retrieval from identity cards*, 2020. 18
- [37] Mothes, Oliver e Joachim Denzler: *Self-supervised data bootstrapping for deep optical character recognition of identity documents*. arXiv preprint arXiv:1908.04027, 2019. 18, 40
- [38] Simon, Marcel, Erik Rodner e Joachim Denzler: *Fine-grained classification of identity document types with only one example*. Em 14th IAPR International Conference on Machine Vision Applications (MVA), páginas 126–129, Tokyo, Japan, 2015. IEEE. 19

- [39] Awal, Ahmad Montaser, Nabil Ghanmi, Ronan Sicre e Teddy Furon: *Complex document classification and localization application on identity document images*. Em *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, páginas 426–431, Kyoto, Japan, 2017. IEEE. 20, 25
- [40] Chollet, François: *Xception: Deep learning with depthwise separable convolutions*. CoRR, abs/1610.02357, 2016. <http://arxiv.org/abs/1610.02357>. 25
- [41] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens e Zbigniew Wojna: *Rethinking the inception architecture for computer vision*. CoRR, abs/1512.00567, 2015. <http://arxiv.org/abs/1512.00567>. 25
- [42] Sandler, Mark, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov e Liang-Chieh Chen: *Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation*. CoRR, abs/1801.04381, 2018. <http://arxiv.org/abs/1801.04381>. 25
- [43] Huang, Gao, Zhuang Liu e Kilian Q. Weinberger: *Densely connected convolutional networks*. CoRR, abs/1608.06993, 2016. <http://arxiv.org/abs/1608.06993>. 25
- [44] Deng, Jia, Wei Dong, Richard Socher, Li Jia Li, Kai Li e Li Fei-Fei: *Imagenet: A large-scale hierarchical image database*. Em *2009 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 248–255, 2009. 26
- [45] Perez, Luis e Jason Wang: *The effectiveness of data augmentation in image classification using deep learning*. CoRR, abs/1712.04621, 2017. <http://arxiv.org/abs/1712.04621>. 27
- [46] Mikołajczyk, Agnieszka e Michał Grochowski: *Data augmentation for improving deep learning in image classification problem*. Em *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, páginas 117–122, 2018. 27
- [47] Gu, Shanqing, Manisha Pednekar e Robert Slater: *Improve image classification using data augmentation and neural networks*. SMU Data Science Review, 2(2):1, 2019. 27
- [48] Hinton, Geoffrey E., Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever e Ruslan Salakhutdinov: *Improving neural networks by preventing co-adaptation of feature detectors*. CoRR, abs/1207.0580, 2012. <http://arxiv.org/abs/1207.0580>. 28
- [49] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever e Ruslan Salakhutdinov: *Dropout: a simple way to prevent neural networks from overfitting*. 15(1):1929–1958, 2014. 28
- [50] Kingma, Diederik P. e Jimmy Ba: *Adam: A method for stochastic optimization*, 2017.