UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

FRANCISCO IGOR DA SILVA LIMA

ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA UM PROBLEMA DE CLASSIFICAÇÃO

QUIXADÁ

2018

FRANCISCO IGOR DA SILVA LIMA

ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA UM PROBLEMA DE CLASSIFICAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Profª. Drª. Ticiana Linhares Coelho da Silva

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L698e Lima, Francisco Igor da Silva.
ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA UM PROBLEMA DE CLASSIFICAÇÃO / Francisco Igor da Silva Lima. – 2018.
46 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2018.
Orientação: Profa. Dra. Ticiana Linhares Coelho da Silva.

1. Aprendizagem Profunda. 2. Redes Neurais (Computação). 3. Visão Computacional. I. Título.
CDD 005.1

FRANCISCO IGOR DA SILVA LIMA

ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA UM PROBLEMA DE CLASSIFICAÇÃO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: __/__/__

BANCA EXAMINADORA

Profª. Drª. Ticiana Linhares Coelho da Silva (Orientadora)
Universidade Federal do Ceará (UFC)

Profº Dr. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

Profº Me. Victor Aguiar Evangelista de Farias
Universidade Federal do Ceará (UFC)

A Deus,

Aos Meus pais, Inacilda e Marcílio.

AGRADECIMENTOS

Agradeço a Deus por seu Amor e Misericórdia, e por ter me dado sabedoria e força suficiente para concluir minha graduação.

Aos meus pais, Inacilda e Marcílio, que sempre estiveram comigo, acordaram cedo quando eu precisava sair cedo, dormiram tarde quando precisei chegar tarde, nada que eu diga e faça pode retribuir o que fizeram por e para mim.

A minha irmã Iara, que pode ir muito além de onde eu cheguei. Iara, você tem um futuro brilhante pela frente!

A minha vó, Dona Nega que, agora do céu, sempre me incentivou a estudar. Obrigado vó, a senhora sempre morará no meu coração.

Aos meus avós, Inácia e Braz, que mesmo distantes estão presentes em meu coração e minhas orações.

As minhas tias e tios, em especial a Dulce, Lucivanda, Derci, Ene, por me incentivarem a estudar.

Aos meus amigos que fiz durante a graduação, em especial, Caio, Cezar, Erika, Isaias, Letícia, Júlio e Rodrigo por tudo que vivemos nesses anos de UFC - Quixadá, e viveremos, sou grato a Deus pela nossa amizade.

Aos amigos que fiz no PET-TI, em especial, Rayanne, Lucas, Tiago, Micaele, Raul, Robberty e Nayara.

A professora Ticiana, por sua orientação e paciência comigo durante esse trabalho, obrigado por tudo Ticiana!

A professora Diana, uma segunda mãe para os alunos de Engenharia de Software, por sua simpatia e alegria. Diana, você não sabe o bem que faz aos seus alunos, obrigado!

A professora Lívia, por sua orientação em um trabalho anterior. Que Deus abençoe você e sua família eternamente, Lívia!

Ao professor Regis, por suas valiosas contribuições nesse trabalho e seu ótimo senso de humor.

Ao professor Victor pelas suas valiosas contribuições nesse trabalho.

Aos demais professores do campus, em especial, Diego, Paulyne, Rainara, João, pelo seu esforço constante de tornar seus alunos não somente bons profissionais, mas boas pessoas.

A todos que compõe a UFC Quixadá, por tornarem o campus um dos melhores ambientes de aprendizagem que poderia ter. Agradeço a todos os funcionários do campus, em

especial a Ana Cláudia, Aline, Dias, Natália, Simone, e o Venício.

Aos professores das escolas Escola Abel Ferreira e do Liceu de Banabuiú, sem vocês eu não seria capaz de chegar onde cheguei, e sei que sempre lembrarei de vocês.

A todos que contribuíram positivamente para a minha formação.

Quem a Deus tem, Nada lhe falta:

Só Deus basta.

(Santa Tereza D'Ávila)

RESUMO

O uso de redes neurais convolucionais para classificação de imagens tem crescido bastante nos últimos anos. Desde 2012 são notáveis os grandes avanços na área de visão computacional. A partir desse período, redes neurais convolucionais ganharam bastante atenção da academia e do mercado. Devido, principalmente, aos seus bons resultados, causados por uma maior disponibilidade de dados, novos algoritmos e maior poder computacional, este último, especialmente, impactados pelos aos avanços da computação em nuvem. O presente trabalho visa realizar um estudo comparativo entre as redes neurais: VGG16, InceptionV3, ResNet50 a fim de descobrir qual a melhor configuração entre estas redes para a solução de um problema de classificação de imagens. Como resultados é possível afirmar a VGG16 não é uma boa rede para esse problema, enquanto a InceptionV3 e a ResNet50 apresentam resultados melhores.

Palavras-chave: Aprendizagem Profunda. Redes Neurais. Visão Computacional.

ABSTRACT

The use of convolutional neural networks for image classification has grown up in the last years. Since 2012 there are notable great advances in the computational vision field. From this period, convolutional neural networks have gained the attention of the academy and the market. Due, principally, to its good results, new algorithms, and greater computational power, the last one, especially caused by the advantages of cloud computing. This work aims to make a comparative study among the neural networks: VGG16, InceptionV3, ResNet50 to discover what is the best configuration among these networks to a solution of an image classification problem. As result it is possible to affirm that VGG16 is not a good network for this problem, while InceptionV3 and ResNet50 present better results.

Keywords: Deep Learning. Neural Network. Computational Vision.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Para seres humanos reconhecerem padrões, objetos e faces pode ser algo simples, porém, para computadores, esta mesma atividade não é tão trivial. Pelo contrário, é algo complexo e caro (computacionalmente) na maioria das aplicações. Portanto, o reconhecimento de objetos por imagens tem sido uma das áreas da computação que mais tem se destacado nos últimos anos, principalmente em aplicações como reconhecimento de faces presente em aplicativos bastante utilizados, como *Facebook* e *Instagram*. Vários trabalhos têm sido propostos neste contexto e utilizam Redes Neurais Convolucionais (CNN) como solução (KUMAR *et al.*, 2009; GUILLAUMIN *et al.*, 2009; KRIZHEVSKY *et al.*, 2012).

Uma Rede Neural Convolucional é um tipo de Rede Neural (GOODFELLOW *et al.*, 2016). Redes Neurais são compostas de camadas de neurônios, onde cada neurônio recebe dados de entrada, realizam computações, e retornam para cada objeto de entrada uma distribuição de probabilidade. Esta distribuição tem a probabilidade do objeto pertencer a cada uma das classes que formam a saída da rede, ou pode retornar unicamente a classe com maior probabilidade (FEI-FEI *et al.*, 2018).

Em diversos problemas, Redes Neurais Convolucionais têm se apresentado como solução promissora para classificação de imagens, problemas estes que são abordados nos trabalhos de (KRIZHEVSKY *et al.*, 2012; KARPATHY *et al.*, 2014). Com essa motivação, decidiu-se realizar uma análise comparativa entre algumas das redes neurais convolucionais propostas em edições da competição do ILSVRC (*ImageNet Large Scale Visual Recognition Competition*). O ILSVRC é uma competição para detecção de objetos em imagens/vídeos em larga escala. Os dados do ILSVRC são oriundos do ImageNet, que é um banco de dados de imagens, contendo aproximadamente 14,2 milhões de imagens, organizado em aproximadamente 21 mil classes, cada classe contendo aproximadamente mil imagens. Vale ressaltar que uma imagem pode pertencer a mais de uma classe. As redes convolucionais que foram escolhidas neste trabalho são as listadas abaixo, pois foram umas das que obtiveram os melhores resultados no ILSVRC nos últimos anos:

1. VGG16
2. Inception
3. ResNet50

Os dados que foram utilizados neste trabalho são oriundos da competição do *Kaggle*[1] sobre detecção de raças de cachorros. O *Kaggle* é uma plataformas de Ciência de Dados, fundado em 2010 com o objetivo de ser um espaço para competições de ciência de dados e aprendizado de máquina. Nessa plataforma, diversas empresas e pesquisadores hospedam desafios e dados para que pessoas de vários locais do mundo possam participar das competições. Dentre essas competições, foi escolhida a *Dog Breed Identification*[2], que consiste em retornar para cada cachorro a probabilidade do mesmo pertencer a uma das raças de cachorro listadas na competição.

Neste trabalho, as Redes Neurais Convolucionais foram comparadas usando as métricas: acurácia e *log loss*. Bem como, o resultado da classificação obtido de cada rede, será também submetido e ranqueado na plataforma *Kaggle*.

O objetivo deste trabalho é verificar qual a melhor CNN para o problema *Dog Breed Identification* e seus hiper parâmetros. O público alvo são pesquisadores, profissionais e entusiastas na área de Aprendizado de Máquina, Aprendizado Profundo, e Inteligência Artificial.

O trabalho está organizado da seguinte forma: no Capítulo 2, são apresentados os objetivos gerais e específicos. Na Capítulo 3, são apresentados os termos e conceitos necessários para entender o trabalho. Na Capítulo 4, são apresentados três trabalhos relacionados com o presente. O Capítulo 5 contém os procedimentos metodológicos que foram realizados durante a execução do trabalho. O Capítulo 6 apresenta os resultados. Por fim, o Capítulo 7 apresenta as conclusões finais do trabalho.

[1] https://www.kaggle.com/

[2] https://www.kaggle.com/c/dog-breed-identification

2 OBJETIVOS

A seguir, serão apresentados os objetivos deste trabalho.

2.1 Objetivo Geral

O objetivo desse trabalho é descobrir a melhor Rede Neural Convolucional dentre as que foram escolhidas: VGG16, InceptionV3 e ResNet50, a fim de analisar e escolher seus melhores hiper-parâmetros para solucionar a competição *Dog Breed Identification* do *kaggle*.

2.2 Objetivos específicos

a) Treinar as redes VGG16, InceptionV3 e ResNet50;
b) Avaliar as redes através das métricas: *log loss* e acurácia;

3 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, são apresentados os conceitos chaves que serão utilizados ao longo de todo o trabalho.

3.1 Aprendizado Supervisionado

Aprendizado Supervisionado segundo (JAMES *et al.*, 2013) ocorre quando existe um conjunto de treino, de tal forma que para cada elemento no conjunto existe um vetor de características, e um componente para cada elemento da conjunto de treino, sendo que estes tipos de algoritmos buscam aprender uma função que mapeia as entradas com as saídas desejadas. Em Redes Neurais, o conjunto de treinamento é utilizado para definir e ajustar continuamente os pesos da rede, de modo a minimizar os erros do modelo criado.

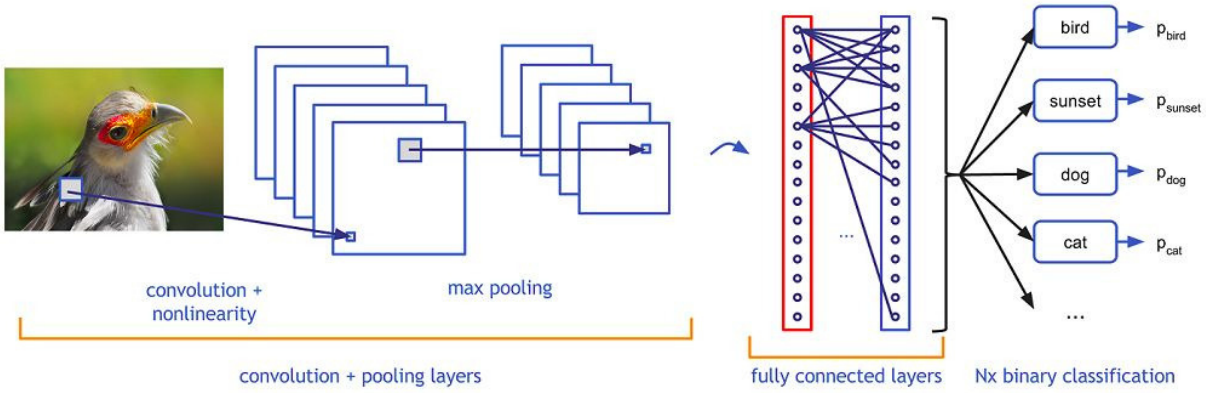
3.2 Redes Neurais

O termo Redes Neurais (ou Aprendizado Profundo) vem da neurobiologia, mas segundo (CHOLLET, 2017) apesar de alguns conceitos virem da neurobiologia, redes neurais não são modelos do cérebro, mas sim um *framework* matemático para aprender representações dos dados. Elas são capazes de aprender realizar tarefas de reconhecimento de imagens, vídeos, fala, e até mesmo processamento de linguagem natural (NLP) (BEZERRA, 2016).

Dentre as redes profundas, se destacam as Redes Neurais Convolucionais (*convolutional neural networks,* CNN ou ainda, *ConvNet*) que segundo (LECUN *et al.*, 1998) e (ZEILER; FERGUS, 2014) se inspiram no funcionamento do córtex visual. Em geral, uma CNN apresentam diversas camadas: convolucionais, subamostragem, normalização, completamente conectadas, entre outras. Nas CNN's que serão utilizados neste trabalho, todas essas camadas estão presentes.

A Figura 1 mostra um exemplo de CNN onde existem as camadas convolucionais, completamente conectada e a camada de saída.

Figura 1 – Exemplo de uma Rede Neural Convolucional



Fonte: Adaptado de (DESHPANDE, 2016)

3.2.1 Camada Convolucional

A camada convolucional é responsável por aplicar filtros para extrair características da entrada. No caso das imagens, as características podem ser arestas, retas, texturas, curvas, entre outros (BEZERRA, 2016). Esses filtros são matrizes de pesos aplicados à saída da camada anterior. A camada anterior pode representar a entrada, por exemplo.

3.2.2 Camada de Subamostragem

As camadas de subamostragem envolvem a redução da resolução da imagens, mas não realizam alterações significativas no aspecto da imagem, o objetivo dessa camada é para diminuir o tamanho da imagem, diminuindo gastos computacionais. Em uma CNN, a camada de subamostragem reduz a dimensionalidade de um mapa de características gerando um novo mapa, uma espécie de resumo. Existem algumas formas de realizar a subamostragem, tais como, selecionar o valor máximo (*max pooling*), ou o valor médio (*average pooling*), ou a norma do conjunto (*L2-pooling*) de pixels vizinhos. Com o uso de subamostragem a *CNN* se torna mais robusta em relação a localização exata das características, propriedade esta que faz com que a rede seja capaz de aprender representações invariantes à pequenas diferenças entre as imagens Bezerra (2016).

Considere um conjunto de características como o da Figura 2 de tamanho 4x4, onde será aplicado um filtro 2x2 com passo igual a 2 (passo significa a quantidade de píxels por vez na qual o filtro se movimenta na imagem). No caso, o filtro é inicialmente aplicado na parte superior esquerda pegando os elementos 1, 1, 5 e 6, em seguida na superior direita, inferior esquerda e finalmente na inferior direita. A aplicação do filtro produz ainda uma matriz 4X4. Imagine que

neste caso, em seguida, seja aplicado como camada de subamostragem o *max pooling*. Dessa forma, o valor selecionado é igual a 6 para o canto superior esquerdo, como o passo é igual a 2, a próxima a área para aplicação do *max pooling* é para os elementos: 2, 4, 7, 8, em que o valor selecionado é 8. Logo após, é escolhido o valor máximo do canto inferior esquerdo e em seguida, do canto inferior direito. A direita da Figura 2 é mostrado o resultado.

Figura 2 – Subamostragem com filtro de tamanho 2x2 e tamanho do passo igual a 2.

Fonte: Adaptado de (DESHPANDE, 2016)

3.2.3 Camada de Normalização

Esta camada, quando utilizada, é posicionada logo após a camada de subamostragem, logo, tendo como entrada a saída da camada de subamostragem. A normalização não é aplicada a imagem como um todo, ela é aplicada sobre partes da imagem, de pixel em pixel. Um exemplo de normalização é a invariância com brilho, que consiste em subtrair a média do valor da vizinhança de um determinado pixel, e dividir pela variância dos valores de pixel da imagem (BEZERRA, 2016).

3.2.4 Camada Completamente Conectada

Em CNNs modernas, é comum encontrar uma ou duas camadas completamente conectada (*Fully Connected*, ou apenas FC). Camadas deste tipo geram descritores de características, que podem ser facilmente classificados pela camada de saída (BEZERRA, 2016).

3.2.5 Camada de Saída

A camada de saída é responsável por gerar uma distribuição de probabilidades de todas as classes para um determinado objeto. Nas redes modernas, geralmente, usa-se a função *Softmax*, que permite interpretar os valores da camada de saída como probabilidades (BEZERRA, 2016). A fórmula da função *Softmax* é a seguinte:

$$o(a_i^{(L+1)}) = \frac{e^{a_i^{(L+1)}}}{\sum_{j=1}^{C} e^{a_j^{(L+1)}}} \tag{3.1}$$

Onde, C é a quantidade de classes existente no *dataset*, $a_i^{(L+1)}$, sendo $1 \leq i \leq C$ são os valores de probabilidade, tal que estão entre os valores de 0 e 1, e sua soma é igual a 1.

3.2.6 Regularização

Uma das técnicas utilizadas para redução de *overfitting* é regularização. A ideia por trás da regularização é tornar o modelo mais simples, pois segundo (CHOLLET, 2017) modelos simples são menos propensos a *overfitting* que modelos complexos. Nesse contexto, um modelo mais simples é um modelo com menos parâmetros. Sendo assim uma forma de reduzir esse problema é adicionando restrições (ou custos) na rede, forçando seus pesos a obter apenas valores pequenos. Existem três tipos principais de regularição: a L1, ou Lasso (*Least Absolute Shrinkage and Selection Operator*), e a L2, ou *Ridge*, e *Dropout*.

3.2.6.1 Regularização L1

Na Regularização L1, o custo é proporcional ao valor absoluto dos pesos. Ela é implementada adicionando o erro definido na Equação 3.2 aos pesos, a fim de ajudar a rede a detectar e ignorar características desnecessárias.

$$E_1 = \alpha \sum_{w} |w| \tag{3.2}$$

Onde, w representa os pesos da rede, e α é o custo adicionado aos pesos.

3.2.6.2 Regularização L2

Na Regularização L2, o custo é proporcional ao valor quadrado dos pesos da rede. Ela é implementada adicionando o erro definido na Equação 3.3, Regularização L2 também é

conhecida como decaimento de peso.

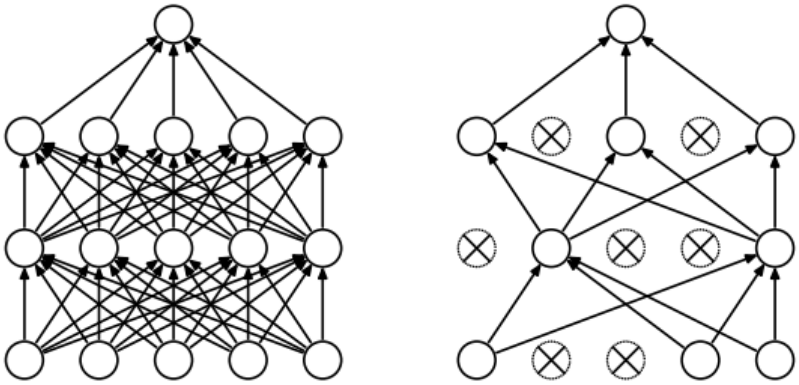
$$E_1 = \alpha \sum_w |w^2| \tag{3.3}$$

Onde, w representa os pesos da rede, e α é o custo adicionado aos pesos.

3.2.6.3 Dropout

Dropout é uma das técnicas de regularização mais eficientes e usadas (CHOLLET, 2017). A ideia principal é desligar aleatoriamente neurônios durante o treinamento. Isto previne que o modelo se adapte bastante aos dados de treinamento. Neste caso, desligar aleatoriamente significa remover o neurônio da rede, e suas conexões de entrada e saída (SRIVASTAVA *et al.*, 2014). A probabilidade de um neurônio ser desligado é dado por p, o valor de p varia de 0 até 1, mas geralmente é escolhido um valor entre 0.2 e 0.5 (CHOLLET, 2017). Uma rede utilizando um *Dropout* de 0.5, pode ser vista na Figura 3.

Figura 3 – Rede Neural com *Dropout*. Esquerda: Uma rede padrão com duas camadas ocultas. Direita: Um exemplo de uma rede neural produzida após aplicar *dropout*. Os neurônios riscados foram removidos da rede



Fonte: Adaptado de (SRIVASTAVA *et al.*, 2014)

3.2.7 Funções de perda

As funções de perda (do inglês *loss functions*), ou funções objetivas, representam o que será minimizado durante o treinamento da rede neural. A escolha da função de custo é extremamente importante, já que a rede irá tentar de toda forma minimizar o custo dessa função (CHOLLET, 2017). Existem diversas funções de perda:

- Entropia cruzada binária, para problemas de classificação binária
- Entropia cruzada categórica, para problemas de classificação de multi classe.

O problema foco deste trabalho é um problema de classificação multiclasse, logo será utilizada a função de perda Entropia cruzada categórica, *Log Loss*.

3.2.8 Retropropagação de erros

Retropropagação de erros (*backward propagation of errors* ou *backpropagation*) é um procedimento que durante o treinamento da rede ajusta os pesos das conexões da rede neural a fim de minimizar a diferença entre a saída da rede (predição) e o valor desejado. Como resultado do ajuste dos pesos, camadas ocultas conseguem identificar importantes características da entrada (RUMELHART *et al.*, 1986).

Para tal, é necessário calcular as derivadas parciais das funções de perda para aprender os melhores pesos das camadas ocultas (RASCHKA; MIRJALILI, 2017). Existem alguns otimizadores que são capazes de realizar tal tarefa, dentre eles, um dos mais utilizados é o Gradiente Descendente (GD), ou uma de suas variações, o Gradiente Descendente Estocástico (SGD) ou o Gradiente Descendente em Lote. Existe ainda o RMSProp (*Root Mean Square Propagation*), que foi definido por (TIELEMAN; HINTON, 2012).

3.2.9 Data Augmentation

Segundo (PEREZ; WANG, 2017) quanto mais dados são fornecidos aos algoritmos de aprendizado de máquina, melhores eles são treinados. Uma das técnicas para aumentar a quantidade de dados é *data augmentation*. Para fazer uso dela é necessário realizar transformações nas imagens. Para cada imagem pode ser gerado uma série de transformações, que por sua vez podem ser: *zoom*, rotação, giro, distorção, entre outros (PEREZ; WANG, 2017). Entretanto, as imagens geradas devem possuir as mesmas classes geradas (WONG *et al.*, 2016). Na Figura 4 é possível ver um exemplo de *data augmentation*.

Figura 4 – Exemplo de *Data augmentation* com imagem de gatos

Fonte: (CHOLLET, 2016)

3.3 Unidades Lineares Retificadas (ReLU)

Durante muito tempo existiu um consenso sobre o uso de funções sigmoides, que são funções de ativação, para implementarem a ativação das camadas ocultas. As camadas ocultas são todas as camadas que apenas a CNN sabe de sua existência, ou seja, praticamente todas as camadas, com exceção da camada de entrada e a de saída, que são comuns a todas redes neurais.

Entretanto, o trabalho (NAIR; HINTON, 2010) questionou esse consenso e propôs uma alternativa mais simples, a função de ativação retificada linear, a ReLU (*Rectified Linear Unit*). Nos experimentos de (KRIZHEVSKY *et al.*, 2012), é demostrado que a ReLU gera um erro de treinamento menor, já que não é tão suscetível ao problema da dissipação dos gradientes que ocorrem nas funções sigmoides. Como a ReLU não envolve exponenciações, ela reduz o tempo de convergência dos parâmetros. A fórmula da ReLU é a seguinte:

$$f(x) = max(x, 0) \tag{3.4}$$

3.4 VGG16

A VGG16 é uma Rede Neural Convolucional proposta por (SIMONYAN; ZISSERMAN, 2014). O termo "VGG" vem do nome do grupo que a desenvolveu, o *Visual Geometry Group*[1], já o número 16, por conta da profundidade da rede. A profundidade de uma CNN se dá pela quantidade de camadas que estão sobrepostas, a entrada da VGG16 consiste, em geral, de imagens de tamanho de 224x224. Na VGG16 o tamanho do filtro é de 3x3, que no caso é o menor filtro que permite ter a noção de esquerda/direita, e de acima/abaixo. O passo nela usado é de tamanho 1.

A VGG16 possui 13 camadas convolucionais e 3 totalmente conectadas. As camadas estão agrupadas em 5 blocos convolucionais, os dois primeiros blocos contêm apenas duas camadas convolucionais, enquanto os demais apresentam 3 camadas. Entre os blocos convolucionais existe uma camada de subamostragem. Nesta abordagem a largura das camadas convolucionais dobra após cada camada de subamostragem até chegar no valor de 512. A arquitetura da VGG16 pode ser observada na Tabela 1.

[1] http://www.robots.ox.ac.uk/ vgg/

Tabela 1 – Arquitetura da VGG16

Entrada(imagens de 224 x 224 em RGB)
Convolução-64 Convolução-64
Subamostragem
Convolução-128 Convolução-128
Subamostragem
Convolução-256 Convolução-256 Convolução-256
Subamostragem
Convolução-512 Convolução-512 Convolução-512
Subamostragem
Convolução-512 Convolução-512 Convolução-512
Subamostragem
Totalmente Conectada 4096
Totalmente Conectada 4096
Totalmente Conectada 1000
soft-max

Fonte: Adaptado de (SIMONYAN; ZISSERMAN, 2014)

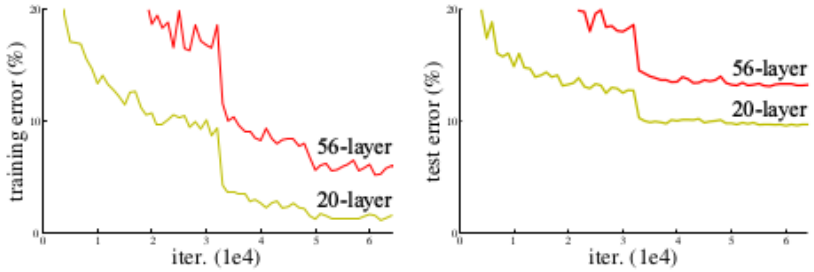
3.5 ResNet50

Residual Learning é uma técnica proposta no trabalho de (HE *et al.*, 2016). Os autores do mesmo reconhecem a importância da profundidade da rede, mas fazem o seguinte questionamento: "apenas aumentar a quantidade de camadas é suficiente para melhorar a acurácia do modelo?". O problema encontrado é que a medida que a profundidade aumenta, a acurácia pode saturar, como observado na Figura 5. Entretanto, isto não é causado por *overfitting*, no caso a adição de novas camadas aumenta o erro de treino. Este problema é conhecido como degradação de gradiente.

As Redes Neurais Convolucionais testadas em (HE *et al.*, 2016) e apresentadas na Figura 5 apresentaram 20 camadas e 56 camadas. Essas Redes Neurais Convolucionais não utilizam *residual learning*. O que se concluiu deste estudo é que quanto mais profunda é a rede, maior pode ser o erro de treinamento e teste.

Para resolver esse problema, o trabalho (HE *et al.*, 2016) levou em consideração a existência de uma Rede Neural rasa, ou seja, com baixa profundidade, e ainda uma segunda CNN

Figura 5 – A esquerda o erro de treino, e na direita o erro de teste na base CIFAR-10.

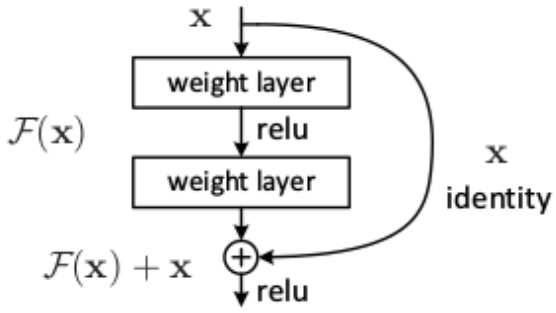


Fonte: (KRIZHEVSKY; HINTON, 2009)

mais profunda (com mais camadas). Para que o erro de treinamento não aumentasse, as camadas que são adicionadas no modelo mais profundo são camadas de mapeamento de identidade, isso indica que o erro de treinamento no modelo mais profundo não deve ser maior do que o erro do modelo mais raso. Entretanto, os experimentos feitos pelos autores encontraram resultados piores nas redes profundas que nas redes mais rasas.

Para isto, foi criado um *framework*, chamado de *residual learning*. Nele, ao invés de esperar que cada camada seja treinada diretamente com a saída da camada anterior, a camada é treinada com um entrada residual. A entrada que deveria ser $F(X)$ é recalculada para $F(X)+X$, sendo que $F(X)$ e X devem ter a mesma forma, como pode ser observado na Figura 6. Com isto, o modelo criado não deve produzir um erro de treinamento maior do que o modelo mais raso.

Figura 6 – Um exemplo de como o *Residual Learning* é adicionado na arquitetura da CNN



Fonte: Adaptado de (HE *et al.*, 2016)

3.6 InceptionV3

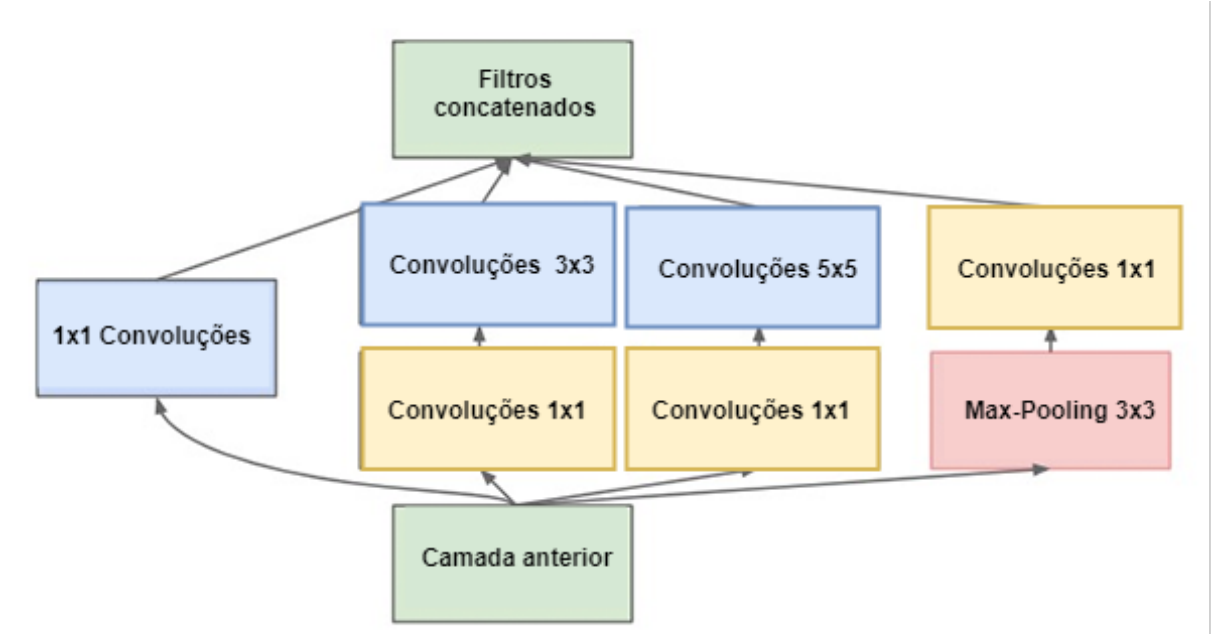
No trabalho de (SZEGEDY *et al.*, 2015), os autores afirmam que no momento que duas camadas são empilhadas, a quantidade de computações que serão realizadas tem um crescimento quadrático. E se caso essas operações forem usadas de maneira ineficiente (por

exemplo, boa parte dos pesos forem próximos de 0), toda essa computação pode ser considerada como perdida. Como recursos computacionais, tais quais GPU's, CPU's e memória, são finitos, é preferível usar estes recursos de forma otimizada do que aumentar indiscriminadamente a profundidade das redes, mesmo que isso acarrete em uma melhora dos resultados.

Uma possível solução para isso seria a mudança de arquiteturas totalmente conectadas para arquiteturas esparsas (que são arquiteturas com menos conexões entre camadas), mesmo dentro de convoluções. Entretanto, mesmo que as operações sejam reduzidas em 100x, a sobrecarga de pesquisas em disco e falhas de *cache* é tão grande que não compensa a mudança para o uso de arquiteturas esparsas. Esta lacuna é ampliada ainda mais pelo o uso de bibliotecas matemáticas extremamente aprimoradas e altamente ajustadas que exploram os mínimos detalhes das CPU's e GPU's. Também vale ressaltar que modelos esparsos não uniformes necessitam de uma infraestrutura de engenharia e computacional mais sofisticada (SZEGEDY *et al.*, 2015).

A ideia principal da arquitetura da InceptionV3 é baseada em descobrir como uma estrutura esparsa local ideal em uma CNN, pode ser aproximada e coberta por camadas densas prontamente disponíveis(SZEGEDY *et al.*, 2015). Sendo assim, o Inception é um módulo que faz convoluções em paralelo com os filtros 1X1, 3X3, 5X5, o que possibilita a descoberta de diversas características em uma mesma camada, como pode ser observado na Figura 7.

Figura 7 – Módulo *Inception*



Fonte: Adaptado de Szegedy *et al.* (2015)

3.7 Métricas

Para entender melhor as métricas para problemas de aprendizado de máquina, é necessário entender a Tabela 2, conhecida como matriz de confusão:

Tabela 2 – Matriz de Confusão

	Classe 1	Classe 2
Preditos pela rede como positivo	TP	FP
Preditos pela rede como negativo	FN	TN

Fonte: Elaborado pelo autor

Os valores da Tabela 2 podem ser entendidos da seguinte forma: para os valores TP (do inglês, *True Positives*) ou verdadeiros positivos, e TN (do inglês, *True Negative*), ou verdadeiros negativos, o classificador os classificou corretamente. Os valores, FP (do inglês, *False Positive*) ou falsos positivos, e FN (do inglês, *False Negative)*, ou falsos negativos, são aqueles que o classificador classificou erroneamente. Um exemplo de valores para uma matriz de confusão pode ser observada na Tabela 3 com classes: gato, cachorro e coelho.

No conjunto de dados referente a Tabela 3 tem 8 instâncias que são gatos, no entanto apenas 5 foram classificadas como gato, e 3 foram classificadas erroneamente como cachorros. Já para a classe Cachorro, tem-se 6 instâncias. Em que 3 foram classificadas corretamente como cachorro, no entanto outras 2 foram classificadas como gato e 1 erroneamente como coelho. Para a classe coelho, o conjunto apresenta 13 instâncias. Porém, 11 das 13 são classificadas corretamente como coelho. As outras 2 são erroneamente classificadas como cachorro. Por meio da matriz, é possível verificar o quanto o classificador erra e acerta para cada classe.

Tabela 3 – Matriz de confusão para um exemplo multi classe

	Gato	Cachorro	Coelho
Preditos pela rede como gato	5(TP)	2(FP)	0(FP)
Preditos pela rede como cachorro	3(FN)	3(TP)	2(FP)
Preditos pela rede como coelho	0(FN)	1(FN)	11(TP)

Fonte: Elaborado pelo autor

Tipicamente, um conjunto de dados é dividido aleatoriamente em três conjuntos de dados: treino, teste, e validação. O de treino, como o próprio nome sugere, é usado para treinar o algoritmo e escolher seus hiper-parâmetros. O conjunto de dados de teste é utilizado para coletar métricas do modelo final. O último conjunto de dados se trata do conjunto de validação. Este é utilizado para verificar se o modelo está convergindo com seus hiper-parâmetros, ou por exemplo,

para reportar qual a acurácia do modelo treinado. No caso deste trabalho, os dados de treino foram divididos em treino e validação: 20% dos dados de treino foram utilizados na validação, o restante continuou como treino. As métricas escolhidas, que são comumente utilizadas, serão explicadas a seguir.

3.7.1 Acurácia

Ao se tratar de problemas de classificação, a acurácia trata da quantidade de elementos classificados corretamente dividido pela quantidade total de elementos no *dataset*. A fórmula da acurácia é a seguinte:

$$\frac{TP+TN}{TP+FP+FN+TN} \tag{3.5}$$

3.7.2 Log Loss

A métrica *Log Loss* é utilizada pela plataforma *kaggle* para realizar o ranqueamento dos resultados de cada participante. Ela é calculada com base na classe real do objeto e nas probabilidades geradas pela solução de cada participante. A fórmula para o cálculo de *Log Loss* é:

$$-\sum_{c=1}^{M} y_{o,c} \log(_{p_{o,c}}) \tag{3.6}$$

Onde, $\log_{p_{o,c}}$ significa o logaritmo da probabilidade de o ser da classe c, $y_{o,c}$ é um indicador binário que informa se o o pertence a classe c, e M o número de possíveis classes.

3.8 Considerações

Neste Capítulo, foram apresentados os principais conceitos para entender o presente trabalho. Nas Seção 3.1 foi apresentado o que era Aprendizado Supervisionado, na Seção 3.2 foi apresentado o que é uma Rede Neural e seu funcionamento. Na Seção 3.3 foi apresentado o que é a função de ativação ReLU. Nas Seções 3.4, 3.5 e 3.6 são apresentadas as três redes usadas nesse trabalho. Na Seção 3.7 são apresentadas as métricas usadas para a avaliação das Redes Neurais. Com estes conceitos é possível compreender os trabalhos relacionados, no próximo capítulo, o Capítulo 4.

4 TRABALHOS RELACIONADOS

Classificação de objetos em imagens, ou mesmo vídeos, não é novidade na computação. Todavia, realizar essa tarefa fazendo o uso de Redes Neurais Convolucionais é algo relativamente novo. Um dos primeiros trabalhos com bom resultado é o de Krizhevsky *et al.* (2012). No trabalho de (KRIZHEVSKY *et al.*, 2012), os autores treinaram uma CNN para classificar um *dataset* de 1,2 milhões de imagens de alta resolução do (RUSSAKOVSKY *et al.*, 2015) de 1000 diferentes categorias. Nos dados de teste, obteve-se uma taxa de *top-1* e *top-5* erro de 37.5% e 17%, respectivamente, o que representava até o momento da publicação do trabalho o melhor resultado no estado-da-arte. Este trabalho está relacionado, pois se trata de uma das primeiras CNNs que obtém uma classificação melhor que os demais algoritmos. Além disso, ambos os trabalhos realizam classificação de imagens em várias classes. Eles se diferem, pois o trabalho de Krizhevsky *et al.* (2012) utiliza de redes rasas se comparadas com as utilizadas neste trabalho.

Em (KARPATHY *et al.*, 2014), foi realizado o treinamento de uma CNN sobre uma base de dados de 1,13 milhão de vídeos de esportes do *YouTube*, onde cada vídeo pertencia a uma ou mais classes dentre 487 existentes. Diferentemente de imagens, vídeos são espaço-temporais. Então os autores utilizaram técnicas para estender a conectividade das redes. A melhor rede criada apresentou uma melhora de resultado de 55.3% para 63.9% sobre algoritmos de *baselines*. Mas apenas uma leve melhora de 59.3% para 60.9% em relação à modelos de *single-frame*, que é uma imagem estática. A interseção entre este trabalho e o de (KARPATHY *et al.*, 2014) ocorre no uso de Redes Neurais Convolucionais. A principal diferença é que o presente trabalho utiliza um *dataset* de imagens, enquanto o relacionado utiliza um *dataset* de vídeos, o que gera um custo de processamento muito maior.

O trabalho de (SUN *et al.*, 2014) propõe que se crie um conjunto de representações de características de alto nível, utilizando redes profundas para reconhecimento de faces. A base de dados usada nesse trabalho foi a mesma do (SUN *et al.*, 2013), que contém 87.628 imagens de 5436 celebridades, com aproximadamente 16 fotos por celebridade. O resultado obtido com as redes neurais foram de 97.45%, o que se aproxima do resultado obtido com seres humanos, que é de 97.53%. O trabalho de (SUN *et al.*, 2014) se assemelha a este, pois ambos tratam do uso de Redes Neurais Convolucionais para classificação de dados entre multi classes. As diferenças entre os trabalhos está no fato deste utilizar técnicas já desenvolvidas pela comunidade, enquanto o trabalho relacionado cria um novo conjunto de características de alto nível para reconhecimento

de faces. Ou seja, cria novas redes convolucionais, não partindo de redes pré-definidas.

4.1 Considerações

As diferenças entre os trabalhos relacionados e o presente está no fato deste utilizar técnicas já desenvolvidas pela comunidade, enquanto os demais criam novas técnicas. Além disso, este trabalho faz um comparativo entre redes neurais já conhecidas, enquanto nenhum dos trabalhos relacionados faz isso. Vale ressaltar que todos os trabalhos utilizam Redes Neurais Convolucionais, e todos são multi classes. Apenas o trabalho de (KARPATHY *et al.*, 2014) classifica vídeos. O próximo capítulo, o Capítulo 5 apresenta os passos necessários para a execução deste trabalho.

Na Tabela 4 é apresentado uma comparação entre os trabalhos com o atual.

Tabela 4 – Tabela comparativa entre os trabalhos relacionados.

	Krizhevsky *et al.* (2012)	Karpathy *et al.* (2014)	Sun *et al.* (2014)	Trabalho Proposto
Tipo de Rede.	Rede Neural Convolucional	Rede Neural Convolucional	Rede Neural Convolucional	Rede Neural Convolucional
Tipo de Classificação	Multi Classes	Multi Classes	Multi Classes	Multi Classes
Tipo do *dataset*	Imagens	Vídeos	Imagens	Imagens
Define um novo tipo de rede neural	Sim	Sim	Sim	Não
O objetivo do trabalho é fazer um compartivo entre diversas CNN's	Não	Não	Não	Sim

Fonte: Informado pelo autor

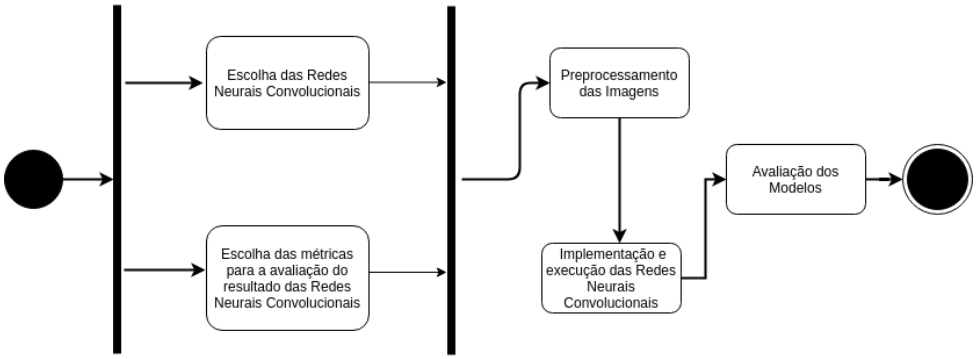
5 PROCEDIMENTOS METODOLÓGICOS

Este capítulo apresenta os passos necessários para a execução deste trabalho. A Figura 8 é um fluxograma onde se pode observar os passos necessários para a execução do trabalho. Os passos são os seguintes:

1. Escolha das Redes Neurais Convolucionais;
2. Escolha das métricas para a avaliação do resultado das Redes Neurais Convolucionais;
3. Preprocessamento das Imagens;
4. Implementação e execução das Redes Neurais Convolucionais;
5. Avaliação dos Modelos;

Vale ressaltar que os passos 1 e 2 acontecem paralelamente, mas não necessitam que sejam executados em paralelo.

Figura 8 – Fluxograma dos procedimentos metodológicos



Fonte: Elaborado pelo Autor.

5.1 Escolha das Redes Neurais Convolucionais

Existem várias arquiteturas de redes neurais definidas por diversos autores. Portanto, é praticamente impossível fazer um estudo comparativo utilizando todos as arquiteturas pré definidas. O objetivo desse passo é escolher quais redes neurais serão utilizadas nesse trabalho.

5.2 Escolha das métricas para a avaliação do resultado das Redes Neurais Convolucionais

Para avaliar algum produto e/ou serviço é necessário definir medidas, também conhecidas como métricas. No contexto de modelos de redes neurais existem diversas métricas

que podem ser utilizadas. O objetivo desse passo é encontrar as melhores métricas que possam ser utilizadas para realizar a avaliação da qualidade das redes neurais.

5.3 Preprocessamento das imagens

O conjunto de treino de imagens contém aproximadamente 10 mil imagens para treino, e cada imagem pertence a uma das 120 classes que existem no conjunto de dados, ou seja aproximadamente 85 imagens por raça de cachorro. Entretanto existem raças com pouco mais de 60 imagens, enquanto existem raças com mais de 120 imagens, ou seja, a distribuição não é uniforme, o que pode prejudicar o treinamento das redes, já que as raças não estão representadas em quantidades iguais.

Para solucionar esse problema foi necessário utilizar a técnica de *data augmentation*, que é capaz de produzir novas imagens a partir da rotação ou cisalhamento das imagens de treino, por exemplo, para evitar *overfitting* que é quando o modelo fica extremamente ajustado aos dados de treino, ou seja, ao classificar novos dados.

Além de usar de *data augmentation*, foi necessário redimensionar o tamanho da imagem, pois seria impossível executar redes mais profundas, como a InceptionV3, devido a um consumo maior de RAM.

5.4 Implementação e execução das Redes Neurais Convolucionais

Para a implementação das redes profundas foi utilizada a biblioteca *Keras*[1] em Python no ambiente de desenvolvimento Jupyter. A escolha da biblioteca se deu por conta dela ser amplamente utilizada no mercado e na academia. Além da linguagem Python ser uma das mais utilizadas para implementação de algoritmos de Aprendizado de Máquina. Para a execução das redes foram utilizados os kernels do Kaggle (que são ambientes computacionais em nuvem que permitem execução de códigos nas linguagens Python e R), que fornecem 14 *Gigabytes* de memória *RAM (Random Access Memory)*, e uma *GPU (Graphics Processing Unit)* NVIDIA Tesla K80.

Para a inicialização dos pesos das redes foram utilizados os pesos dos modelos já treinados no ImageNet, uma vez que o *Keras* disponibiliza os pesos de treinamento do ImageNet. Recentemente, CNN's têm sido amplamente utilizadas para extrair características de imagens

[1] https://keras.io/

e construir modelos capazes de classificar eficientemente objetos nestas imagens. Em 90% ou mais das aplicações, não é necessário se preocupar em propor e construir uma rede profunda convolucional. Ao invés de criar uma arquitetura própria para determinado problema, é possível verificar inicialmente qualquer arquitetura já proposta na competição *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) que apresente bom desempenho, baixar um modelo pré-treinado e ajustá-lo aos seus dados. Raramente é necessário treinar uma rede profunda convolucional a partir do zero ou criar uma arquitetura a partir do zero (FEI-FEI *et al.*, 2018).

5.5 Avaliação dos Modelos

Após o treinamento das redes, foi realizada a classificação do conjunto de testes disponível na mesma plataforma. Ao final, os resultados foram submetidos na plataforma para ranqueamento. Para escolher a melhor configuração de cada rede neural foi necessário treinar a rede diversas vezes com a mesma entrada, com mesmo número de interações, também conhecidos como épocas, só que com configurações diferentes. Para cada rede gerada, foram coletadas todas as métricas definidas do passo anterior. Para a escolha da melhor rede para o problema *Dog Breed Identification*, foi necessário verificar quais dos modelos gerados pelas redes utilizadas, apresentaram os melhores valores nas métricas.

6 RESULTADOS

Neste Capítulo são apresentados os resultados encontrados nos experimentos.

6.1 Escolha das Redes Neurais Profundas

Nas últimas edições do ImageNet, as soluções vencedoras foram as Redes Neurais Convolucionais, dentre elas:

1. VGG16 (SIMONYAN; ZISSERMAN, 2014)
2. InceptionV3 (SZEGEDY *et al.*, 2015)
3. ResNet50 (HE *et al.*, 2016)

6.2 Escolha das métricas para a avaliação do resultado das Redes

As métricas que serão utilizadas nesse trabalho são:

1. Acurácia
2. *Log Loss*

6.3 Preprocessamento das imagens

Devido a uma limitação de RAM, o tamanho das imagens não poderia ser maior que 197x197x3. Esse número foi obtido realizando testes com as redes. Este foi o maior número onde foi possível executar todas as redes. Vale ressaltar que a entrada padrão das imagens para as redes VGG16 e ResNet50 são 224x224, enquanto a InceptionV3 é 299x299, ou seja, apenas a InceptionV3 teve uma perda grande no tamanho da imagem.

Para utilizar de *Data Augmentation* foi utilizado o módulo *preprocessing.image.Image DataGenerator* do Keras, que leva como argumentos as transformações que serão utilizadas. As transformações utilizadas, e seus respectivos valores são apresentados a seguir, os valores escolhidos foram os valores que o criador do Keras recomenda em um de seus tutoriais (CHOLLET, 2016).

- *horizontal_flip* (booleano), na qual gera um giro horizontal, o valor utilizado foi verdadeiro;
- *zoom_range*, que representa um intervalo para um zoom aleatório, o valor utilizado neste parâmetro foi 0.2;
- *rotation_range*, na qual define um intervalo em graus para uma rotação aleatória, o valor

escolhido foi 30.

- *width_shift_range* e *height_shift_range*, representam um intervalo que pode ocorrer um corte, tanto horizontalmente como verticalmente, o valor utilizado neste parâmetro foi 0.2.

6.4 Implementação e execução das Redes Neurais Profundas

Cada rede rede foi executada 7 vezes, utilizando as seguintes configurações:

1. Modelo "puro"
2. Modelo com *Data Augmentation*
3. Modelo com Regularização
4. Modelo com *Data Augmentation* e regularização
5. Modelo com *Data Augmentation*, regularização e *Dropout* 0.1
6. Modelo com *Data Augmentation*, regularização e *Dropout* 0.3
7. Modelo com *Data Augmentation*, regularização e *Dropout* 0.5

O uso de *Data Augmentation* ocorreu devido existirem poucas imagens por classe no conjunto de treino, o que pode reduzir a capacidade das redes neurais em extrair características de baixo nível das imagens, consequentemente, diminuindo a capacidade de identificar características de alto nível, e realizar boas predições. O uso de regularização e *Dropout* seu deu para evitar *overfitting* (CHOLLET, 2017). A regularização, nesse caso, se refere especificamente a L1 e L2.

6.5 Coleta das métricas

Após o treino das redes, foram coletadas as métricas *Log Loss* e Acurácia da última época da rede. Os resultados podem ser vistos nas Tabelas 5 (resultado obtido com o RMSProp) e 6 (resultado obtido com o SGD). Os valores coletados na base de validação podem ser observados nas Tabelas 7 (resultado obtido com o RMSProp) e 8 (resultado obtido com o SGD). E por fim os valores obtidos na etapa de testes podem ser observados na Tabela 9. Não foi possível verificar a acurácia no conjunto de testes pois o problema não informa quais são as classes dos objetos de teste, para descobrir o *log loss* foi necessário submeter as predições de cada rede no site de submissão da competição[1].

A VGG16, no conjunto de testes, quando utilizando SGD informou que a probabili-

[1] https://www.kaggle.com/c/dog-breed-identification/submit

dade de qualquer imagem pertencer a classe Spaniel Bretão é de 1, ou seja, para esta configuração todas as imagens pertencem a mesma classe, entretanto existem apenas 73 imagens de Spaniel Bretão no conjunto de treino, já quando se faz uso de RMSProp, aproximadamente 90% dos objetos são classificados como Mabeco e 10% como Bluetick Coonhound. Já quando faz uso de regularização, *data augmentation* e utiliza o RMSProp, ela classificou todas as imagens como Leão-da-Rodésia, que apresenta 88 no conjunto de treino. O mesmo acontece quando se utiliza regularização, *data augmentation* e *Dropout* 0.5 que classificou os cachorros Basset Hound e Schipperke quando utilizou RMSProp e SGD, respectivamente.

No conjunto de testes quando a InceptionV3 utilizava de *data augmentation*, regularização e RMSProp retornou praticamente as mesmas probabilidades para todos os elementos, por exemplo, a probabilidade de uma imagem ser Boston bull era a mesma para todas as imagens. Isso também ocorreu quando ela utilizava de *data augmentation*, regularização, RMSProp e *Dropout* 0.1.

Tabela 5 – Resultados obtidos durante o treinamento das redes neurais usando RMSProp

	VGG16	InceptionV3	ResNet50
Modelo "Puro"	***Log Loss*: 15.98 Acurácia: 0.01**	*Log Loss*: 5.14 Acurácia: 0.01	***Log Loss:* 1.57 Acurácia: 0.58**
Modelo com *Data Augmentation*	***Log Loss*: 15.98 Acurácia: 0.01**	*Log Loss*: 5.15 Acurácia: 0.01	*Log Loss*: 4.03 Acurácia: 0.09
Modelo com Regularização	*Log Loss*: 16.05 Acurácia: 0.01	*Log Loss*: 4.85 Acurácia: 0.01	*Log Loss*: 10.4 Acurácia: 0.7
Modelo com Regularização e *Data Augmentation*	*Log Loss*: 16.02 Acurácia: 0.01	*Log Loss*: 4.83 Acurácia: 0.01	*Log Loss*: 9.27 Acurácia: 0.08
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.1	*Log Loss*: 16.03 Acurácia: 0.01	*Log Loss*: 4.84 Acurácia: 0.01	*Log Loss*: 9.29 Acurácia: 0.09
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.3	*Log Loss*: 16.02 Acurácia: 0.01	***Log Loss*: 4.72 Acurácia: 0.08**	*Log Loss*: 9.67 Acurácia: 0.11
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.5	*Log Loss*: 16 Acurácia: 0.01	*Log Loss*: 4.84 Acurácia: 0.01	*Log Loss*: 7.86 Acurácia: 0.11

Fonte: Elaborado pelo autor

Tabela 6 – Resultados obtidos durante o treinamento das redes neurais usando SGD

	VGG16	InceptionV3	ResNet50
Modelo "Puro"	***Log Loss***: **16** **Acurácia: 0.01**	***Log Loss*: 0.01** **Acurácia: 0.99**	***Log Loss*: 0.07** **Acurácia: 0.99**
Modelo com *Data Augmentation*	***Log Loss***: **16** **Acurácia: 0.01**	*Log Loss*: 0.3 Acurácia: 0.91	*Log Loss*: 4.03 Acurácia: 0.09
Modelo com Regularização	*Log Loss*: 16.36 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.40 Acurácia: 0.99
Modelo com Regularização e *Data Augmentation*	*Log Loss*: 16.30 Acurácia: 0.01	*Log Loss*: 0.66 Acurácia: 0.91	*Log Loss*: 0.48 Acurácia: 0.98
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.1	*Log Loss*: 16.50 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.65 Acurácia: 0.95
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.3	*Log Loss*: 16.34 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.66 Acurácia: 0.95
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.5	*Log Loss*: 16.29 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.63 Acurácia: 0.95

Fonte: Elaborado pelo autor

Tabela 7 – Resultados obtidos durante a etapa de validação das redes neurais usando RMSProp

	VGG16	InceptionV3	ResNet50
Modelo "Puro"	*Log Loss*: 16 Acurácia: 0.01	***Log Loss*: 0.01 Acurácia: 0.99**	***Log Loss*: 0.01 Acurácia: 0.99**
Modelo com *Data Augmentation*	***Log Loss*: 15.98 Acurácia: 0.01**	*Log Loss*: 0.3 Acurácia: 0.91	*Log Loss*: 4.03 Acurácia: 0.09
Modelo com Regularização	*Log Loss*: 16.07 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.40 Acurácia: 0.99
Modelo com Regularização e *Data Augmentation*	*Log Loss*: 16.02 Acurácia: 0.01	*Log Loss*: 0.66 Acurácia: 0.9	*Log Loss*: 0.48 Acurácia: 0.98
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.1	*Log Loss*: 16.50 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.65 Acurácia: 0.95
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.3	*Log Loss*: 16.34 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.66 Acurácia: 0.95
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.5	*Log Loss*: 16.3 Acurácia: 0.01	*Log Loss*: 0.39 Acurácia: 0.99	*Log Loss*: 0.64 Acurácia: 0.95

Fonte: Elaborado pelo autor

Tabela 8 – Resultados obtidos durante a etapa de validação das redes neurais usando SGD

	VGG16	InceptionV3	ResNet50
Modelo "Puro"	*Log Loss*: 16.01 Acurácia: 0.01	*Log Loss*: 1.45 Acurácia: 0.65	*Log Loss*: 2.17 Acurácia: 0.54
Modelo com *Data Augmentation*	***Log Loss*: 15.98 Acurácia: 0.01**	***Log Loss*: 0.11** Acurácia: 0.97	***Log Loss*: 0.09** Acurácia: 0.98
Modelo com Regularização	*Log Loss*: 16.37 Acurácia: 0.01	*Log Loss*: 1.81 Acurácia: 0.65	*Log Loss*: 2.17 Acurácia: 0.61
Modelo com Regularização e *Data Augmentation*	*Log Loss*: 16.3 Acurácia: 0.01	*Log Loss*: 0.44 **Acurácia: 0.98**	*Log Loss*: 0.43 **Acurácia: 0.99**
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.1	*Log Loss*: 16.49 Acurácia: 0.01	*Log Loss*: 1.84 Acurácia: 0.65	*Log Loss*: 0.62 Acurácia: 0.95
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.3	*Log Loss*: 16.33 Acurácia: 0.01	*Log Loss*: 1.79 Acurácia: 0.66	*Log Loss*: 0.58 Acurácia: 0.97
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.5	*Log Loss*: 16.3 Acurácia: 0.01	*Log Loss*: 1.76 Acurácia: 0.66	*Log Loss*: 0.6 Acurácia: 0.97

Fonte: Elaborado pelo autor

Tabela 9 – *Log Loss* obtido durante a etapa de teste das redes neurais usando SGD e RMSProp

	VGG16	InceptionV3	ResNet50
Modelo "Puro"	RMSProp: 34.24 SGD: 34.28	RMSProp: 6.62 SGD: 1.48	RMSProp: 7.72 SGD: 2.06
Modelo com *Data Augmentation*	RMSProp: 34.26 SGD: 34.25	RMSPROP: 4.79 SGD: 1.42	RMSProp: 4.37 SGD: 3.14
Modelo com Regularização	RMSProp: 34.27 SGD: 34.28	RMSProp: 4.8 SGD: 1.41	RMSProp: 6.84 **SGD: 1.76**
Modelo com Regularização e *Data Augmentation*	RMSProp: 34.26 SGD: 34.25	RMSProp: 4.79 SGD: 1.64	RMSProp: 4.15 SGD: 2.59
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.1	RMSProp: 34.3 SGD: 34.28	RMSProp: 4.79 SGD: 1.44	RMSProp: 4.14 SGD: 2.63
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.3	**RMSProp: 34.2** SGD: 34.28	RMSProp: 4.82 **SGD: 1.4**	RMSProp: 4.03 SGD: 2.52
Modelo com Regularização, *Data Augmentation* e *Dropout* 0.5	RMSProp: 34.23 SGD: 34.31	RMSProp: 4.79 SGD: 1.42	RMSProp: 4.8 SGD: 2.84

Fonte: Elaborado pelo autor

7 CONCLUSÃO E DISCUSSÕES

Este trabalho tinha como objetivo descobrir qual a Rede Neural Convolucional e os melhores hiperparâmetros para o problema *Dog Breed Identification*, que continha aproximadamente 10 mil imagens para treino e outras 10 mil para teste, dividas em 120 classes. Tendo em vista o recente, e bem sucedido, uso de redes neurais convolucionais em problemas de visão computacional.

Para tal foi necessário descobrir quais eram as redes neurais que mais estavam sendo utilizadas ultimamente em problemas de visão computacional, em paralelo a isto foi necessário escolher as métricas que seriam utilizadas para a avaliação das redes, após isto foi necessário realizar um preprocessamento das imagens, implementá-las e verificar qual modelo apresentou melhores resultados.

Com base nos resultados apresentadas no Capítulo de Resultados, a rede VGG16 apresentou os piores resultados, nos conjuntos de treino e teste sua acurácia não conseguiu superar 0.01, e seu loss foi de aproximadamente 16 para todas, o que é um valor altíssimo, tendo em vista que a *baseline* deste problema era de 4.79 de *log loss*. Além disso, a VGG16 reporta um *log loss* de aproximadamente 34, o resultado dela ficou entre os piores de toda a competição.

Para as redes InceptionV3 e ResNet50, os resultados utilizando SGD foram melhores ao resultado do RMSProp, para a VGG16 algumas vezes o resultado da SGD (34.28) é levemente inferior ao RMSProp (34.24), entretanto, todos os resultados são ruins.

Na etapa de validação a ResNet50 apresentou resultados bastantes similares que a InceptionV3 usando o SGD e quando o *Dropout* não era usado. Entretanto, quando se utiliza de *Dropout* a acurácia da InceptionV3 cai consideravelmente, enquanto a da ResNet50 permance com seus 0.97, aproximadamente. O *Log loss* da InceptionV3 é baixo quando se utiliza de *data augmentation*, mas quando o modelo só conta com regularização, o *Log loss* sobe para 1.81, ao se utilizar ambos o *loss* chega a 0.43.

Considerando os resultados de testes, a InceptionV3 quando faz uso de SGD apresenta os melhores resultados dentre as três redes, sendo que seu *log loss* mais baixo é de 1.4, todavia, este resultado não ficaria bem *ranqueado* na competição, ficando na posição 880 de 1286. independente da configuração usada os resultados da InceptionV3 são parecidos, a exceção fica quando o modelo é "puro"e utiliza de RMSProp como otimizador, o *log loss* cai de 6.62 para aproximadamente 4.8 para as demais configurações de redes. Já a ResNet50 quando utilizou de RMSProp apresentou resultados melhores que a InceptionV3 todas as vezes com exceção

de quando o modelo está puro, isso na etapa de testes. O modelo puro quando utilizava SGD apresentou o segundo melhor *loss*.

Ainda com base nos dados de testes, é possível afirmar que para este problema, quando uma rede em seu modelo puro apresenta bons resultados alterações adicionais, como regularização e *Dropout*, não fazem muita diferença, apesar de apresentar uma leve melhora na maioria das vezes. O mesmo acontece quando um modelo apresenta resultados ruins, por mais que se faça alterações em sua arquitetura ou se utilize de *data augmentation* no conjunto de treino, o modelo continuará apresentando resultados ruins.

Como trabalhos futuros pretende-se analisar outras redes, como *MobileNets* definida por (HOWARD *et al.*, 2017), *DenseNets* definida por (HUANG *et al.*, 2017), e *NASNet* definidas em (ZOPH *et al.*, 2017). Outro trabalho futuro seria aumentar o quantidade de transformações da *data augmentation*.

REFERÊNCIAS

BEZERRA, E. Introdução à aprendizagem profunda. **Tópicos em Gerenciamento de Dados e Informações.**, Porto Alegre: Sociedade Brasileira de Computação, p. 57–86, 2016.

CHOLLET, F. **Building powerful image classification models using very little data**. 2016. Disponível em: <https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>.

CHOLLET, F. **Deep learning with python**. [S.l.]: Manning Publications Co., 2017.

DESHPANDE, A. **A Beginner's Guide To Understanding Convolutional Neural Networks**. 2016. Disponível em: <https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>.

FEI-FEI, L.; JOHNSON, J.; YEUNG, S. **A Beginner's Guide To Understanding Convolutional Neural Networks**. 2018. Disponível em: <http://cs231n.stanford.edu/>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GUILLAUMIN, M.; VERBEEK, J.; SCHMID, C. Is that you? metric learning approaches for face identification. In: IEEE. **Computer Vision, 2009 IEEE 12th international conference on**. [S.l.], 2009. p. 498–505.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

HUANG, G.; LIU, Z.; MAATEN, L. V. D.; WEINBERGER, K. Q. Densely connected convolutional networks. In: **CVPR**. [S.l.: s.n.], 2017. v. 1, n. 2, p. 3.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An introduction to statistical learning**. [S.l.]: Springer, 2013. v. 112.

KARPATHY, A.; TODERICI, G.; SHETTY, S.; LEUNG, T.; SUKTHANKAR, R.; FEI-FEI, L. Large-scale video classification with convolutional neural networks. In: **Proceedings of the IEEE conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1725–1732.

KRIZHEVSKY, A.; HINTON, G. **Learning multiple layers of features from tiny images**. [S.l.]: Citeseer, 2009.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1097–1105.

KUMAR, N.; BERG, A. C.; BELHUMEUR, P. N.; NAYAR, S. K. Attribute and simile classifiers for face verification. In: IEEE. **Computer Vision, 2009 IEEE 12th International Conference on**. [S.l.], 2009. p. 365–372.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th international conference on machine learning (ICML-10)**. [S.l.: s.n.], 2010. p. 807–814.

PEREZ, L.; WANG, J. The effectiveness of data augmentation in image classification using deep learning. **arXiv preprint arXiv:1712.04621**, 2017.

RASCHKA, S.; MIRJALILI, V. **Python machine learning**. [S.l.]: Packt Publishing Ltd, 2017.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group, v. 323, n. 6088, p. 533, 1986.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **CoRR**, abs/1409.1556, 2014. Disponível em: <http://arxiv.org/abs/1409.1556>.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. **The Journal of Machine Learning Research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

SUN, Y.; WANG, X.; TANG, X. Hybrid deep learning for face verification. In: IEEE. **Computer Vision (ICCV), 2013 IEEE International Conference on**. [S.l.], 2013. p. 1489–1496.

SUN, Y.; WANG, X.; TANG, X. Deep learning face representation from predicting 10,000 classes. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1891–1898.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2015. p. 1–9.

TIELEMAN, T.; HINTON, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. **COURSERA: Neural networks for machine learning**, v. 4, n. 2, p. 26–31, 2012.

WONG, S. C.; GATT, A.; STAMATESCU, V.; MCDONNELL, M. D. Understanding data augmentation for classification: when to warp? **arXiv preprint arXiv:1609.08764**, 2016.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 818–833.

ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; LE, Q. V. Learning transferable architectures for scalable image recognition. **arXiv preprint arXiv:1707.07012**, Technical Report, v. 2, n. 6, 2017.