UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

RODRIGO EMERSON VALENTIM DA SILVA

UM ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA A CLASSIFICAÇÃO DE IMAGENS

QUIXADÁ

2018

RODRIGO EMERSON VALENTIM DA SILVA

UM ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA A CLASSIFICAÇÃO DE IMAGENS

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

Orientadora: Prof. 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)

S583e Silva, Rodrigo Emerson Valentim da.
Um estudo comparativo entre redes neurais convolucionais para a classificação de imagens / Rodrigo Emerson Valentim da Silva. – 2018.
51 f. : il. color.

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

1. Aprendizagem Profunda. 2. Rede Neural Convolucional. 3. Imagem-Classificação. I. Título.
CDD 004.6

RODRIGO EMERSON VALENTIM DA SILVA

UM ESTUDO COMPARATIVO ENTRE REDES NEURAIS CONVOLUCIONAIS PARA A CLASSIFICAÇÃO DE IMAGENS

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

Aprovada em: __/__/__

BANCA EXAMINADORA

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

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

Prof. Tércio Jorge da Silva
Universidade Federal do Ceará (UFC)

À minha família, namorada, amigos e professores. Obrigado por tudo.

AGRADECIMENTOS

Primeiramente agradeço a Deus, por ter chegado onde cheguei.

A minha avó Maria de Lourdes, por tudo que fez por mim, ter me criado, educado e influenciado na pessoa que eu sou. Na ausência de pai e mãe, a senhora conseguiu ser os três.

Agradeço a minha mãe Adriana Valentim, pois mesmo tão distante na minha infância nunca escondeu que seu principal objetivo era dar uma condição de vida melhor para a nossa família e, infelizmente, às vezes as coisas não saem como queremos. Continue sendo a sonhadora que me motiva.

Agradeço ao meu pai José Haroldo, uma figura paterna que mesmo distante, sempre buscou uma forma de me aconselhar, sempre querendo que eu seguisse o caminho do bem. Obrigado por ter me aconselhado tão bem e por ser tão sincero comigo.

Agradeço a minha tia Tatiana Valentim, por ter me dado todo suporte para que eu conseguisse estudar, que me deu todo amor me fazendo sentir-se mais um filho dentro da sua casa.

Agradeço a minha irmã Rayce Karolaini, por ser a minha melhor amiga e ter cuidado tão bem da nossa avó. Com certeza a nossa vó está no céu extremamente grata por tudo que você fez.

Agradeço a minha namorada Patrícia Queiroz, por ser uma companheira compreensível e amável, obrigado por cuidar bem de mim.

Agradeço especialmente à minha orientadora Ticiana Linhares, por todos os conhecimentos que me ensinou, pela paciência, pelo tempo que dedicou a mim com seu excelente trabalho e por ser uma grande inspiração na minha carreira acadêmica.

Agradeço aos professores participantes da banca examinadora Regis Pires e Tércio Jorge pelo tempo, pelas valiosas colaborações e sugestões.

Agradeço aos meus amigos que residem comigo Lincoln Silva, Ítalo Oliveira, Natanael Silva e Françoes Pereira, vocês são uma família pra mim.

Agradeço aos meus amigos da UFC Fagner, João Mateus, Naélio, Andreazo, Gabriel, Luís, Lucivan e Igor Lima, graças a nossa união, eu cheguei aqui.

Agradeço a Universidade Federal do Ceará, e a todos os funcionários e docentes que ali trabalham.

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

O reconhecimento de objetos em imagens está se tornando cada vez mais fácil para os computadores, devido aos avanços em estudos de visão computacional que utilizam redes neurais de aprendizagem profunda. Essas redes tiveram grandes avanços a partir de 2012 e vêm sendo usadas em diversos problemas de classificações que envolvem imagens. Neste trabalho, são feitas três implementações de redes neurais convolucionais, sendo elas: VGG-16, ResNet-50 e *Inception*. O objetivo deste trabalho é realizar um comparativo entre as implementações, avaliando qual delas resolve melhor um problema de classificação de imagens, levando em consideração algumas métricas, para a competição de classificação de personagens dos *Simpsons*. Dentre as métricas escolhidas, o melhor resultado de acurácia foi o da ResNet50 com 93.4% e o melhor resultado de *log loss* foi o da *Inception* com 0.26, o que demonstrou que as redes conseguiram resolver bem o problema.

Palavras-chave: Aprendizagem profunda. Rede Neural Convolucional. Imagem-Classificação

ABSTRACT

The recognition of objects in images is becoming easier for the computers, due to the advances in studies of computational vision, that uses neural networks of deep learning. These networks had big advances since 2012 and have been used on different problems of classifications that evolve images. In this work, are made three implementations of convolutional neural networks, that are: VGG-16, RestNet-50 and Inception. The purpose of this work is making a comparative of the implementations, evaluating which one is better to solve a problem of image classification, considering some metrics, for the competition of classification of the Simpsons' characters. Between the chosen metrics, the better accuracy result was the ResNet50 with 93.4% and the best result of log loss was the Inception with 0.26, that showed that the networks were able to solve the problem just fine.

Keywords: Deep learning. Convolutional neural networks. Classification of images

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O reconhecimento de objetos em imagens é uma atividade simples para humanos (com *expertise* no assunto) e está se tornando cada vez mais fácil para os computadores, devido aos avanços em estudos de visão computacional que utilizam redes neurais de aprendizagem profunda.

Aprendizagem Profunda é uma sub-área da Aprendizagem de Máquina, que emprega algoritmos para processar dados se inspirando no processamento feito pelo cérebro humano, pois elas simulam um conjunto de neurônios conectados que são organizados em camadas, tal que cada camada faz o processamento de informações e passa o resultado de entrada para a camada seguinte (HAYKIN *et al.*, 2009).

Os precursores destes avanços foram Krizhevsky *et al.* (2012). Eles mostraram o poder destas redes, treinando uma das melhores Redes Neurais Convolucionais (CNN) até aquele ano e conseguiram os melhores resultados na competição *ImageNet Large Scale Visual Recognition Challenge*[1] (ILSVRC) em 2012. A competição *ImageNet* consiste em avaliar diversos algoritmos para detecção de objetos e classificação de imagens em larga escala. A motivação desta competição é permitir que os pesquisadores comparem o progresso na detecção em uma ampla variedade de objetos e medir o progresso da área de visão computacional.

A competição é feita anualmente e até hoje traz grandes avanços na área de visão computacional. Na competição, os participantes testam seus algoritmos de reconhecimento de objetos em imagens com a base de dados do ImageNet, que contém um conjunto de dados de mais de 14.2 milhões de imagens de alta resolução com cerca de 22 mil categorias (KRIZHEVSKY *et al.*, 2012).

Motivados por essa competição, várias soluções e algoritmos de alta precisão foram criadas como a do *Visual Geometry Group* (VGG) que propuseram arquiteturas de redes neurais profundas, sendo elas as redes VGG-16, VGG-19 (SIMONYAN; ZISSERMAN, 2014). Estas arquiteturas obtiveram os melhores resultados no ano de 2014. O *Google* propôs a arquitetura *Inception*, que conseguiu alcançar um desempenho muito bom com relativamente baixo custo computacional e rendeu um desempenho de última geração na edição da competição de 2015 (SZEGEDY *et al.*, 2017). No mesmo ano, He *et al.* (2016) foram os campeões com a proposta da ResNet. Esta solução apresentou o melhor desempenho no conjunto de critérios da competição.

Todas as redes citadas acima estão públicas e podem ser usadas para resolução de

[1] http://www.image-net.org/challenges/LSVRC/

diversos problemas, tais como: previsão da continuidade em sequências de vídeo (VILLEGAS *et al.*, 2017), classificação de lesões de pele (ESTEVA *et al.*, 2017). Outros tipos de redes neurais também são aplicadas para resoluções de outros problemas como é o caso das redes neurais recorrentes que são utilizadas para problemas como: tradução entre diferentes línguas (BRITZ *et al.*, 2017) e reconhecimento de fala (CHAN *et al.*, 2016). É possível utilizar as redes campeãs da ILSVRC, ajustando alguns parâmetros para outras bases de dados de imagens. As redes dessa competição já estão pré-treinadas a partir da base de dados do *ImageNet*, que é bastante ampla. Dessa forma, o trabalho é reduzido, pois não é preciso criar uma rede do zero.

Entretanto, cada conjunto de dados pode apresentar uma rede mais satisfatória ou que tem melhor desempenho. Visando contribuir para a escolha da mais indicada para cada problema que envolve imagens, neste trabalho foram feitas as implementações das redes neurais VGG-16, ResNet-50 e *Inception* com o objetivo de fazer um comparativo de qual delas melhor resolve um problema de classificação de imagens para a competição de classificação de personagens dos *Simpsons*[2]. O objetivo desta competição é criar um algoritmo para classificar qual é o personagem que aparece em uma determinada imagem. A escolha dessa competição se deu pelo fato dos dados se apresentarem bastante diferentes dos existentes no *ImageNet*. Dessa forma, é possível perceber o poder de generalização dessas redes para problemas bem específicos como é o caso deste trabalho.

O conjunto de dados para essa competição é composto com 11054 (onze mil e cinquenta e quatro) imagens para treino, mais 2757 (dois mil setecentos e cinquenta e sete) imagens para validação do algoritmo e 456 (quatrocentos e cinquenta e seis) imagens para teste. As redes deverão rotular essas imagens em 10 classes. Esta base de dados foi retirada do *Kaggle*[3] que é uma plataforma para competições de ciência de dados fundada em 2010, sendo hoje, um dos principais ambientes de colaboração entre cientistas de dados do mundo.

Neste trabalho, as redes implementadas foram comparadas usando as métricas *recall*, precisão, acurácia e *log loss*. Estas métricas são importantes para avaliar o desempenho das redes escolhidas. O público alvo deste trabalho são pesquisadores, profissionais e entusiastas na área de Aprendizado de Máquina, Aprendizado Profundo, e Inteligência Artificial que buscam estudar sobre Redes Neurais Convolucionais com o uso de redes pré-treinadas para resolver problemas de classificação de imagens.

O trabalho está estruturado da seguinte forma: o Capítulo 2 são expostos os conceitos

[2] https://www.kaggle.com/alexattia/the-simpsons-characters-dataset
[3] https://www.kaggle.com/

básicos que são necessários para a fundamentação do trabalho. No Capítulo 3 apresenta os trabalhos relacionados. O Capítulo 4 descreve a metodologia a ser utilizada. O capítulo 5 mostra os resultados, e no Capítulo 6 é feito as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, são introduzidos os conceitos, notações e definições fundamentais sobre rede neural convolucional. Além disso, será abordado os conceitos das métricas, as redes pré-treinadas utilizadas, os otimizadores escolhidos e o conceito de regularização.

2.1 Aprendizado Profundo

O aprendizado profundo é um subcampo específico de aprendizado de máquina que representa uma nova abordagem das representações de aprendizado a partir de dados. Nessa abordagem, é enfatizado o aprendizado através de camadas sucessivas. O termo aprendizado profundo, se refere ao número de camadas que um modelo tem, quanto mais camadas, mais profundo é o modelo. No aprendizado profundo, essas representações em camadas são geralmente Redes Neurais Artificiais (CHOLLET, 2017).

As Redes Neurais Artificiais (RNA), buscam modelar a forma de funcionamento do cérebro humano. Elas simulam um conjunto de neurônios conectados que são organizados em camadas, onde cada camada faz um processamento da informação e passa para a camada seguinte (HAYKIN *et al.*, 2009). Entretanto, Chollet (2017), deixa claro que não há evidências de que o cérebro implemente algo como os mecanismos de aprendizagem usados em RNAs, e prefere definir aprendizagem profunda como sendo uma estrutura matemática para aprender representações a partir de dados.

Uma RNA é composta por camadas que estão encadeadas e mapeiam os dados de entrada para fazer predições de acordo com as classes definidas. Cada camada de entrada da rede contém neurônios que codificam os valores de entrada, ou seja, as saídas de algumas camadas se tornam entradas para as camadas seguintes.

As camadas possuem parâmetros a serem estimados que são chamados de pesos, os quais são usados para armazenar o conhecimento adquirido nas etapas de aprendizagem. Estes pesos são ajustados na medida que as camadas vão extraindo *features* com as informações de entrada. Assim, a rede aprende com os dados de entrada e consegue estimar uma saída. No contexto de classificação de imagens, a rede pode estimar se uma imagem é de um cachorro ou um gato, por exemplo.

Neste trabalho são utilizadas redes convolucionais. Uma rede neural convolucional (CNN ou ConvNet) é uma classe de rede neural artificial. As CNNs são usadas para classificar

imagens, realizar reconhecimento de objetos ou até mesmo o reconhecimento de áudios. Em Chollet (2017), são apresentadas as quatro principais camadas de uma CNN: Convolução, Unidades Lineares Retificadas (do inglês *Rectified Linear Units* - ReLU), *Max-Pooling* e Camada Totalmente Conectada.

Na camada convolucional, os neurônios funcionam como filtros que são aplicados aos dados de entrada, que no caso deste trabalho, é uma imagem. Um filtro é uma matriz de pesos. O objetivo principal da convolução em uma CNN é extrair *features* da imagem de entrada (RASCHKA; MIRJALILI, 2017). A convolução preserva a relação espacial entre os pixels, aprendendo as características da imagem a partir de suas regiões diferentes. Na operação de convolução é gerado um mapa de *features*.

Para ficar mais claro, considere a matriz de entrada 5 x 5 como sendo uma imagem que será processada por uma rede convolucional.

$$entrada = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Considere também a matriz de filtro sendo a matriz de pesos que irá ser utilizada para encontrar o mapa de *features*.

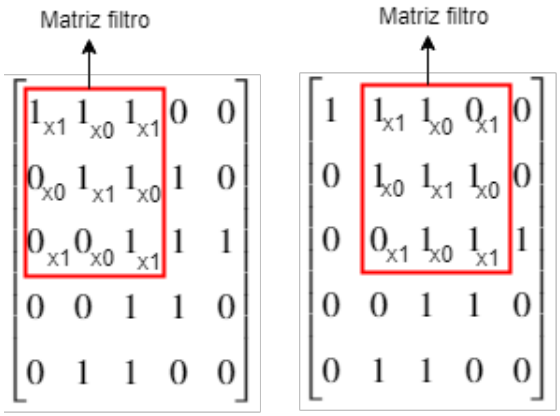
$$filtro = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

O processo para criar o mapa de *features*, que também é uma matriz, é feito com o produto de matrizes da matriz de entrada com a matriz de filtro, seguindo o seguinte procedimento: deslizar a matriz de pesos sobre a matriz de entrada pixel a pixel. Por exemplo, um filtro típico em uma camada convolucional pode ter tamanho 3x3x3, ou seja, 3 pixels de largura e altura, e o último 3 representando as cores primárias (RGB, por exemplo). Durante o processo de

convolução é deslizado cada filtro pela largura e altura da matriz de entrada, e assim são calculados os produtos de ponto entre as entradas do filtro e a entrada em qualquer posição. À medida que é deslizado o filtro pela largura e altura da matriz de entrada é produzido um mapa de características bidimensional que fornece as respostas desse filtro em todas as posições espaciais.

Para simplificar o entendimento, a Figura 1 mostra o processo de deslizar matriz filtro pela a a matriz de entrada. A Figura 1 mostra os dois primeiros passos para criação do mapa de características.

Figura 1 – Processo de convolução mostrado nos dois primeiros passos.



Fonte: Próprio autor

Na Figura 1, o quadrado vermelho e os valores menores representam a matriz filtro definida acima, esse processo termina quando a matriz filtro passar por toda a matriz de entrada. O resultado do processo é o mapa de recursos representado pela a seguinte matriz:

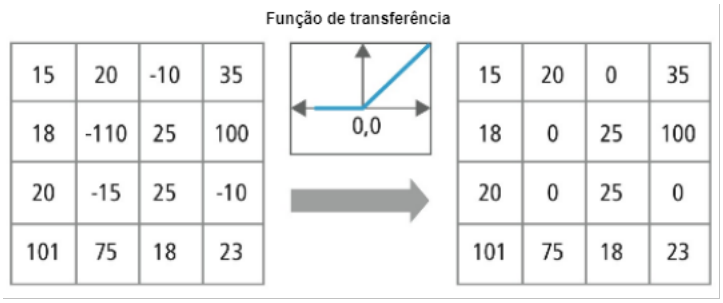
$$recursos = \begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix}$$

Com o mapa de *features* feito na camada convolucional, a operação de ReLU substitui todos os valores de pixels negativos no mapa por zero. O objetivo da ReLU é introduzir a não-linearidade na rede, isso é necessário, pois os dados que a CNN usa para aprender se comportam, na maioria das vezes, de forma não linear. A fórmula da ReLU é a seguinte:

$$f(x) = max(x, 0) \tag{2.1}$$

A Figura 2 mostra a aplicação da função ReLU em um mapa de características. Ela atribui zero a todos os valores negativos.

Figura 2 – Aplicação da função ReLU.



15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

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

Existem outras funções de ativação tais como: *sigmoides* e tangente hiperbólica, porém neste trabalho optou-se pela a ReLU, pois ela produz um erro de treinamento menor comparado com as outras funções, por conta de não ser tão suscetível ao problema da dissipação dos gradientes. Além disso, essa função de ativação reduz o tempo de convergência dos parâmetros, pois ela é simplesmente a função identidade para valores positivos (KRIZHEVSKY *et al.*, 2012).

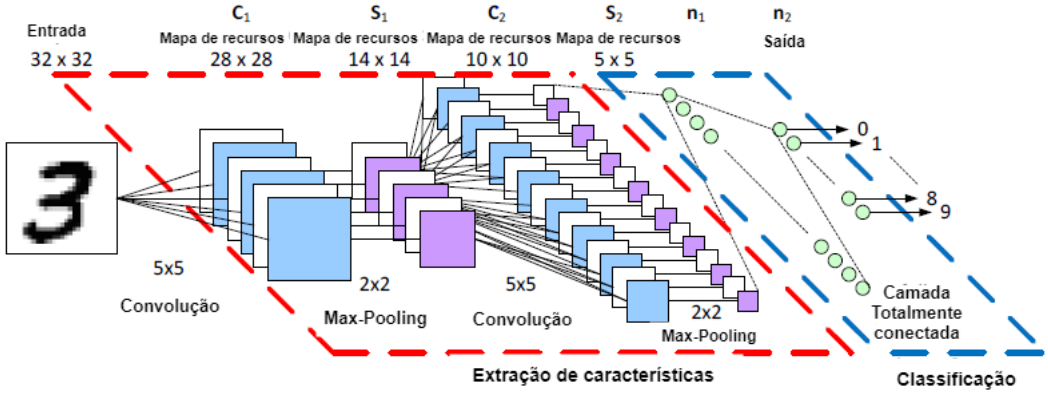
Depois de aplicar a ReLU, as redes convolucionais fazem o *Max-Pooling*, que consiste em extrair janelas dos mapas de características de entrada e gerar o valor máximo de cada canal (CHOLLET, 2017). Ou seja, o *Pooling* reduz a dimensionalidade de cada mapa de *features*, mas retém as informações mais importantes. Essa técnica permite criar representações de entrada menores e mais gerenciáveis, fazendo assim, o número de parâmetros e cálculos na rede reduzirem.

Por fim, a camada totalmente conectada usa uma função de ativação na camada de saída para a classificação. Está totalmente conectada implica dizer que todos os neurônios da camada anterior estão conectados a todos os neurônios da camada seguinte. Ela pega os resultados da camada de Convolução e da camada de *Pooling*, e então usa esses resultados para classificar a qual classe pertence a imagem de entrada.

A Figura 3 é um exemplo de uma CNN para o reconhecimento de dígitos. Ela recebe uma imagem de um dígito como entrada e tenta prever a qual classe ela é pertencente. Neste exemplo, são 10 classes uma para cada dígito de 0-9.

A Figura 3 representa uma CNN de reconhecimento de dígitos. Ela tem como entrada uma imagem com dimensões 32 x 32 e faz uma convolução com matrizes 5 x 5 que resulta no C1 que é o primeiro mapa de características com dimensões 28 x 28. Em seguida, é feito o passo de *Max-Pooling* que pega os valores mais altos e resulta no S1 com dimensões 14 x 14. Com o resultado de S1, a rede faz uma nova convolução que resulta em C2 e um novo *Max-Pooling*

Figura 3 – Exemplo de uma rede neural convolucional.



Fonte: Adaptado de: Peemen *et al.* (2011)

que resulta em S2. O passo final é a camada totalmente conectada que recebe os neurônios de saída representados pelos círculos verdes e os classifica de acordo com as classes estabelecidas no treinamento.

2.2 Inception

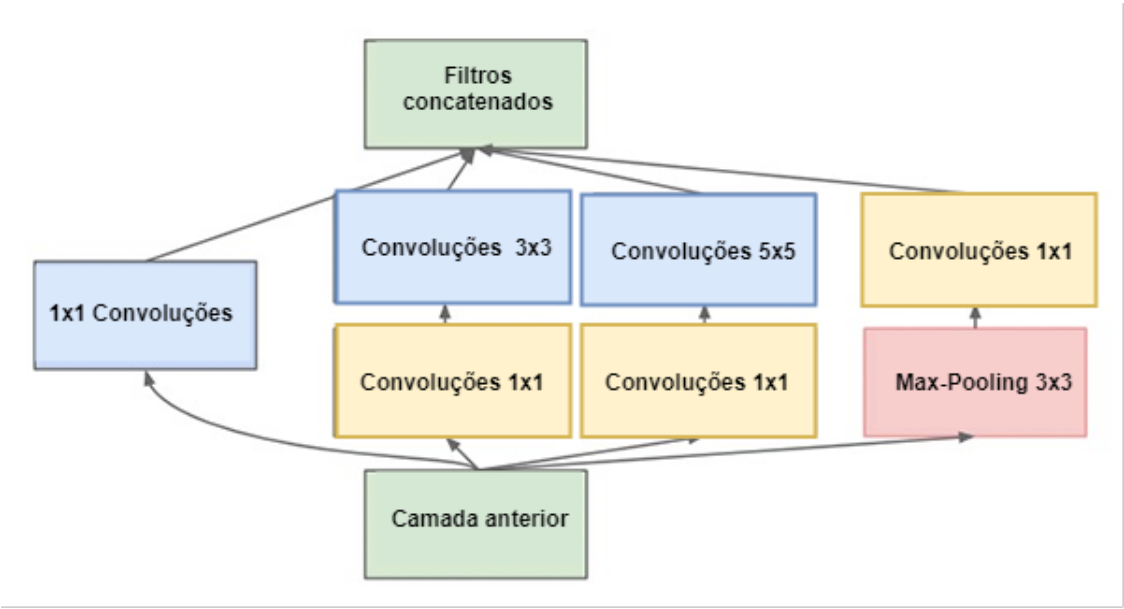
A rede GoogLeNet foi vencedora do ILSVRC no ano de 2014, ela é uma CNN de Szegedy *et al.* (2017). Sua principal contribuição foi o desenvolvimento de um módulo de *Inception* que reduziu drasticamente o número de parâmetros na rede para 4 milhões, comparado a rede AlexNet com 60 milhões. Além disso, esta rede usa o Average-Pooling em vez de camadas totalmente conectadas na parte superior da CNN, eliminando uma grande quantidade de parâmetros menos importante (CS231N, 2018).

Objetivo principal do módulo *Inception* é atuar como um extrator de características em vários níveis computando convoluções 1x1, 3x3 e 5x5 dentro do mesmo módulo da rede. A Figura 4 mostra a configuração do módulo.

Existem várias versões de redes que incorporam o módulo *Inception*, sendo a mais recente, chamada de *Inception-v4*. Porém, neste trabalho foi utilizada a versão *Inception-v3*, pois ela já se encontra na biblioteca *Keras*.

Inception-v3 é uma arquitetura voltada para a resolução de classificação de imagens que foi treinada no conjunto de dados *ImageNet*. Esta rede que apresentou bom desempenho com relativamente baixo custo computacional no ano de 2015, pois ela reduziu os parâmetros estimados pela rede, fazendo com que ela possua melhor desempenho computacional em relação as redes VGG durante seu treinamento (SZEGEDY *et al.*, 2017).

Figura 4 – Módulo *Inception*.



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

2.3 ResNet

A ResNet (do inglês *Residual Network*) é uma rede neural convolucional da *Microsoft* que venceu o concurso ILSRVC de 2015 no conjunto de dados *ImageNet*. O desempenho dessa rede foi superior ao de seres humanos.

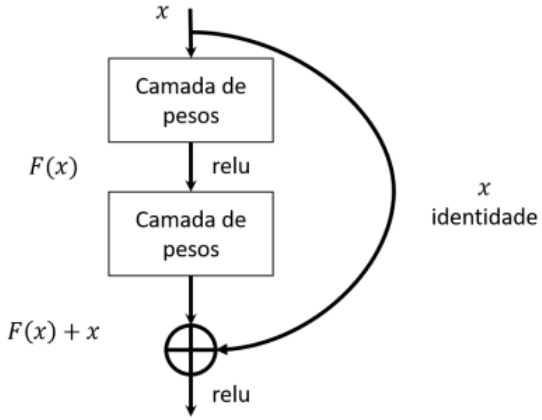
Quando redes mais profundas começam a convergir, o problema de degradação pode aparecer: com o aumento da profundidade da rede, a precisão fica saturada e, em seguida, degrada rapidamente. Inesperadamente, tal degradação não é causada pelo *overfitting*, e adicionar mais camadas ao modelo profundo ocasionaria maior erro de treinamento. A ResNet foi construída por blocos residuais, estes blocos tem uma entrada x que passa por uma série de operações de convolução-relu-convolução. O resultado da operação f(x) é adicionado à entrada original x. A Figura 5 mostra a arquitetura do bloco. Essas camadas adicionadas são o mapeamento de identidade e as outras camadas são copiadas do modelo mais raso aprendido. Esta solução indica que um modelo mais profundo não deve produzir um erro de treinamento maior do que sua versão da rede mais rasa (HE *et al.*, 2016).

A equação 2 mostra o processamento que o bloco residual faz.

$$H(x) = f(x) + x \tag{2.2}$$

Nas CNNs VGG16 e *Inception*, o resultado de processamento H(x) é igual a f(x), ou seja, espaço de saída é completamente alterado em relação ao espaço de entrada x. Já na ResNet a função f(x) é apenas uma regularização. Com isso, o espaço de saída é somente uma alteração

Figura 5 – Bloco residual da rede ResNet.



Fonte: Adaptado de: He *et al.* (2016).

do espaço de entrada. Assim, a ResNet mostra que os mapas residuais são mais fáceis de serem otimizados por meio dessa alteração no espaço de entrada (HE *et al.*, 2016).

Neste trabalho, foi implementada a ResNet50 que é uma das versões fornecidas que está na biblioteca *Keras*, sendo uma rede com 50 camadas.

2.4 Visual Geometry Group (VGG16)

A arquitetura de rede VGG utiliza camadas convolucionais 3 x 3 empilhadas umas sobre as outras com profundidade crescente (SIMONYAN; ZISSERMAN, 2014). VGG16 significa que a rede tem 16 camadas.

Em Simonyan e Zisserman (2014) foi investigado o efeito da profundidade da rede convolucional com relação a precisão no reconhecimento de imagem em larga escala. Foi constatada boa precisão para o reconhecimento de imagens utilizando redes de 16-19 camadas. A Tabela 1 mostra a configuração da rede VGG16.

A Tabela 1, mostra como é a arquitetura da rede VGG16. Nela é possível ver 13 camadas que fazem convolução retirando as *features* das imagens e mais 3 camadas totalmente conectadas para fazer as classificações.

Simonyan e Zisserman (2014) acharam o treinamento VGG16 e VGG19 desafiadores especificamente em relação à convergência nas redes mais profundas. Então, para facilitar o treinamento, eles primeiro treinaram versões menores de VGG com 11 camadas que serviram como inicializações para as redes maiores e mais profundas como a VGG16 e 19.

Neste trabalho, foi implementada esta rede pré-treinada, a fim de fazer o comparativo com as redes ResNet e *Inception* levando em consideração as métricas escolhidas.

Tabela 1 – Arquitetura da VGG-16

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: Próprio autor.

2.5 Métricas

Nesta subseção são apresentadas as métricas de classificação utilizadas neste trabalho para fazer o comparativo entre as redes VGG16, ResNet e Inception. As métricas são importantes para avaliar o desempenho das redes escolhidas. Nas tarefas de classificação, que é o caso deste trabalho, busca-se prever qual é a categoria a que uma amostra pertence como, por exemplo, determinar se uma imagem é de um cachorro ou de um gato. As fórmulas para calcular as métricas foram obtidas da obra Hackeling (2014).

Na subseção 2.5.1 do Capítulo 2, é apresentado o conceito de Matriz de Confusão, na subseção 2.5.2 é mostrado o conceito de acurácia, na subseção 2.5.3 é abordado o conceito de Precisão, na subseção 2.5.4 é mostrado o conceito de *recall* e subseção 2.5.5 o conceito de *log loss*.

2.5.1 Matriz de confusão entre as classes

Uma matriz de confusão compara as previsões de um modelo com o padrão verdadeiro. A diagonal da matriz representa classes que foram corretamente previstas (do inglês, *True Positive* - TP), enquanto elementos fora da diagonal representam indivíduos que foram classificados de forma errada (do inglês, *False Positive* - FP).

A partir da matriz de confusão, é possível calcular as métricas para avaliar se o algoritmo está ou não conseguindo bons resultados. A Figura 6 mostra um exemplo de matriz de confusão que será utilizadas nas Seções seguintes para fazer os cálculos das métricas.

Figura 6 – Matriz de confusão.

		Classe Real	
Classe Predita		Cão	Gato
	Cão	5 (TP)	1 (FP)
	Gato	2 (FN)	6 (TN)

Fonte: Próprio autor

Na Figura 6 mostra no lado esquerdo na vertical o valor predito que o algoritmo classificou e no topo o valor o valor real da classe. Ela ainda revela que para a classe cão, o algoritmo classificou 5 instâncias como cão e que realmente eram cães (TP). O algoritmo classificou uma instância como gato que na verdade era cão (do inglês, *False Positive* - FP). Para a classe gato o algoritmo classificou 2 instâncias como cão (FN) quando na verdade eram gatos e 6 como gatos que realmente eram gatos (do inglês, *True Negative* - TN). Os valores utilizados foram apenas para exemplificar. Considera-se nesse caso da matriz, que o resultado de classificação da classe cão é positiva e o da classe gato é negativo.

2.5.2 Acurácia

A Acurácia em problemas de classificação é o número de previsões corretas feitas pelo modelo sobre todos os tipos de previsões feitas.

Para calcular a acurácia de um algoritmo é utilizada a seguinte fórmula:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

2.5.3 Precisão

A precisão é a razão de previsões corretas (TP) para o total de observações positivas previstas da classe. Ou seja, no exemplo da Figura 6, quantos cães foram preditos pelo algoritmo que realmente são cachorros.

Para calcular a precisão do algoritmo com relação uma classe (por exemplo, classe positiva) é utilizado a seguinte fórmula:

$$P = \frac{TP}{TP+FP} \tag{2.4}$$

2.5.4 Cobertura

A cobertura (do inglês *recall*) é a razão do número de exemplos classificados como pertencentes a classe real, que realmente são daquela classe (TP, se for analisada a positiva), para o total de exemplos que pertencem a esta classe (TP+FN)

Para calcular o *recall* de uma classe (positiva, por exemplo) é utiliza a seguinte fórmula:

$$R = \frac{TP}{TP+FN} \tag{2.5}$$

2.5.5 Log Loss

Log Loss, é uma função de perda de classificação geralmente usada como uma métrica de avaliação nas competições do *Kaggle*. O sucesso nessas competições depende do quanto consegue-se minimizar a perda, ou seja, minimizar o *log loss*. Esta métrica é basicamente equivalente a maximizar a precisão do nosso classificador que no contexto deste trabalho será as redes pré-treinadas VGG-16, ResNet e *Inception*. A formula da *Log Loss* é:

$$-\sum_{c=1}^{M} y_{o,c} \log p_{o,c} \tag{2.6}$$

Onde, $\log_{p_{o,c}}$ significa a probabilidade de uma instância 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. Entre as métricas escolhidas, a *log loss* é a métrica de maior relevância, pois é essa métrica que as redes neurais tentam diminuir durante o processo de treinamento.

2.6 Otimizadores

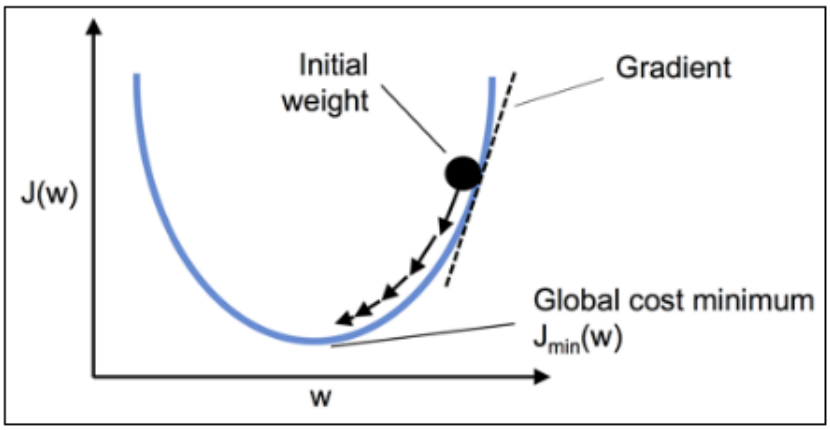
Um dos principais objetivos dos algoritmos de aprendizado de máquina supervisionados é otimizar o processo de redução de uma função objetivo, comumente chamada de função de custo J (RASCHKA; MIRJALILI, 2017). Reduzir a função J é importante, pois a rede precisa aprender os pesos que melhor representam o relacionamento entre os dados. Esses pesos formarão o modelo preditivo que permitirá que a rede faça previsões quando apresentada a novos conjuntos de dados.

Para descobrir esses pesos, a rede é treinada para fazer previsões o mais próximo possível dos valores reais (valores históricos usados durante o treinamento). Para medir isso, é preciso uma métrica de quão erradas são as previsões. No caso das Redes Neurais, essa métrica é representada pela função de custo *log Loss*:

$$-\sum_{c=1}^{M} y_{o,c} \log p_{o,c} \tag{2.7}$$

O algoritmo de otimização mais simples para encontrar os pesos é o Gradiente Descendente (GD). A Figura 7 a seguir, descreve que a ideia por trás é analogamente descer uma colina em busca do mínimo de custo local ou global. Em cada iteração, é dado um passo na direção oposta do gradiente onde o tamanho do passo é determinado pela taxa de aprendizagem (RASCHKA; MIRJALILI, 2017).

Figura 7 – Gradiente Descendente.



Fonte: Raschka e Mirjalili (2017)

Com a descida do gradiente, são realizados pequenos passos em direção ao mínimo global. Neste caso, a rede precisa alterar os pesos em etapas que reduzem o erro. Fazendo uma analogia, o erro é o vale que o algoritmo precisa chegar ao fundo. Uma vez que o caminho

mais rápido em uma montanha está na direção mais íngreme, as etapas tomadas devem estar na direção que minimiza o erro.

O GD atualizar os pesos, dando um passo na direção oposta do gradiente ΔJ(w) da função de custo J (w):

$$w := w + \Delta w \tag{2.8}$$

Onde a mudança de peso Δw é definida como o gradiente negativo multiplicado pela taxa de aprendizagem η:

$$\Delta w = -\eta \Delta J(w) \tag{2.9}$$

Para calcular o gradiente da função custo, é preciso calcular a derivada da função custo em relação a cada peso wj. As redes utilizam um algoritmo chamado *Backpropagation* que otimiza o processo de calcular as derivadas fazendo a rede ser mais eficiente.

Apesar do GD ser um bom algoritmo, ele encontra dificuldade de atualizar os pesos quando o conjunto de dados é muito grande, pois sua abordagem é conhecida como descida gradiente em lote, essa abordagem pode ser computacionalmente cara, pois é preciso reavaliar todo o conjunto de dados de treinamento para cada um passo em direção ao mínimo global (RASCHKA; MIRJALILI, 2017).

Uma alternativa popular ao GD é o Gradiente Descendente estocástico (do inglês *Stochastic Gradient Descent*-SGD), também chamado de gradiente iterativo ou *online*. Ao invés de atualizar os pesos com base na soma dos erros acumulados sobre todos os dados, ele aplica GD a amostras aleatórias de dados de treinamento. As vantagens em relação à descida do gradiente em lote é que a convergência da rede é alcançada mais rapidamente através de mini-lotes por causa das atualizações de peso mais frequentes, ou seja, o treinamento de uma rede com muitos dados será bem mais rápido com o SGD (RASCHKA; MIRJALILI, 2017).

O SGD também, tem alguns parâmetros que podem ajudar no treinamento da rede, são eles:

- **Taxa de aprendizagem**: quão rápido o otimizador tentará treinar a rede neural. Muito alto não conseguirá treinar. Muito baixo irá treinar muito devagar.
- ***Momentum***: Quanto a direção de mudança de peso anterior deve ser usada na etapa atual.
- ***Decay***: É utilizado para diminuir a taxa de aprendizagem conforme os erros diminuem.

Outro algoritmo, de otimização de rede é o *Root Mean Square Propagation* (RMS-prop). Ele utiliza o método taxa de aprendizagem adaptável. Ele parte da premissa que a

magnitude do gradiente pode ser muito diferente para diferentes pesos e pode mudar durante a fase de treinamento, isso dificulta a escolha de um taxa de aprendizado global única. Para aprendizado em lote completo, pode-se lidar com esta variação usando apenas o sinal do gradiente, entretanto, para aprendizagem *online* o RMSprop é mais indicado, pois as atualizações de peso são todas mesma magnitude. Ele combina a ideia de usar o sinal do gradiente com a ideia de adaptar o tamanho do passo separadamente para cada peso Hinton *et al.* (2012).

Neste trabalho, foram utilizados os otimizadores SGD com todos os parâmetro citados acima, e o RMSprop com o objetivo de descobrir qual dos dois resolve melhor a classificação dos personagens dos *Simpsons*.

2.7 Regularização

Um dos problemas em *Machine Learning* é como fazer um algoritmo ter bom desempenho não apenas no conjunto de dados de treino, mas também com novos dados. Muitas estratégias em *Machine Learning* são explicitamente desenhadas para reduzir o erro na fase de testes, permitindo uma taxa de erro maior no treino. Essas estratégias são conhecidas como regularização, elas também são aplicadas em *Deep Learning*.

Uma técnica comumente utilizada é a regularização L2. A ideia da regularização de L2 é adicionar um termo extra à função custo, ou seja, levando em consideração a função de *log loss*, é adicionado o termo L2:

$$-\sum_{c=1}^{M} y_{o,c} \log p_{o,c} + \frac{\lambda}{2n} \sum_{w} w^2 \tag{2.10}$$

a L2 é a soma dos quadrados de todos os pesos na rede, dimensionado por um fator $\lambda/2n$, onde $\lambda > 0$ é conhecido como o parâmetro de regularização, e n o tamanho do conjunto de treinamento. O objetivo da regularização é fazer com que a rede prefira aprender pequenos pesos, sendo todas as outras coisas iguais. Pesos grandes só serão permitidos se melhorarem consideravelmente a primeira parte da função de custo (NIELSEN, 2015).

Outra técnica de regularização que é popularmente usada em *Deep Learning* é o *droput*. O *dropout* é uma técnica de regularização criado por Geofrey Hinton em 2012, que ajuda a mudar a saída dos neurônios de uma rede neural profunda, e que pode ser aplicado em qualquer camada das redes neurais profundas. Essa técnica, desativa alguns dos neurônios da camada associada com alguma probabilidade p. Desativar um neurônio significa mudar o valor de saída

para 0. No final, os neurônios que sofreram *dropout* têm os parâmetros reajustados para que seja possível encontrar os melhores pesos (NIELSEN, 2015).

3 TRABALHOS RELACIONADOS

A seguir, são apresentados os principais trabalhos relacionados.

3.1 IMAGENET CLASSIFICATION WITH DEEP CONVOLUTIONAL NEURAL NETWORKS

Krizhevsky *et al.* (2012) treinaram uma rede neural convolucional profunda para classificar os 14,2 milhões imagens de alta resolução no ImageNet da competição LSVRC-2010 para 1000 classes diferentes. Nos dados do teste, foram alcançadas taxas de erro top-1 e top-5 de 37,5% e 17,0%, respectivamente, que foi considerado o melhor estado-da-arte em problemas de classificação de imagens até aquele ano.

Eles fizeram uma rede neural que possui 60 milhões de parâmetros e 650 mil neurônios com cinco camadas convolucionais, algumas das quais são seguidas por camadas de *Max-Pooling*, e três camadas totalmente conectadas com um *softmax* final de 1000 vias.

Os autores utilizam GPUs para conseguir o resultado um resultado de processamento mais rápido. Na operação de convolução para reduzir o *overfitting* na camada totalmente conectados eles usaram o método de *dropout* que tornou a rede mais eficaz.

Os trabalhos se assemelham, pois ambos utilizam redes convolucionais e diferem-se porque os autores aplicam sua solução para a base de dados do *ImageNet*, enquanto este trabalho faz comparativo entre as redes VGG16, ResNet e *Inception* no problema de classificação dos personagens dos *Simpsons*.

3.2 RECONHECIMENTO DE TUMORES CEREBRAIS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS

CRESPO e Hidalga (2017), implementaram uma CNN capaz de pré-classificar imagens com a presença de tumor cerebral e separá-las das imagens sadias, para uma detecção primária da doença. Eles identificaram a cobertura e a precisão obtidas pelo algoritmo, tanto nas imagens sadias, quanto nas imagens com presença de algum fator patogênico. Para saber a capacidade de generalização da rede ele utilizou o método de validação cruzada.

CRESPO e Hidalga (2017), os autores propõem um algoritmo com 3 camadas convolucionais todas elas com função de ativação sendo ReLU. A primeira camada convolucional possui 32 neurônios e utiliza uma janela de convolução de 2D 3 X 3, que recebe uma imagem

com dimensões de 512 X 512 com 3 canais. Vinculada a essa camada, tem-se 2 camadas de Max-Pooling 2 X 2. A segunda e terceira camadas convolucionais são semelhantes à primeira, diferindo apenas no número, pois são utilizados 16 e 8 neurônios, respectivamente.

O algoritmo proposto por CRESPO e Hidalga (2017), obteve um índice de 87% de Recall para imagens com a presença da doença e de 66% para imagens sadias, apresentando, assim, resultados satisfatórios para uma pré-detecção da doença. A rede em questão ainda obteve cerca de 94% de precisão para classificar imagens sadias e de 62% para imagens com a presença da doença.

Neste trabalho faremos um comparativo entre as redes convolucionais VGG16, ResNet e a *Inception* no problema de classificação de personagens dos *Simpsons*. Os dois trabalhos se assemelham pois as redes tentaram descobrir a qual classe pertencem as imagens dado um conjunto de classes.

3.3 LARGE-SCALE VIDEO CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS

Villegas *et al.* (2017), fizeram uma avaliação empírica de CNNs em larga escala para classificação de vídeo usando um conjunto de dados de 1,13 milhões de vídeos do YouTube pertencentes a 487 classes. No trabalho eles estudaram o desempenho de CNNs para classificação dos vídeos, onde as redes tiveram acesso a informações estáticas presentes nas imagens e acesso a sua evolução temporal.

Tendo em vista a dificuldade de treinar uma CNN, pois do ponto de vista computacional, uma CNN exige um tempo relativamente grande para ser treinada de modo que consiga otimizar seus parâmetros do seu modelo, e esta dificuldade é ainda maior quando se estende a conectividade da arquitetura no tempo porque a rede deve processar não apenas uma imagem, mas vários quadros de vídeo de uma vez, os autores mostram uma abordagem que para acelerar o desempenho em tempo de execução de um CNN, eles modificam a arquitetura para conter dois fluxos de processamento: um fluxo de contexto que aprende recursos em quadros de baixa resolução e um fluxo para alta resolução que opera somente na parte intermediária do quadro. Com isso, foi observado um aumento de 2-4x no desempenho em tempo de execução da rede devido à reduzida dimensionalidade da entrada, mas mantendo a precisão da classificação.

A melhor rede criada apresentou uma melhora de resultado de 55.3% para 63.9% sobre algoritmos de linhas de base. Mas apenas uma leve melhora de 59.3% para 60.9% em

relação à modelos de single-frame. Os trabalho se assemelham, pois ambos usam redes neurais Convolucionais para classificação e diferem-se porque neste trabalho é feito classificação de imagens e não de vídeos.

3.4 DEEP LEARNING FACE REPRESENTATION FROM PREDICTING 10,000 CLASSES

O trabalho de Sun *et al.* (2014) propõe que se crie conjunto de representações de características de alto nível utilizando redes profundas para reconhecimento de faces.

Os autores inovaram em seu trabalho pois os algoritmos da época de verificação faces de melhor desempenho normalmente representavam faces com recursos de baixo nível utilizando modelos superficiais. Com o uso das redes profundas eles conseguiram uma solução eficaz para extrair características visuais de alto nível.

A base de dados usado nesse trabalho foi a LFW, que é uma um banco de dados de fotografias de rosto projetado para estudar o problema do reconhecimento facial que contém 87.628 imagens de 5436 celebridades, com aproximadamente 16 fotos por celebridade.

A rede dos autores alcançou um resultado de 97,45% acurácia na verificação de face em LFW usando apenas faces fracamente alinhadas, esse resultado quase tão bom quanto o desempenho humano de 97,53%. Também foi observado que à medida que o número de identidades de treinamento aumentava, o desempenho da verificação é constantemente aprimorado, pois a capacidade de discriminação e generalização dos recursos aprendidos aumentam, entretanto, a tarefa a tarefa de treinamento se torna mais complexa.

A Tabela 2 mostra um comparativo entre os os trabalhos relacionados e este trabalho.

Tabela 2 – **Comparação entre os trabalhos relacionados e o proposto**

Trabalho	Tipo de redes	Tipo do problema	Dados	Problema
CRESPO e Hidalga (2017)	Implementação própria	Classificação binária	Imagens	Reconhecimento de tumores cerebrais com CNN
Villegas *et al.* (2017)	Implementação própria	Classificação multiclasse	Vídeos	Classificação de vídeo CNN
Krizhevsky *et al.* (2012)	Implementação própria	Classificação multiclasse	Imagens	Classificação de milhões de imagens
Trabalho Proposto	Redes pré-treinadas VGG16, ResNet50, Inception	Classificação multiclasse	Imagens	Classificação dos personagens dos Simpsons

Fonte: Informado pelo autor

4 METODOLOGIA

Nesta Capítulo, são apresentados os passos necessários para a execução do trabalho.

4.1 Escolha do Conjunto de Dados

Dado que o objetivo deste trabalho que é utilizar Redes Neurais Convolucionais para classificação de imagens, fez-se necessário buscar um conjunto de imagens que contenham imagens com classes diversificadas e diferente das imagens que estão presentes no conjunto do *ImageNet* para demonstrar o poder de generalização das CNNs.

4.2 Pré-processamento dos dados

Depois que as Redes Neurais Convolucionais são selecionadas, é necessário fazer o pré-processamento das imagens que serão utilizadas. Nesta etapa é feita a leitura, redimensionamento e processo de aumento do conjunto de dados utilizando a biblioteca keras [1]. Também é feita a divisão das imagens para treino, validação e teste.

4.3 Seleção das Redes Neurais Convolucionais

Para classificação de imagens existem diversas Redes Neurais Convolucionais que são frutos da competição *ImageNet*. A cada ano é escolhido uma nova campeã que consegue obter desempenhos surpreendentes em bases dados tão diversas e com mais de mil classes, desta forma, resta adaptar essas redes para resolução do problema proposto neste trabalho.

4.4 Implementação das Redes Neurais Convolucionais

Depois da escolha das Redes Neurais Convolucionais, são feitas as implementações de cada rede utilizando linguagem de programação *Python* e a biblioteca de aprendizado profundo *Keras* que permite a implementação em alto nível, de Redes Neurais. A escolha do *Keras* se deu por ela ser API de alto nível para redes neurais, escrita em *Python*. Foi desenvolvida com foco em permitir a experimentação rápida. Ela permite a criação de protótipos de forma fácil e rápida por meio de modularidade e extensibilidade, suporta redes convolucionais e funciona perfeitamente na CPU e GPU (*Graphic Process Unit*) (CHOLLET *et al.*, 2015).

[1] https://keras.io/

4.5 Treinamento das Redes Neurais Convolucionais

Depois da etapa de implementação das redes, é feito o treinamento. Primeiro foram carregados os pesos de cada rede e em seguida foram escolhidos os parâmetros de treinamento como número de épocas e o tamanho dos lotes (do inglês *batch size*) para fazer o treinamento.

4.6 Análise de Desempenho das Redes Neurais Convolucionais Implementadas

Após o treinamento das redes escolhidas, são analisadas as métricas *recall*, acurácia, precisão e *log loss* de cada rede. Essas métricas são as mais utilizadas em problemas de classificação. A rede que teve menor *log loss* no problema de classificação dos personagens do *Simpsons* foi a escolhida.

5 RESULTADOS

Neste Capítulo são apresentados os resultados dos experimentos realizados neste trabalho.

5.1 Escolha do Conjunto de Dados

O conjunto de dados utilizado neste trabalho foi disponibilizado em um desafio da plataforma *Kaggle* criado por um membro da plataforma que é fã dos *Simpsons*. O desafio é basicamente classificar, dada uma imagem, qual é o personagem que está na imagem. Para isso, é disponibilizado um conjunto de dados com imagens 2D. Este trabalho usa apenas os dez primeiros personagens que são os que têm maior o número de imagens. O conjunto de dados está dividido da seguinte forma: 11,054 imagens para treino, 2,757 imagens para validação e 456 imagens para teste.

O conjunto de dados possui imagens de 10 personagens, em termos técnicos, cada personagem representa uma classe de dados, são eles: *Homer Simpson*, *Moe Szyslak*, *Lisa Simpson*, *Bart Simpson*, *Marge Simpson*, *Krusty Clow*, *Principal skinner*, *Charles Montgomery Burns*, *Milhouse Van Houten*, *Ned Flanders*.

Cada classe possui um número aproximadamente igual de imagens, fazendo com que o treinamento não beneficie uma determinada classe. A Tabela 3 mostra a quantidade de imagens para cada classe.

Tabela 3 – Quantidade de imagens para cada classe.

Classe	**Quantidade de imagens**
Homer Simpson	2,246
Ned Flanders	1,454
Moe Szyslak	1,452
Lisa Simpson	1,354
Bart Simpson	1,342
Marge Simpson	1,291
Krusty The Clown	1,206
Principal Skinner	1,194
Charles Montgomery Burns	1,193
Milhouse Van Houten	1,079

Fonte: Próprio autor

5.2 Pré-Processamento dos Dados

A etapa de pré-processamento dos dados é importante, pois carrega os dados dos diretórios, os normalizam em uma escala que se torna mais fácil para a computação, nela são redimensionadas todas as imagens para a mesma dimensão de 75 X 75, uma vez que, as imagens apresentam dimensões diferentes e as redes exigem dimensões de no mínimo 75 X 75. Também nessa etapa, é feito o aumento de dados (do inglês *data augmentation*) que é um técnica que faz modificações nas imagens a cada iteração durante o treinamento com o objetivo de fazer as redes não conhecerem os dados de treinamento completamente, para que assim, as redes não percam sua capacidade de generalização se tornando especialista apenas nos dados de treinamento. A Figura 8 a seguir, mostra um exemplo de uma imagem que passou pelo processo de *augmentation*.

Figura 8 – Aumento dos dados

(a) Original (b) Modificação 01 (c) Modificação 02

Fonte: Imagem retirada do conjunto de dados

É possível perceber na Figura 8 que a imagem do *Homer Simpson* teve modificações de espelhamento e rotações. Esse processo é repetido para cada imagem do conjunto de treinamento e validação adicionando novas modificações. Todo esse processo, foi feito com a função de pré-processamento que pertence a biblioteca *Keras*.

5.3 Seleção das Redes Neurais Convolucionais

O critério de escolha das redes, foi baseado nas redes que tiveram os melhores resultados na competição do *ImageNet*, visto que, essas redes foram criadas pelos principais grupos de estudos de redes neurais do mundo, configurando-se o estado da arte na área. Outro fator, não menos importante, é que elas estão disponíveis publicamente e são necessárias poucas

modificações para adaptá-las para outros problemas. A biblioteca *Keras* tem todas essas redes internamente, o que facilita mais ainda o trabalho na hora de usá-las.

Desta forma, foram escolhidas as redes VGG-16 que foram proposta por Simonyan e Zisserman (2014), que obtiveram os melhores resultados no ano de 2014. O Google propôs a rede *Inception*, que conseguiu alcançar um desempenho muito bom e relativamente de baixo custo computacional que rendeu um desempenho de última geração na edição da competição de 2015 (SZEGEDY *et al.*, 2017). E no mesmo ano, He *et al.* (2016) foram os campeões com a proposta da ResNet, esta solução apresentou o melhor desempenho no conjunto de critérios da competição.

5.4 Implementação das Redes Neurais Convolucionais

Depois de feita a seleção das redes, foi feita a implementação de cada rede, para isso foi usada a linguagem de programação *Python* com a biblioteca *Keras*.

A implementação das redes foram feitas em *Kernels* da plataforma *Kaggle*, pois a plataforma oferece um ambiente para desenvolvimento virtual com GPU e 14 *Ggiga bytes* de memória *RAM*, o que facilita o treinamento, outro fator, é que os dados já se encontra na plataforma então facilita na importação e utilização.

No processo de implementação, é preciso adaptar as redes para o problema que se busca resolver, pois as redes pré-treinadas foram feitas para resolver o problema do *ImageNet*, ou seja, todas possuem saídas do tamanho da quantidade de classes do *Imagenet*, que são 1000 classes. Porém, o conjunto de dados utilizado neste trabalho possui apenas 10 classes.

Para resolver esse problema, as redes são baixadas com ausência do último bloco de camadas, pois é neste bloco de camadas que acontece o processo de classificação. Logo, foi preciso criar uma camada adaptada para a classificação de 10 classes. Para isso, foi adicionada uma camada *flatten* para reduzir a dimensionalidade dos dados para dimensão 1D, uma camada densa com ativação *RElu* e uma camada densa contendo 10 unidades representando o número de classes. Por fim, a rede é compilada com a nova configuração.

5.5 Treinamento das Redes Neurais Convolucionais

Por fim, é feita a etapa de treinamento. Nesta etapa, são passados como parâmetro a rede o número de épocas. Uma época é uma iteração passando por todos os dados do conjunto

de treino. O *Batch size* que é o número de dados utilizados para atualização dos pesos e os dados treino e validação. Feito isso, são coletados os resultados por meio da função *history* do *Keras*, e em seguida são as feitas as predições para o conjuntos de teste, coletando todas as métricas para serem analisadas.

5.6 Resultados da Análise de Desempenho das Redes Neurais Convolucionais Implementadas

Depois do treinamento é feita a análise de desempenho. Essa etapa se faz necessária, pois é nela que é possível avaliar qual rede consegue os melhores resultados para o problema da classificação dos personagens dos *Simpsons* que foi o conjunto de dados escolhidos neste trabalho.

Para avaliar quão bons foram os resultados obtidos pelas Redes Neurais, foram avaliadas as métricas descritas no Capítulo 2, Sendo que, a rede que alcançar o menor valor de *log loss* será considerada a rede com o melhor desempenho para este problema. Entretanto, para outros problemas de classificação as demais métricas podem ser mais relevantes. Diante disso, são utilizadas a Matriz de Confusão, *precision* e *recall* com um objetivo de ter uma visão mais ampla da melhor resolução para o problema. Para fazer uma análise mais aprofundada foram utilizados os otimizadores de rede RMSprop e o SGD e a utilização de técnica de regularização.

A seguir, na Tabela 4, é possível ver os valores de *log loss* obtidos em cada época para cada rede convolucional quando usado o otimizador RMSprop. É possível verificar que as redes com melhores valores de *log loss* foram VGG-16 e *Inception*. A última coluna que está em negrito representam os melhores valores na décima época.

Tabela 4 – Valores de *log loss* por época (epc) por rede com RMSprop

Rede	epc1	epc2	epc3	epc4	epc5	epc6	epc7	epc8	epc9	ep10
VGG16	2.798	15.07	4.85	13.33	3.35	1.72	1.63	3.92	1.42	**1.35**
Inception	16.11	13.21	16.05	13.35	15.87	12.72	15.76	14.10	15.62	**13.39**
ResNet50	12.56	14.91	10.81	16.11	12.71	14.78	11.52	15.70	11.34	**15.64**

Fonte: Próprio autor

A Tabela 4 mostra ainda valores de *log loss* distantes de zero, mesmo para a Rede VGG-16 que obteve um valor final de 1.35. As demais redes estão com valores de *log loss* mais distantes de zero, o que demonstra que essa configuração não está conseguindo convergir para bons valores com 10 épocas.

A seguir, na Tabela 4, é possível ver os valores de acurácia de validação em cada época por rede com RMSprop, onde é possível verificar que as redes com maiores valores de acurácia foram VGG-16 e *Inception*, com os valores de acurácia de 58.9% e 16.5% ao final das 10 épocas, o que demonstra que ainda há o que melhorar nestas redes.

Tabela 5 – Acurácia de cada rede com RMSprop

Rede	epc1	epc2	epc3	epc4	epc5	epc6	epc7	epc8	epc9	ep10
VGG16	0.157	0.064	0.153	0.129	0.072	0.505	0.472	0.288	0.526	**0.589**
Inception	0.000	0.180	0.003	0.171	0.015	0.210	0.022	0.125	0.030	**0.165**
ResNet50	0.220	0.074	0.328	0.000	0.210	0.082	0.284	0.025	0.296	**0.029**

Fonte: Próprio autor

Este resultado ruim no treinamento com RMSprop também é constatado no conjunto de teste, pois também foram coletadas as matrizes de confusão de cada rede. As células da matriz são de cor azul e conforme o número de ocorrências de valores nela é maior, o tom de azul fica mais escuro, isso favorece uma melhor visualização. As redes com melhores desempenho tem a diagonal principal da matriz com a cor de azul mais escura, mostrando que o valor predito da rede é o mesmo valor real da classe. As Figuras 9 mostram a matriz de confusão para as redes VGG16, *Inception* e ResNet, respectivamente. É possível perceber que somente a VGG16 conseguiu resultados razoáveis e as demais redes erraram praticamente tudo, isso é constado olhando a tonalidade do azul na diagonal principal.

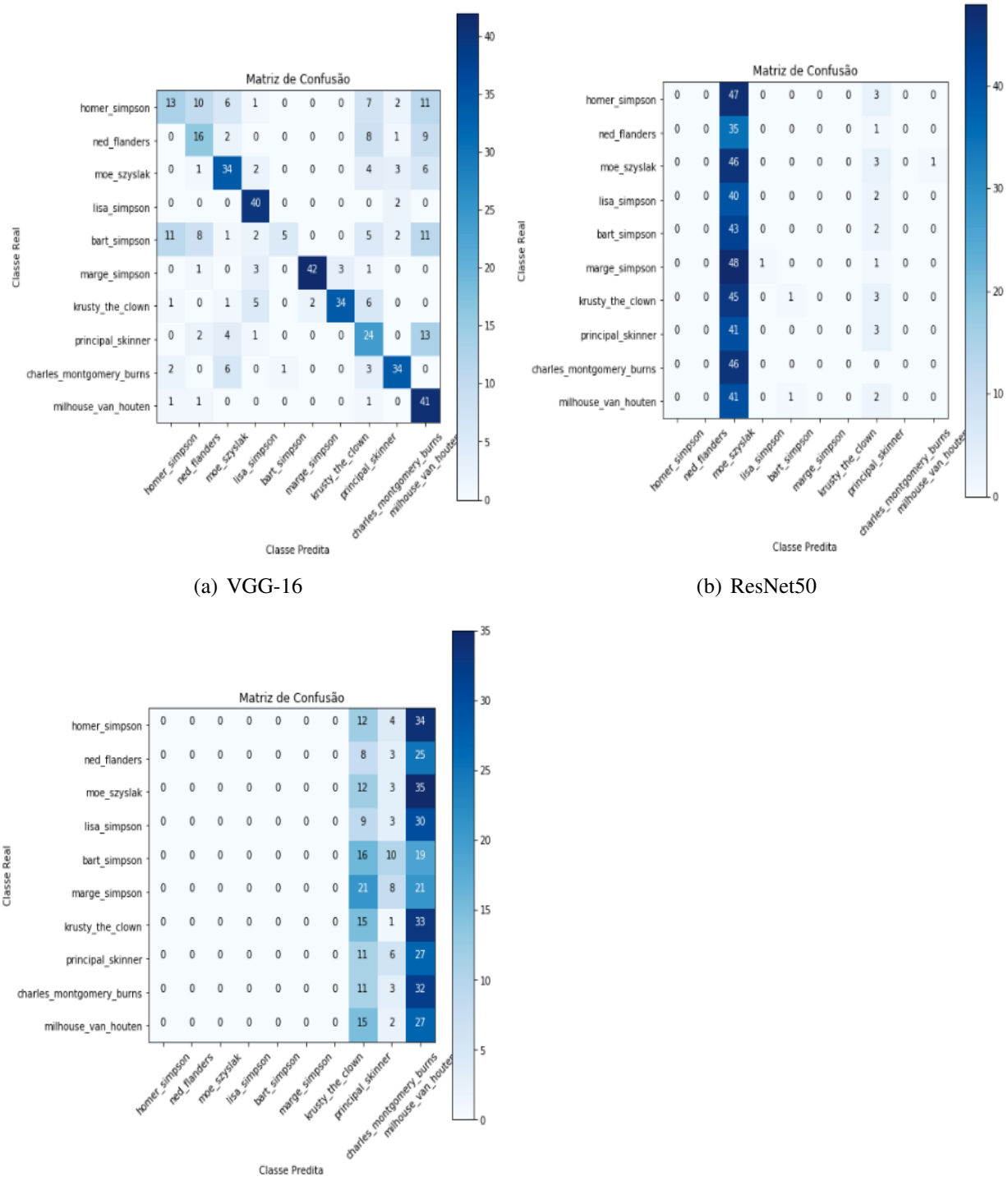
Para avaliar as métricas *precision* e *recall*, foi utilizado a função de *Classification Report* da biblioteca *Scikit-learn* [1]. Ela mostra uma visão mais ampla dos resultados das métricas para as redes. A partir da Figura 10 até a Figura 12, é possível perceber que com exceção da VGG-16, as demais Redes tiveram resultados muito ruins para o conjunto de teste.

As métricas da *Inception* e da ResNet50 estão bem próximas e muito baixas. O fato do RMSprop utilizar taxas de aprendizagem adaptativas faz com que as redes mais profundas tenham mais dificuldade de convergir para os melhores pesos. Seria necessário corrigir isso aumentando o número de épocas, adicionando regularização, *dropout*, entre outras estratégias.

A fim de encontrar melhores resultados, foi utilizado o otimizador SGD em todas as redes. O processo feito para as redes com RMSprop foi repetido para SGD. Na Tabela 6, é possível ver os valores de *log loss* obtidos em cada época para cada rede convolucional, onde é possível verificar que as redes com melhores valores de *loss* foram *Inception* e VGG-16. Na última coluna é marcado com asterisco o menor valor de *loss* na décima época.

[1] https://scikit-learn.org/stable/

Figura 9 – Matriz de Confusão de cada Rede



(a) VGG-16

(b) ResNet50

(c) *Inception*

Fonte: Próprio autor

Também é possível notar na Tabela 6 que os valores estão decaindo e possivelmente, com um maior número de épocas, esses valores convergiriam a um número próximo de zero. A mudança do otimizador foi uma boa escolha, visto que, com o SGD já se conseguiu valores de *loss* menores que 1, o que demonstra que as redes agora estão errando bem menos.

Figura 10 – *Classification Report* da VGG16

	precision	recall	f1-score	support
homer_simpson	0.46	0.26	0.33	50
ned_flanders	0.41	0.44	0.43	36
moe_szyslak	0.63	0.68	0.65	50
lisa_simpson	0.74	0.95	0.83	42
bart_simpson	0.83	0.11	0.20	45
marge_simpson	0.95	0.84	0.89	50
krusty_the_clown	0.92	0.69	0.79	49
principal_skinner	0.41	0.55	0.47	44
charles_montgomery_burns	0.77	0.74	0.76	46
milhouse_van_houten	0.45	0.93	0.61	44
avg / total	0.67	0.62	0.60	456

Fonte: Próprio autor

Figura 11 – *Classification Report* da *Inception*

	precision	recall	f1-score	support
homer_simpson	0.00	0.00	0.00	50
ned_flanders	0.00	0.00	0.00	36
moe_szyslak	0.00	0.00	0.00	50
lisa_simpson	0.00	0.00	0.00	42
bart_simpson	0.00	0.00	0.00	45
marge_simpson	0.00	0.00	0.00	50
krusty_the_clown	0.00	0.00	0.00	49
principal_skinner	0.08	0.25	0.13	44
charles_montgomery_burns	0.07	0.07	0.07	46
milhouse_van_houten	0.10	0.61	0.17	44
avg / total	0.02	0.09	0.03	456

Fonte: Próprio autor

Figura 12 – *Classification Report* da ResNet

	precision	recall	f1-score	support
homer_simpson	0.00	0.00	0.00	50
ned_flanders	0.00	0.00	0.00	36
moe_szyslak	0.11	0.92	0.19	50
lisa_simpson	0.00	0.00	0.00	42
bart_simpson	0.00	0.00	0.00	45
marge_simpson	0.00	0.00	0.00	50
krusty_the_clown	0.00	0.00	0.00	49
principal_skinner	0.15	0.07	0.09	44
charles_montgomery_burns	0.00	0.00	0.00	46
milhouse_van_houten	0.00	0.00	0.00	44
avg / total	0.03	0.11	0.03	456

Fonte: Próprio autor

Tabela 6 – Valores de loss por época (epc) por Rede

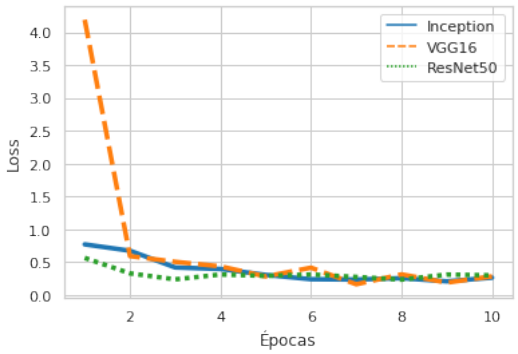
Rede	**epc1**	**epc2**	**epc3**	**epc4**	**epc5**	**epc6**	**epc7**	**epc8**	**epc9**	**ep10**
Inception	0.769	0.672	0.417	0.394	0.304	0.238	0.233	0.254	0.206	**0.262***
VGG16	4.190	0.590	0.503	0.434	0.280	0.414	0.161	0.312	0.187	**0.281**
ResNet50	0.563	0.327	0.236	0.310	0.289	0.310	0.273	0.232	0.309	**0.297**

Fonte: Próprio autor

A Figura 13 mostra uma visão geral dos desempenhos de cada rede por época, em relação ao *log loss*. Apesar das redes terem valores semelhantes, é possível perceber que a *Inception* e a VGG-16 obtiveram os menores valores de *log loss* e que a partir da época 6 o *log*

loss da *Inception* e ResNet50 apresentam comportamentos quase lineares. A VGG-16 foi a que teve uma partida de *log loss* mais alta, entretanto, conseguiu acompanhar as demais redes a partir da segunda época.

Figura 13 – Visão Geral do *log loss* de cada rede por época



Fonte: Próprio autor

A seguir, na Tabela 7, é possível ver os valores de acurácia em cada época por rede, onde é possível verificar que as redes com maiores valores de acurácia foram ResNet50 e a *Inception*, com os valores de acurácia de 93,4% e 92,9%. Uma mudança significativamente grande em relação as redes treinadas com RMSprop.

Na Figura 14 é possível analisar graficamente a evolução das acurácias de cada Rede por época. Na figura é possível perceber que a VGG-16 começou com um valor de acurácia bem menor de que as demais redes, mas que a partir da época 2 ela conseguiu o mesmo comportamento das demais redes.

Tabela 7 – Valores de acurácia em porcentagem por época (epc) por rede

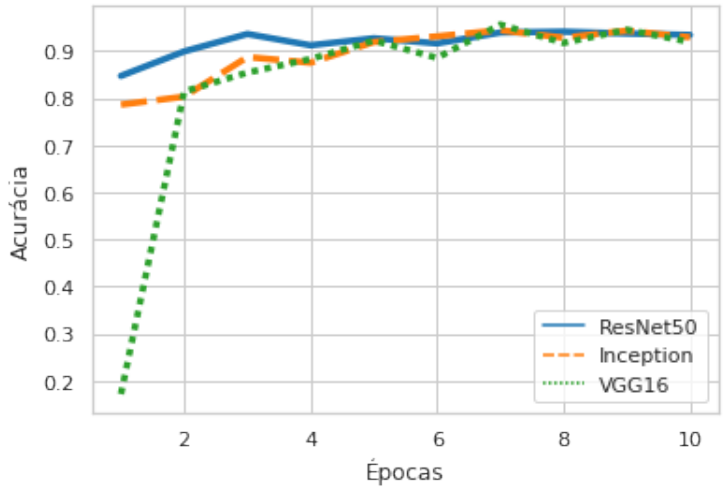
Rede	**epc1**	**epc2**	**epc3**	**epc4**	**epc5**	**epc6**	**epc7**	**epc8**	**epc9**	**ep10**
ResNet50	0.847	0.899	0.936	0.912	0.927	0.916	0.939	0.941	0.937	**0.934***
Inception	0.786	0.803	0.887	0.875	0.919	0.931	0.944	0.928	0.943	**0.929**
VGG16	0.1715	0.814	0.854	0.883	0.922	0.8852	0.956	0.917	0.945	**0.919**

Fonte: Próprio autor

Apesar das redes ResNet50 e *Inception* terem sido melhores, é possível perceber que elas estão bem semelhantes e que é preciso de um treinamento maior, ou seja, um maior número de épocas para que seja possível distinguí-las melhor.

Também foram coletadas as matrizes de confusão. A Figura 15 mostra a matriz de confusão para cada rede utilizando o SGD. Nela é possível observar que em todas as redes, as diagonais principais apresentam números mais altos, isso representa que a rede acertou bastante,

Figura 14 – Visão Geral da acurácia de cada rede por época



Fonte: Próprio autor

visto que, estas diagonais representam as predições corretas feita pela a Rede. Também é possível ver que as redes VGG-16 e *Inception* foram as que mais erraram no conjunto de teste.

É possível observar que a VGG-16 na Figura 15 (a) que a rede errou 4 classificações quando a classe era *Marge Simpson*, já a rede *Inception* errou 5 classificações quando a classe era *Moe Szyslak* como é mostrado na Figura 15 (c), enquanto, a rede ResNet50 foi a melhor classificadora no conjunto de testes, repetindo o comportamento da fase de treinamento.

Por fim, na Figura 16, é possível ver para cada rede e para cada classe, os valores das métricas precisão, *recall*, além do *support*, que a soma da quantidade de dados para cada classe.

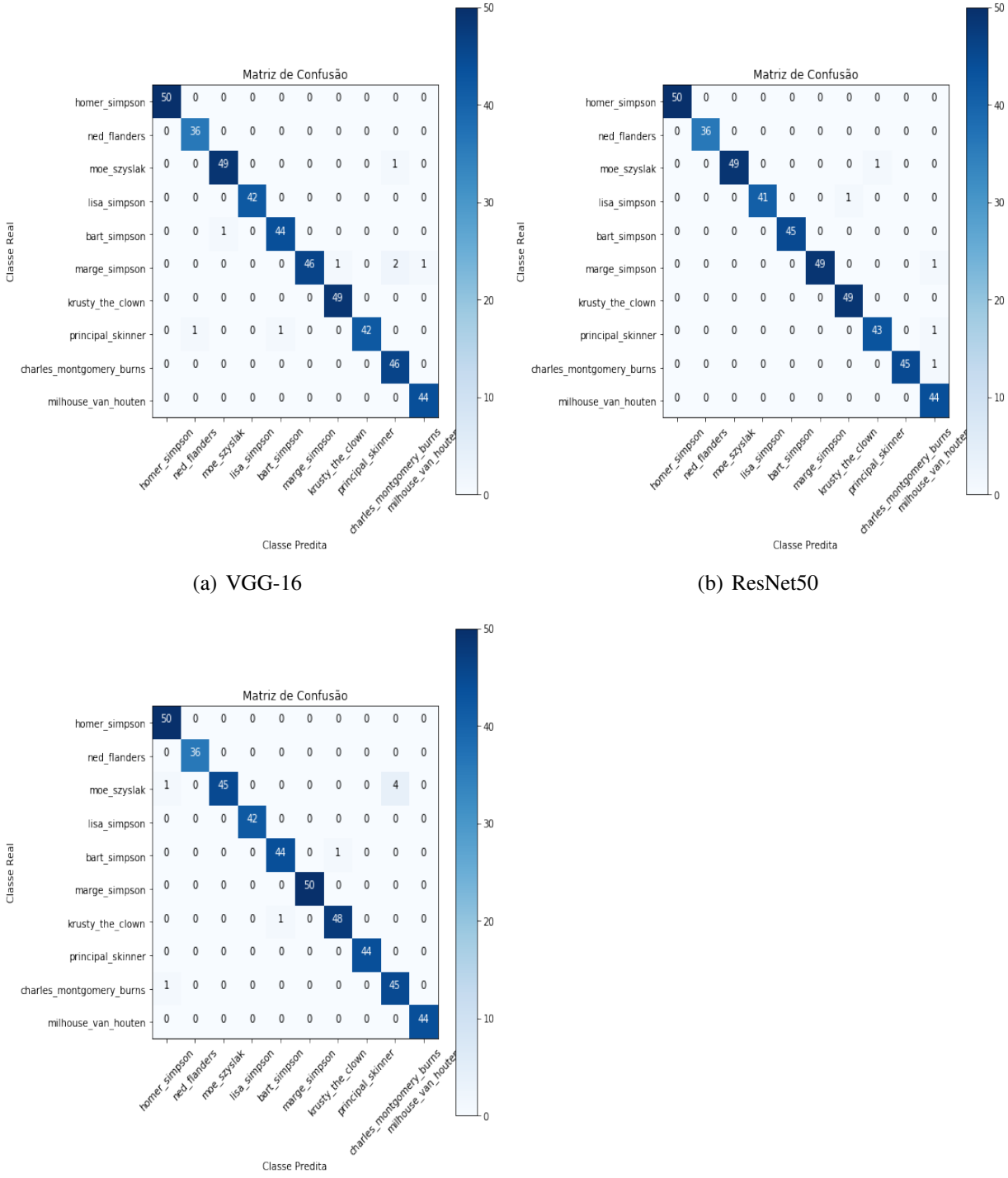
Pela Figura 16, é possível observar que no geral a VGG-16 e a *Inception* foram iguais com 98% para cada métrica. Enquanto a ResNet50 obteve um resultado de 99% para cada métrica. Uma evolução muito grande comparadado as Redes com RMSprop.

Apesar de no geral as redes com SGD terem conseguido resolver de forma satisfatória o problema, se constatou que a rede com maior número de camadas convolucionais foi melhor na acurácia. Isso nem sempre é uma regra, mas é possível constatar que quanto maior o número de camadas convolucionais, mais características a rede consegue extrair dos dados, o que faz ela aprender mais com os dados. Outro fator que pode ser testado é aumentar o número de épocas e conseguir melhorar os resultados de todas as redes.

Para conseguir observar o efeito do aumento do número das épocas junto com a técnicas de *Dropout* e regularização, foram re-treinadas as redes que utilizaram RMSprop para verificar se essas técnicas conseguem melhorar o desempenho das redes.

A Tabela 6 mostra o os resultados de acurácia das redes com o otimizador RMSprop

Figura 15 – Matriz de Confusão de cada Rede



(a) VGG-16

(b) ResNet50

(c) *Inception*

Fonte: Próprio autor

com 10 épocas sem *dropout* e sem regularização, e na terceira coluna mostra o resultado que as redes obtiveram com 100 épocas utilizando *dropout* e regularização. Para simplificar a visualização está sendo mostrado só os últimos valores obtidos.

É possível verificar na Tabela 8 uma melhoria significativa para as redes *Inception* e

Figura 16 – *Classification Report* de cada Rede

	precision	recall	f1-score	support
homer_simpson	1.00	1.00	1.00	50
ned_flanders	0.97	1.00	0.99	36
moe_szyslak	0.98	0.98	0.98	50
lisa_simpson	1.00	1.00	1.00	42
bart_simpson	0.98	0.98	0.98	45
marge_simpson	1.00	0.92	0.96	50
krusty_the_clown	0.98	1.00	0.99	49
principal_skinner	1.00	0.95	0.98	44
charles_montgomery_burns	0.94	1.00	0.97	46
milhouse_van_houten	0.98	1.00	0.99	44
avg / total	0.98	0.98	0.98	456

(a) VGG-16

	precision	recall	f1-score	support
homer_simpson	1.00	1.00	1.00	50
ned_flanders	1.00	1.00	1.00	36
moe_szyslak	1.00	0.98	0.99	50
lisa_simpson	1.00	0.98	0.99	42
bart_simpson	1.00	1.00	1.00	45
marge_simpson	1.00	0.98	0.99	50
krusty_the_clown	0.98	1.00	0.99	49
principal_skinner	0.98	0.98	0.98	44
charles_montgomery_burns	1.00	0.98	0.99	46
milhouse_van_houten	0.94	1.00	0.97	44
avg / total	0.99	0.99	0.99	456

(b) ResNet50

	precision	recall	f1-score	support
homer_simpson	0.96	1.00	0.98	50
ned_flanders	1.00	1.00	1.00	36
moe_szyslak	1.00	0.90	0.95	50
lisa_simpson	1.00	1.00	1.00	42
bart_simpson	0.98	0.98	0.98	45
marge_simpson	1.00	1.00	1.00	50
krusty_the_clown	0.98	0.98	0.98	49
principal_skinner	1.00	1.00	1.00	44
charles_montgomery_burns	0.92	0.98	0.95	46
milhouse_van_houten	1.00	1.00	1.00	44
avg / total	0.98	0.98	0.98	456

(c) *Inception*

Fonte: Próprio autor

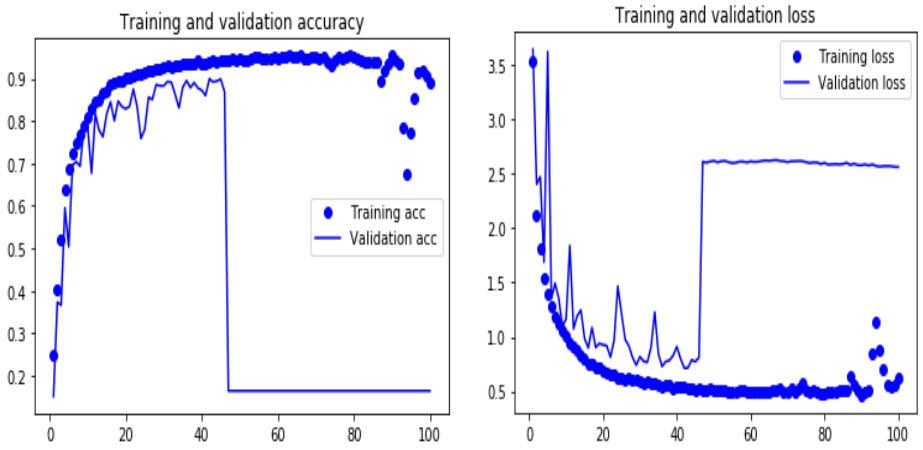
Tabela 8 – Comparativo de acurácia das redes com 10 e 100 épocas

Redes	**10 Épocas**	**100 Épocas**
VGG-16	0.589	0.163
Inception	0.165	0.896
ResNet50	0.029	0.921

Fonte: Próprio autor

ResNet50, entretanto, a VGG-16 piorou seus resultados, pois sofreu *overfitting*. O *overfitting* é identificado quando o erro de teste, obtido pela validação, começa a aumentar depois de ter diminuído. Para evitar o *overfitting*, a aprendizagem deve ser interrompida quando os erros não variarem mais. Na VGG-16 foi notado que o seu treinamento era pra ser interrompido na época 44, pois foi quando a rede conseguiu o resultado 89.46% de acurácia na validação, como é mostrado na Figura 17 a seguir.

Figura 17 – *Overfitting* na rede VGG16 com 100 épocas



(a) *Overfitting* no treinamento (b) *Overfitting* na validação

Fonte: Próprio autor

Por fim, a Tabela 9 mostra o resultado do *log loss* das redes com o otimizador RMSprop com 10 épocas sem utilizar a regularização e o *dropout* e com 100 épocas utilizando. Na Tabela 8 nota-se que os Valores de *log loss* da *Inception* e da ResNet50 caíram drasticamente demonstrando que as Redes melhoraram, enquanto a VGG-16 teve um aumento no valor confirmando *overfitting*. Se o treinamento para a VGG-16 fosse interrompido na época 44, o valor de *log loss* seria de 0.78, sendo assim, ela seria a melhor entre as demais redes neste cenário.

Tabela 9 – Comparativo de *log loss* das redes com 10 e 100 épocas

Redes	**10 Épocas**	**100 Épocas**
VGG-16	1.35	2.56
Inception	13.39	1.0504
ResNet50	15.64	2.8388

Fonte: Próprio autor

Embora, os resultados tenham sido melhorados com o aumento no número de épocas e o uso das técnicas de *Dropout* e regularização, as redes treinadas com SGD ainda conseguiram resolver de maneira mais eficiente e simples o problema, ou seja, a escolha de um otimizador é crucial para resolver um problema de classificação de imagens. Outro fator, não menos importante, foi que o treinamento com 100 épocas demorou mais de cinco horas para cada rede. Enquanto o uso do SGD permitiu alcançar bons resultados com apenas 10 épocas.

6 CONCLUSÃO E DISCUSSÕES

Este trabalho teve como objetivo fazer um estudo comparativo entre algoritmos de redes neurais convolucionais para classificação de imagens utilizando o problema de classificação dos personagens dos *Simpsons* do *Kaggle*. Foram utilizadas 3 redes neurais convolucionais diferentes: ResNet50, Inception e VGG-16. Para cada rede foram feitos experimentos alterando o número de épocas, utilizando técnicas de regularização e*dropout*. Também foram utilizados dois tipos de otimizadores de redes que foram o SGD e o RMSprop. Diante destes experimentos, foram constatados alguns problemas e possíveis soluções.

O principal problema constatado, foi que a escolha de um otimizador é crucial para a resolução de um problema de classificação utilizando aprendizado profundo. As redes ResNet, *Inception* e VGG-16 quando utilizaram o SGD como otimizador conseguiram resolver o problema de forma satisfatória com 10 épocas. Sendo que o melhor resultado de acurácia foi o da ResNet-50 com 93.4% e o melhor resultado de *log loss* foi o da Inception com 0.26. Entretanto, quando foi utilizado o RMSprop, o melhor resultado de acurácia e de *log loss* foram da VGG-16 com 58,9% e 1.35, respectivamente.

Foi buscada uma forma de resolver o problema dos resultados insatisfatórios quando as redes utilizaram o RMsprop. Foram propostas algumas técnicas como regularização, *dropout* e aumento no número de épocas para que fosse possível conseguir melhores resultados. Os resultados melhoraram significativamente e as redes chegaram a resolver o problema de forma satisfatória, com exceção da VGG-16 que foi treinada com 100 épocas e sofreu *overfitting*, o que piorou seus resultados. Já as redes *Incecption* e ResNet, quando foram treinadas com 10 épocas utilizando o otimizador RMsprop, tinha conseguindo valores de acurácia de 16.5% e 2.9%. Quando treinadas com 100 épocas, utilizando técnicas de regularização e *dropout* com o mesmo otimizador, os resultados de acurácia foram de 89.6% e 92.1%, respectivamente. Ou seja, conseguiriam resolver o problema de forma satisfatória, embora tivessem levado bem mais tempo de treinamento. Também tiveram uma mudança significativa no *log loss* partindo de 13.39, 15.64 para 1.05, 2.83, nessa ordem.

Este trabalho mostrou alguns contextos de aplicação de redes neurais convolucionais, visando contribuir com pesquisadores iniciantes na área. Todas as implementações estão públicas no Github [1] e podem ser utilizadas como estudo para melhores resultados neste problema ou podem ser adaptadas para outros, tendo todas as bibliotecas necessárias e o pré-processamento

[1] https://github.com/RodrigoValentim2/The-Simpsons-Characters

de imagens utilizando o *Keras*.

REFERÊNCIAS

BRITZ, D.; GOLDIE, A.; LUONG, T.; LE, Q. Massive exploration of neural machine translation architectures. **arXiv preprint arXiv:1703.03906**, 2017.

CHAN, W.; JAITLY, N.; LE, Q.; VINYALS, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In: IEEE. **Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on**. [S.l.], 2016. p. 4960–4964.

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

CHOLLET, F. *et al.* **Keras**. 2015. Disponível em: <https://keras.iol>. Acesso em: 22 jun. 2018.

CRESPO, N.; HIDALGA, S. A. de L. **Reconhecimento de tumores cerebrais utilizando redes neurais convolucionais**. Universidade Federal do Pampa, 2017.

CS231N. **CS231n Convolutional Neural Networks for Visual Recognition**. [S.l.], 2018. Disponível em: <http://cs231n.github.io/convolutional-networks/>. Acesso em: 05 maio. 2018.

ESTEVA, A.; KUPREL, B.; NOVOA, R. A.; KO, J.; SWETTER, S. M.; BLAU, H. M.; THRUN, S. Dermatologist-level classification of skin cancer with deep neural networks. **Nature**, Nature Publishing Group, v. 542, n. 7639, p. 115, 2017.

HACKELING, G. **Mastering Machine Learning with scikit-learn**. [S.l.]: Packt Publishing Ltd, 2014.

HAYKIN, S. S.; HAYKIN, S. S.; HAYKIN, S. S.; HAYKIN, S. S. **Neural networks and learning machines**. [S.l.]: Pearson Upper Saddle River, NJ, USA:, 2009. v. 3.

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.

HIJAZI, S.; KUMAR, R.; ROWEN, C. Using convolutional neural networks for image recognition. **Cadence Design Systems**, San Jose, CA, USA, 2015.

HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. **Cited on**, p. 14, 2012.

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.

NIELSEN, M. A. **Neural networks and deep learning**. [S.l.]: Determination press USA, 2015.

PEEMEN, M.; MESMAN, B.; CORPORAAL, H. Speed sign detection and recognition by convolutional neural networks. In: **Proceedings of the 8th International Automotive Congress**. [S.l.: s.n.], 2011. p. 162–170.

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

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

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.; IOFFE, S.; VANHOUCKE, V.; ALEMI, A. A. Inception-v4, inception-resnet and the impact of residual connections on learning. In: **AAAI**. [S.l.: s.n.], 2017. v. 4, p. 12.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. *et al.* Going deeper with convolutions. In: **CVPR**. [S.l.], 2015.

VILLEGAS, R.; YANG, J.; HONG, S.; LIN, X.; LEE, H. Decomposing motion and content for natural video sequence prediction. **arXiv preprint arXiv:1706.08033**, 2017.