Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Aplicação de Deep Learning para análise de sentimentos a partir de imagens

Autor: Jôberth Rogers Tavares Costa
Orientador: Prof. Dr. Glauco Vitor Pedrosa

Brasília, DF
2022

Jôberth Rogers Tavares Costa

Aplicação de Deep Learning para análise de sentimentos a partir de imagens

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Glauco Vitor Pedrosa

Brasília, DF

2022

Jôberth Rogers Tavares Costa
Aplicação de Deep Learning para análise de sentimentos a partir de imagens/ Jôberth Rogers Tavares Costa. – Brasília, DF, 2022-
54 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Glauco Vitor Pedrosa

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2022.

1. Deep Learning. 2. Processamento de Linguagem Natural. I. Prof. Dr. Glauco Vitor Pedrosa. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aplicação de Deep Learning para análise de sentimentos a partir de imagens

CDU 02:141:005.6

Jôberth Rogers Tavares Costa

Aplicação de Deep Learning para análise de sentimentos a partir de imagens

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 05 de maio de 2022 – Data da aprovação do trabalho:

Prof. Dr. Glauco Vitor Pedrosa
Orientador

Prof. Dr. Daniel Sundfeld Lima
Convidado 1

Profa. Dra. Rejane M. da Costa Figueiredo
Convidado 2

Brasília, DF
2022

Resumo

Com o crescente número de compartilhamentos de imagens nos últimos anos, o surgimento de pesquisas na área da inteligência artificial para o reconhecimento de conteúdos visuais também cresceu consideravelmente. Entre essas pesquisas destaca-se o uso do *Deep Learning* para o entendimento automatizado do sentimento compartilhado através das mídias digitais. Entretanto, predizer sentimentos é uma tarefa complexa, visto que, devido à subjetividade humana, conteúdos similares podem ser interpretados de maneiras diferentes. Este trabalho apresenta o desenvolvimento de duas abordagens diferentes envolvendo a área de *Deep Learning*, de modo a analisar os sentimentos expressos por imagens digitais considerando duas classes (Positiva e Negativa). Uma abordagem analisa diretamente o conteúdo visual da imagem, enquanto a outra faz uma interpretação textual do conteúdo da mídia para posteriormente, categorizar o sentimento da figura com base na legenda extraída. Ambas as abordagens tiveram resultados praticamente semelhantes, mas durante o processo de *benchmark*, o resultado favoreceu a abordagem visual com um valor de 63% de acurácia devido sua simplicidade em relação ao fluxo e a dificuldade da segunda abordagem em trazer sentimentos nas legenda durante o processo de descrição da figura. Para trabalhos futuros é sugerido o melhoramento das arquiteturas desenvolvidas até o momento e possivelmente a adição de novas arquiteturas de redes convolucionais para agregar ao *benchmark* coletado.

Palavras-chave: *deep learning*, análise de sentimentos em imagens, redes neurais, processamento de linguagem natural

Abstract

With the increasing number of image shares in recent years, the emergence of research in the field of artificial intelligence for the recognition of visual content has also grown considerably. Among these researches, the use of Deep Learning for the automated understanding of shared feelings through digital media stands out. However, predicting feelings is a complex task, since, due to human subjectivity, similar contents can be interpreted in different ways. This work presents the development of two different approaches involving the area of Deep Learning, to analyze the feelings expressed by digital images considering two classes (Positive and Negative). One approach directly analyzes the visual content of the image, while the other makes a textual interpretation of the media content to later categorize the feeling of the figure based on the extracted caption. Both approaches had practically similar results, but during the benchmark process, the result favored the visual approach with an accuracy value of 63% due to its simplicity of flow and the difficulty of the second approach in bringing feelings into the legend during the figure description process. For future work, it is suggested to improve the architectures developed so far and possibly the addition of new architectures of convolutional networks to add to the benchmark collected.

Key-words: deep learning, image sentimental analysis, neural networks, natural language processing.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

DL	Deep Learning
ML	Machine Learning
CNN	Rede Neural Convolucional
PLN	Processamento de Linguagem Natural
AVG	Average
BTT	Backpropagation Through Time
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
CSV	Comma Separeted Values
SVM	Support Vector Machine

Sumário

1 Introdução

1.1 Contexto

Nos últimos anos, uma das grandes áreas de pesquisa da inteligência artificial é a compreensão do sentimento humano, cujo proposito é investigar técnicas baseadas em aprendizado de máquinas para classificar sentimentos associados a diversos contextos. O contexto mais comum e popular na literatura é a classificação de sentimentos a partir de textos, onde sua aplicação abrange desde a classificação de emoções em *reviews* das experiências de consumidores, sejam elas positivas ou negativas, até mesmo no uso da técnica para previsão das emoções referente ao que é compartilhado em redes sociais como o Twitter (BOLLEN; MAO; PEPE, 2011).

Uma das abordagens baseadas em sentimentos, com pesquisas crescentes na literatura, visa a classificar os sentimentos a partir de imagens. As imagens compartilhadas nas redes sociais, por exemplo, são elementos cujo conteúdo pode conter diversos sentimentos associados. Esses sentimentos podem ser notados tanto em sua composição visual, quanto nas descrições textuais atribuídas pelo próprio autor (ou usuário) que compartilhou o conteúdo.

O uso da análise de sentimentos expressos por imagens é uma ferramenta muito útil em diversos cenários, por exemplo, na identificação de conteúdos visuais impróprios nas redes sociais, com o intuito de bloquear as mídias inadequadas ao resto dos usuários quando necessário (TAHIR et al., 2019). Outra aplicação possível é a análise de padrão de um perfil dentro de uma plataforma mediante ao que o usuário compartilha, para identificar perfis com tendências depressivas ou agressivas (LI et al., 2019).

Entretanto, existe um conjunto de desafios para o desenvolvimento de uma ferramenta automática na análise de sentimentos a partir de imagens digitais. Um dos desafios se refere ao fato de que a emoção é um sentimento abstrato para o ser humano, logo a detecção semântica através de uma imagem ocorre geralmente usando objetos presentes em composição visual como carros, prédios, animais, etc. Outro desafio é o fato do sentimento visual diferir de uma pessoa para outra, pois a opinião sobre uma determinada imagem é subjetiva (BORTH et al., 2013). O caso de memes, por exemplo, é um problema, visto que seu conteúdo pode não ter nenhum sentimento ou levar à interpretações ambíguas (TRUONG; LAUW, 2017a).

Classificadores baseados em redes neurais, especialmente as redes que utilizam a abordagem *Deep Learning* (ou aprendizado profundo), são valiosos instrumentos da área de aprendizado de máquina para a tarefa de classificação de sentimentos a partir de

imagens digitais. Isso se deve ao fato do uso de camadas profundas, que permitem extrair e caracterizar aspectos importantes da estrutura da imagem, de modo a encontrar padrões discriminativos que tendem a classificar com maior eficácia o conteúdo da imagem.

O escopo deste trabalho é a avaliação do desempenho de classificadores baseados em *Deep Learning* na tarefa de classificação de sentimentos expressos por imagens considerando duas classes (*labels*): "Positiva" ou "Negativa". A Figura 1 mostra exemplos de cada uma dessas duas classes. Imagens classificadas como Positiva, tendem a ter conteúdos mais felizes ou situações mais agradáveis ao cérebro humano, já as classificadas como Negativas, tendem ser o oposto, abordando contextos tristes ou violentos.

Figura 1 – Exemplos de imagens categorizadas em dois sentimento:

(a) Positivo (b) Negativo

Fonte: (VADICAMO et al., 2017)

O desenvolvimento de um classificador de sentimentos baseados em imagens é uma tarefa complexa e desafiadora, visto que pode existir ambiguidade na interpretação do conteúdo da imagem e, por isso, depende fortemente do contexto envolvido na cena.

O reconhecimento de objetos, por exemplo, quando comparado à categorização de pessoas é ainda mais desafiador. Geralmente uma pessoa contem expressões e dependendo da harmonia de músculos em suas feições, as redes neurais conseguem distinguir facilmente o que esse indivíduo está sentindo. Para objetos, o reconhecimento pode envolver mais de uma emoção compreendida dependendo do contexto que a imagem está envolvida, como é o caso mostrado na Figura 2, que ilustra o mesmo objeto em contexto diferentes e que geram sentimentos opostos.

1.2 Objetivos e Problema de Pesquisa

O objetivo geral deste trabalho é investigar técnicas de *Deep Learning* para classificar sentimentos positivos e negativos expressos a partir de imagens digitais.

Figura 2 – Imagens contendo o mesmo objeto, porém em diferentes contextos.

(a) Figura de homem suspeito segurando uma faca

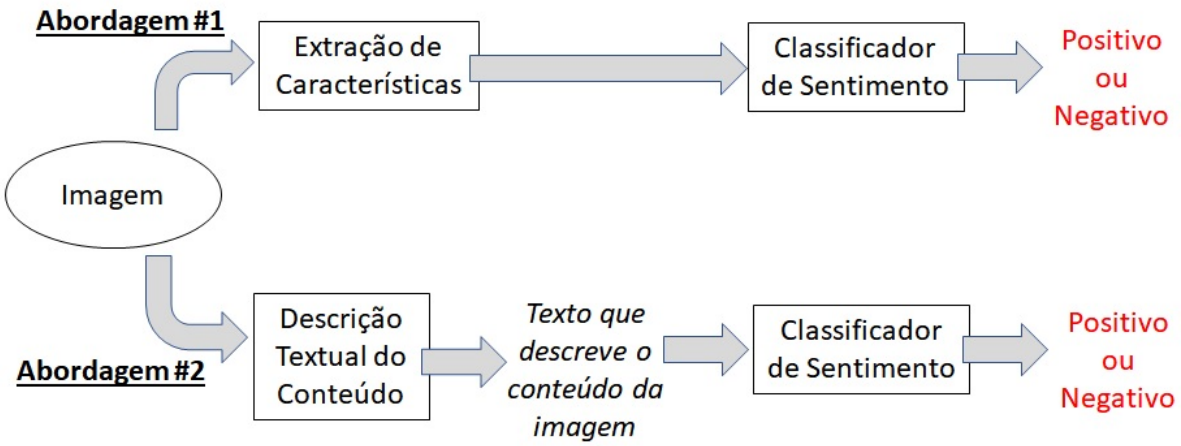
(b) Figura de faca cortando legumes

Fonte: (Pixabay, 2022)

Como problema de pesquisa, a fim de investigar abordagens que pudessem mitigar o problema da ambiguidade na interpretação do conteúdo de uma imagem, este trabalho comparou duas abordagens diferentes baseadas em *Deep Learning* para classificação de sentimentos. Essas duas abordagens estão esquematizadas na Figura 3 e são definidas como:

- Abordagem #1: classifica o sentimento de imagens a partir da extração de características visuais da imagem;
- Abordagem #2: extrai legendas (descrição) do conteúdo da imagem e classifica o sentimento do texto obtido dessa descrição.

Figura 3 – Abordagens de classificação exploradas nesse trabalho



Fonte: Autoria Própria

Para atingir o objetivo geral e contemplar a investigação proposta nesse trabalho, três objetivos específicos foram propostos, são eles:

- Treinar um modelo de *Deep Learning* para classificar o conteúdo visual de uma imagem em um sentimento positivo ou negativo.
- Treinar outro modelo de *Deep Learning* para transcrever o conteúdo da imagem e posteriormente classificar esse texto extraído entre o sentimento positivo ou negativo.
- Comparar as duas abordagens sobre o mesmo *benchmark* e verificar qual possui o melhor desempenho em termo de acurácia.

1.3 Organização da Monografia

Esta monografia está organizada da seguinte forma:

- Capítulo 2: descreve os conceitos e definições importantes para o entendimento do trabalho, assim como os trabalhos correlatos;
- Capítulo 3: apresenta a metodologia de desenvolvimento do trabalho. Nessa seção serão abordadas as estratégias e recursos necessários ao desenvolvimento do trabalho;
- Capítulo 4: apresenta os resultados coletados no decorrer do desenvolvimento do projeto de acordo com as abordagens levantadas no capítulo de metodologia do trabalho;
- Capítulo 5: apresenta a conclusão e possíveis projetos para o futuro.

2 Fundamentação Teórica

2.1 Análise de Sentimento Baseada em Imagens

A área de análise de sentimento é um sub-campo da área de processamento de linguagem natural (NPL). Geralmente, é a área responsável por minerar opiniões em volume de dados principalmente em texto, comumente usada para verificar a polaridade semântica de uma determinada frase textual ou um problema de classificação se levar para a área de inteligência artificial. Logo, esse ramo é famoso pela versatilidade de transformar dados não estruturados em alta escala de modo a responder questões específicas, além de usar esses mesmos dados para prever tendências futuras junto aos comportamentos populacionais e usar a informação recolhida por várias pessoas de modo a coletar opiniões sobre os mais variados temas de forma estruturada (OLIVEIRA et al., 2018).

Com o aumento de abordagens de inteligência artificial em mídias digitais, essa técnica também vem sendo usada junto a imagens (tema abordado nesse trabalho), com objetivo de gerar descrições sobre a sua composição, usando como base outros algoritmos de *Deep Learning* como entrada para suas análises e posteriormente classificar entre os diferentes sentimentos treinados. Hoje com o crescimento dos recursos computacionais e o aumento no desempenho dos algoritmos, ficou mais fácil usar técnicas robustas voltadas a mídias visuais. A abordagem usa a mesma linha de pensamento da análise de sentimento em texto, mas focado no uso de *Deep Learning* para gerar a descrição e assim validar o sentimento através da composição semântica (GAJARLA; GUPTA, 2015).

2.1.1 Trabalhos Correlatos

Para o tema de análise de sentimento a partir de imagens é possível encontrar na literatura alguns trabalhos desenvolvidos. A seguir serão apresentados alguns desses trabalhos:

- KUMAR e JAISWAL (KUMAR; JAISWAL, 2017) destacam o uso da técnica de *Deep Learning* com o uso de Redes Neurais são de grande vantagem na geração de texto, modelagem e classificação de frases e etc. Com a ajuda das Redes Neurais Convolucionais, a possibilidade de usar este recurso para transcrever imagens sucintamente junto a análise de texto através do processamento de linguagem natural (NPL), pode ser um grande aliado na classificação de sentimento em recursos visuais.
- No trabalho desenvolvido por (BORTH et al., 2013), é ressaltado a importância

do uso de *Adjective Noun Pairs* (ANP) na identificação de sentimentos durante o processo de classificação. A técnica referenciada é robusta comparada as técnicas ao uso singular de adjetivos ou substantivo para identificar um determinado sentimento, pois os dois elementos são combinados de forma que o adjetivo proporciona um valor semântico maior, definindo uma qualidade ao substantivo e o substantivo por sua vez é caracterizado pelo adjetivo. Outro ponto destacado é a aplicação de uma ontologia para definir o escopo a qual sentimentos as imagens seriam classificadas. Para (BORTH et al., 2013) o uso de ontologia junto ao desenvolvimento proporcionou um embasamento aprofundado a teorias concretas da psicologia para definir os sentimentos mais comuns.

- A técnica usada por (ORTIS et al., 2020) explora a classificação de diferentes sentimentos correlacionados a aspectos visuais e textuais presentes em imagens. O tópico principal do trabalho é a comparação entre a classificação de sentimento entre textos objetivos e subjetivos. Os textos subjetivos são termos onde o próprio usuário ou autor da imagem adicionam para descrevê-la (título, descrição, *tags*, etc). Os textos objetivos são aqueles gerados por um algoritmo de *Deep Learning* relacionados ao conteúdo visual presente na imagem para descrevê-la. No estudo realizado é possível notar o impacto positivo dos textos gerados por algoritmos de *Deep Learning* comparado aos descritos pelos autores das imagens. Devido à quantidade de ruído presente nas descrições de textos subjetivos, a predição pode ser comprometida durante a análise de texto.

- TRUONG e LAUW (TRUONG; LAUW, 2017b) propuseram usar Redes Neurais Convolucionais também para classificar sentimentos em imagens presentes em *reviews* do *Yelp.com*. Em seu trabalho, os autores dividiram o processo de treinamento em três abordagens diferentes relacionadas a adaptação da arquitetura *AlexNet*, uma rede neural convolucional ganhadora do desafio ImageNet LSVRC em 2012 com 84% de acurácia, onde é composta por 5 camadas convolucionais e 3 camadas conectadas. A primeira é orientada ao conteúdo presente na própria imagem, a segunda orientada ao usuário (sentimentos expressados por quem escreveu o *review*) e a terceira em relação aos sentimentos associado a itens (animais, pessoas, objetos, etc.) presentes na composição da imagem.

- No artigo publicado por (MITTAL; SHARMA; JOSHI, 2018) é realizado uma análise em estudos antigos que envolve o tema de classificação de sentimento em imagens usando técnicas de *Deep Learning*. Na análise apresentada, são abordados diferentes arquiteturas de redes neurais entre elas a *Deep Neural Network* (DNN), *Convolutional Neural Network* (CNN), *Region-based CNN* (R-CNN) e *Fast R-CNN*. Após a análise dos estudos é concluído que o uso de CNN é mais eficiente e preciso comparado ao resto das arquiteturas levantadas, com aumento de quase 20 por cento

de precisão comparado aos outros modelos apresentados. O motivo de sua eficiência é o ótimo desempenho, com o menor número de requisitos e recursos para realização das tarefas.

- No trabalho proposto por (GAJARLA; GUPTA, 2015), há a implementação de algumas adaptações de Redes Neurais Convolucionais para fazer a classificação de sentimento de imagens através do conteúdo visual. Dentre elas encontra se as redes neurais VGG-Places205, One vs All SVM, VGG-ImageNet e ResNet-50. Foram considerados durante a classificação as categorias "amor", "felicidade", "violência", "medo" e "tristeza", além de subdividir estas entre as categorias "Positivas" e "Negativas". Na etapa de coleta e tratamento de dados, foi usado o *dataset* do Flickr, com a ajuda da API disponibilizada pela plataforma. Ao analisar os resultados obtidos, o algoritmo que mais se adequou durante o desenvolvimento foi o ResNet-50, com uma acurácia de 73,3 por cento ao classificar entre positivo e negativo e 40,5 por cento na classificação das cinco categorias acima.

2.2 Tipos de Aprendizado de Máquina

Na área de aprendizado de máquina há diferentes formatos de aprendizagem para os algoritmos assimilarem os padrões no problema. Esses modos são:

- Aprendizado supervisionado;
- Aprendizado não supervisionado;
- Aprendizado por reforço.

O aprendizado supervisionado tem por objetivo, mapear um conjunto de dados e a partir deles, achar uma função f onde tente generalizar o conjunto de dados alimentado pelo algoritmo durante a fase de treino. Logo, a partir do treino do algoritmo, a função tende ao conjunto dado interpretado por $g = (i, f(i))$, dado que i são os dados e $f(i)$ suas respectivas respostas (LOVE, 2002).

No aprendizado não supervisionado o algoritmo usa dados ou parâmetros iniciais para iterar esses dados e poder classifica-los da forma correta, procurando também uma função generalista para as informações dadas. O aprendizado não supervisionado é usado especialmente quando dado um conjunto de dados i não contem rótulos oficiais para serem dados ao algoritmo para que treinem usando uma abordagem guiada, logo é usado informações estruturais presentes no *dataset* analisado para encontrar padrões similares (LOVE, 2002).

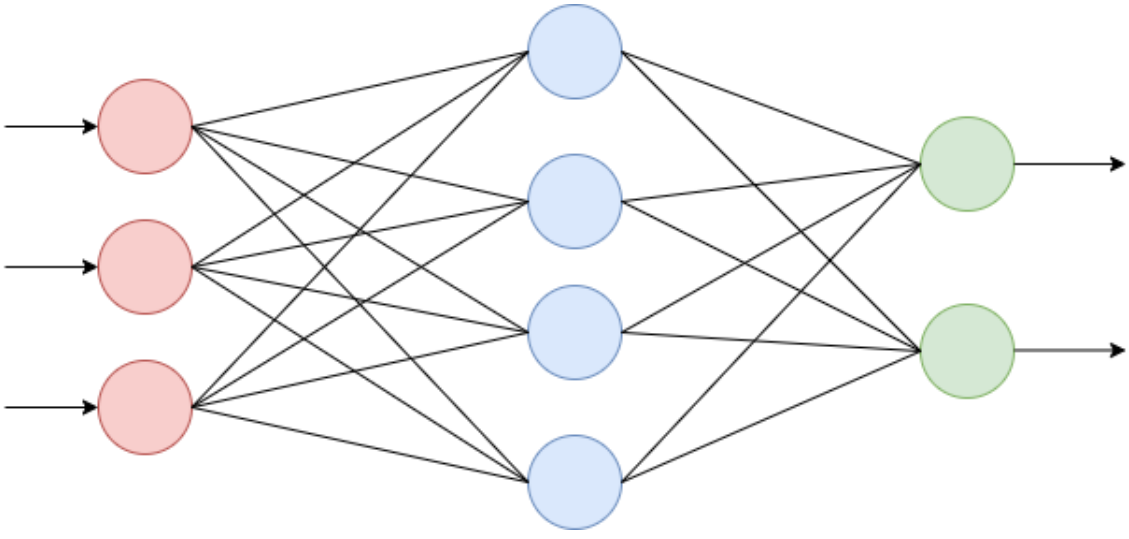
Já no aprendizado por reforço, são criados agentes usados para encontrar a melhor solução desejada pelo arquiteto da solução. Esses agentes individualmente não refletem nenhum resultado significativo para solução, logo sua significância é dada por suas interações junto a ambientes isolados e complexos. Após este passo, os melhores resultados são replicados para uma próxima iteração usando como base o sistema de recompensas e penalidades (SUTTON, 1992).

2.3 Aprendizado Supervisionado usando Aprendizagem Profunda

De acordo com (BEGUM; FATIMA; SABAHATH, 2019) a aprendizagem profunda ou também conhecida como *Deep Learning* é um sub ramo da área de inteligência artificial, responsável por modelar e abstrair formas da realidade, com o intuito de encontrar padrões através de um conjunto de dados para um determinado objetivo ou regra de negócio proposta. Esta área foi criada com objetivo de simular o comportamento do cérebro humano, mapeando alguns termos e funções presentes na neurologia para seus algoritmos, onde o fluxo é iniciado com a unidade mais básica chamada Perceptron. Os Perceptrons são responsáveis por transformar um conjunto de sinais de entrada a um conjunto de sinais de saída, análogo ao comportamento cerebral. Na Figura 4 é possível visualizar uma arquitetura básica de Perceptrons interconectados e sua similaridade aos nerônios biológicos (VALLIANI; RANTI; OERMANN, 2019).

O termo "profundo" se refere ao conjunto de neurônios em diversas camadas usadas para formar uma espécie de rede, visando reconhecer padrões através de dados coletados do mundo real, de modo a converter em saídas significativas para problemas de classificação e regressão.

Figura 4 – Arquitetura básica de uma Rede Neural com 3 camadas e vários neurônios em cada camada.



Fonte: Autoria Própria

Existem diversas formas exploradas para encontrar esses padrões e algumas delas se encontram abaixo.

2.3.1 Redes Neurais

As redes neurais são algoritmos matemáticos presentes no ramo de *Machine Learning*, usado para o aprendizado de um determinado grupo de dados. Uma rede neural definida como (BEGUM; FATIMA; SABAHATH, 2019):

> Uma tripla ordenada (N, V, w) com dois conjuntos N, V e uma função w, onde N é o conjunto de neurônios e V um conjunto $\{(i, j)|i, j \in N\}$ cujo os elementos são chamados de conexões entre o neurônio i e o neurônio j. A função w: V $\rightarrow$ R define os pesos, onde w, o peso da conexão entre o neurônio i e o neurônio j.

Este algoritmo tem como inspiração a arquitetura do cérebro humano, usando como base os neurônios, a comunicação entre eles, além do efeito de sinapse (comunicação entre neurônios). Os principais componentes dentro de uma rede neural são os neurônios, suas conexões e o uso de funções de ativação para a passagem de dados para outros neurônios. Por muito anos o algoritmo de SVM (Support Vector Machine) foi considerado um dos melhores para identificar padrões em dados, mas com a idealização e criação dos algoritmos de redes neurais junto ao crescimento computacional, foi possível alavancar a área de inteligência artificial voltada ao *Deep Learning* a outro nível.

2.3.2 Perceptron

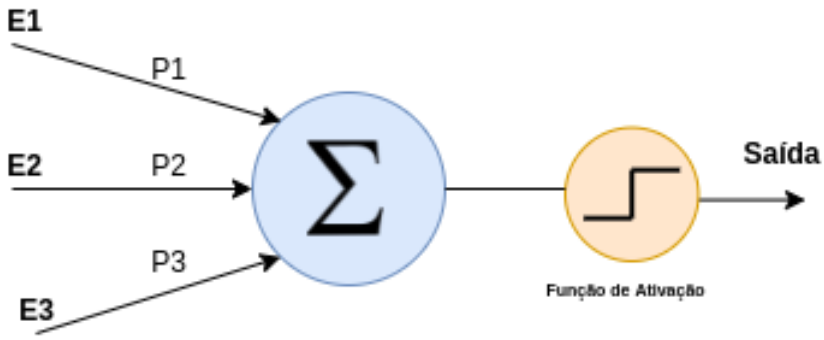
Pela definição de (KRIESEL, 2007), o perceptron é a unidade mais básica de uma rede neural, composto por apenas um neurônio, suas entradas, pesos e saída. Os perceptrons combinados com vários outros constituem os algoritmos de redes neurais e suas variações. Um perceptron é composto por um conjunto de valores de entradas denominados $E = \{e_0, e_1, ...e_n\}$ e de pesos $P = \{p_0, p_1, ...p_n\}$. Durante o processamento do perceptron exitem as etapas de programação e ativação, para geração do resultado do neurônio.

A função de propagação é a primeira etapa do processamento, onde resulta em um escalar α. O processamento da função de propagação é a junção dos valores de E e P multiplicados e somados, definidos pela fórmula abaixo:

$$f_{prop} = \sum_{i=0}^{n} e_i \cdot p_i \tag{2.1}$$

Após o resultado do f_{prop} com o valor de α, é usado a função de ativação, onde seu comportamento tem por objetivo imitar os sinais de sinapse do cérebro humano, inibindo

Figura 5 – Arquitetura simples de um Perceptron e suas principais etapas.



Fonte: Autoria própria

ou excitando o neurônio para passar a mensagem para o próximo neurônio ou saída da rede neural. Para poder excitar ou não o neurônio, é usado um valor limitante onde a partir da função de ativação $f_{ativ.}$ é passado o valor ou não adiante.

$$\beta = f_{ativ.}(\alpha, lim) \tag{2.2}$$

2.3.3 Rede Neural Convolucional

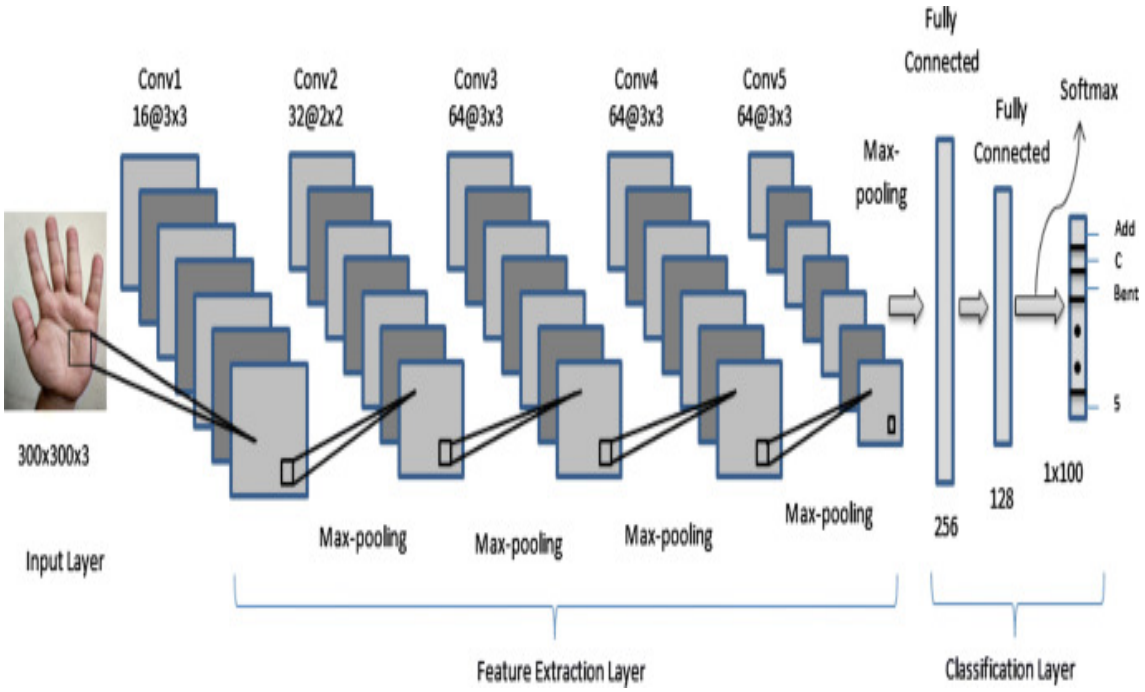
Um dos motivos para o avanço da área de visão computacional foi a criação das Redes Neurais Convolucionais ou *Convolutional Neural Networks* (CNNs) do inglês. Algoritmos usados especificamente para processar dados em formato de múltiplos *arrays*, impulsionado pelos pesquisadores franceses (LECUN et al., 1998). Diferente do modelo de rede neural multi camada, o uso de camadas convolucionais nas arquiteturas presentes tem o objetivo de enaltecer bordas, curvas e linhas na imagem e usar esses padrões encontrados durante o treinamento e a predição para criar uma linha de base para classificar ou reconhecer os objetos para o contexto do problema.

Geralmente a arquitetura básica de uma CNN é composta por três elementos principais em sua constituição, são elas as camadas convolucionais, camadas de *pooling* e as camadas conectadas, onde cada uma das etapas é responsável por extrair as *features* e processar os valores obtidos para achar os pesos necessários que compõe o padrão. A Figura 6 mostra de forma simples as etapas da arquitetura de uma CNN.

2.3.3.1 Convolução

A convolução é usada para extrair as características mais importantes presentes nas entradas. Durante a etapa de treinamento, a convolução age como filtros ajustados no decorrer da análise dos dados para se adequar melhor ao padrão do conjunto, com referência a elementos comumente presentes em imagens, como é o caso de cores, arestas, bordas entre outros (ALBAWI; MOHAMMED; AL-ZAWI, 2017).

Figura 6 – Exemplo visual de uma Rede Neural Convolucional.



Fonte: (WADHAWAN; KUMAR, 2020)

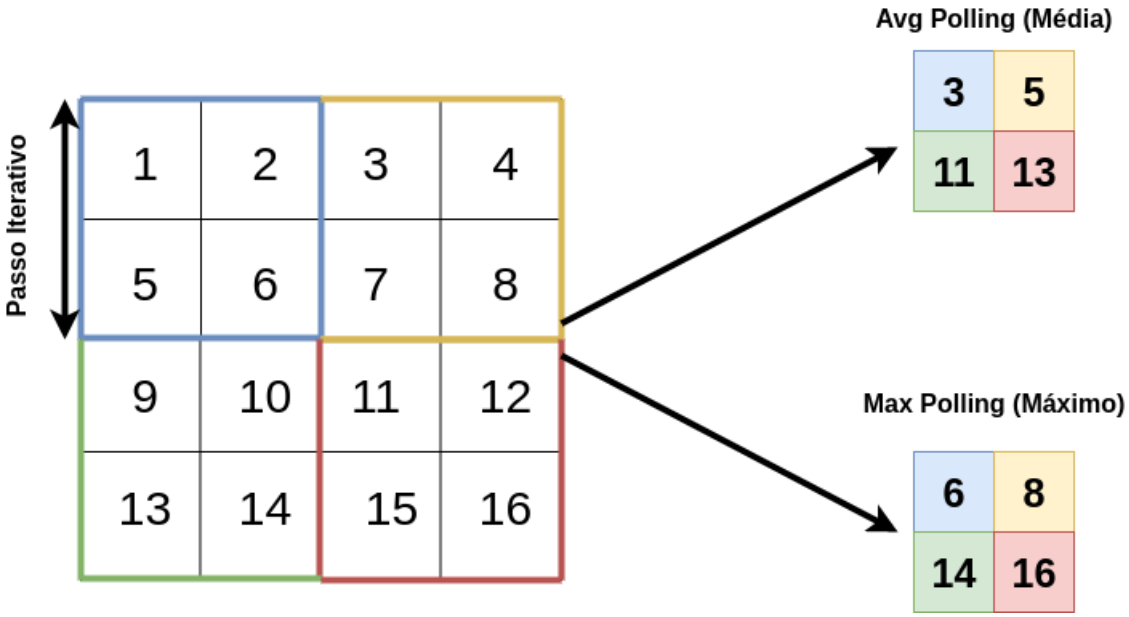
Quanto mais camadas de convolução, mais complexa e melhor a estrutura compreende a forma dos dados tratado com mais *features* extraídas. Enquanto manter um número alto de camadas pode ser um ponto positivo para o reconhecimento do molde, maior o poder computacional é necessário para processar todo o conteúdo e maior o tempo para analisar. Logo, toda a arquitetura deve ser bem definida, visando balancear essas variáveis para alcançar a maior eficácia e evitar *overfitting* ou *undefitting* da rede (O'SHEA; NASH, 2015).

2.3.3.2 Pooling

A segunda etapa pós convolução é a de *Pooling*, onde seu objetivo principal é extrair as características recuperadas das etapas convolucionais de modo a reduzir o tamanho dos dados, utilizando os atributos mais marcantes do conjunto processado, com o intuito de reduzir o *overfitting* e o custo computacional durante as etapas de treinamento (O'SHEA; NASH, 2015).

Na Figura 7 é possível visualizar as etapas principais de execução de *Pooling*. Primeiramente é definido o tamanho do passo (também chamado *stride*) iterativo de análise dos dados. Em seguida, é escolhido uma entre as técnicas de *Max ou Average Pooling* para computar o valor máximo ou média dos valores do passo e posteriormente gerar um novo conjunto menor, para melhorar o desempenho da rede com um volume menor para as próximas camadas (O'SHEA; NASH, 2015).

Figura 7 – Exemplo da técnica de *Pooling*



Fonte: Autoria própria

2.3.4 Multicamadas conectadas

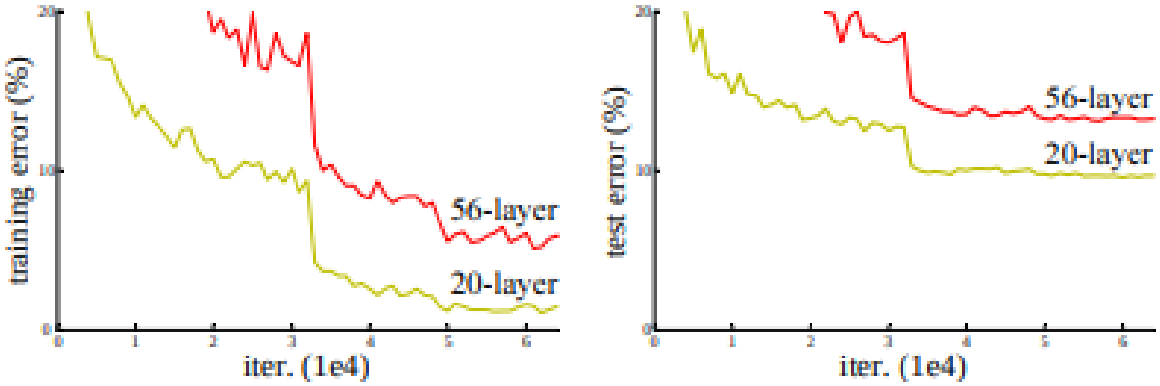
Esta é a última etapa da arquitetura convolucional, mas também é semelhante a arquitetura mais simples implementada no universo de aprendizado profundo, como é possível visualizar na Figura 4. Este modelo é composto por vários *perceptrons* conectados aninhados em camadas, de modo a analisar os valores extraídos das camadas de convolução e de *pooling* e em seguida classificar os dados atuais entre as *labels* treinadas durante a etapa de treinamento da rede neural (ALBAWI; MOHAMMED; AL-ZAWI, 2017).

2.3.5 Rede Neural Residual

Durante os testes organizados por (HE et al., 2015), foi verificado se o aprendizado de uma rede neural é proporcional a capacidade de adicionar cada vez mais camadas em sua arquitetura (modelos acima de 50 camadas). Após treinar duas arquiteturas com números de camadas diferentes, foi possível observar que o erro nas etapas de teste e treinamento da rede aumentava mesmo com o acréscimo de camadas, como é possível observar na Figura 8. Essa eventualidade é dada graças ao problema de gradiente de fuga (ou *vanish gradient* do inglês) enfrentado nas etapas de aprendizado da rede.

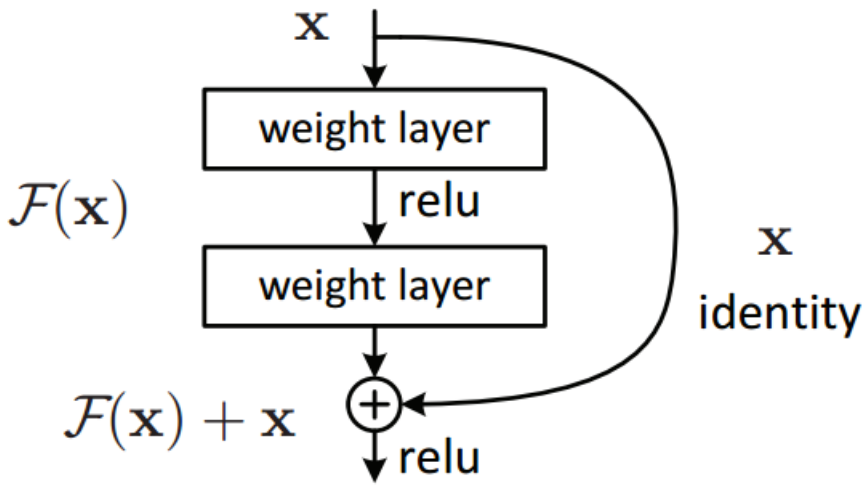
Em 2015 durante o evento ILSRVC, foi apresentada por (HE et al., 2015) a arquitetura residual como uma alternativa de rede neural artificial convolucional para contornar o erro de gradiente e degradação dos parâmetros (aprendizado antes da chegada da última camada), para oferecer precisão com altas profundidade de redes e um alto grau de otimização. A parte principal do algoritmo consiste em um bloco residual ilustrado na Figura 9.

Figura 8 – Gráficos com os erros de treino e teste após o acréscimo de mais camadas durante as etapas de treinamento e teste proposto por (HE et al., 2015).



Fonte: (HE et al., 2015)

Figura 9 – Bloco residual presente na arquitetura de rede neural residual.



Fonte: (HE et al., 2015)

O problema é resolvido devido à etapa de salto ou atalho presente entres as camadas durante o treinamento, para não atrapalhar o fluxo de dados nesta etapa evitando que a degradação do gradiente aconteça. O movimento de salto na arquitetura também ajuda a evitar que a rede aprenda a encontra um conjunto de valores adequados por conta própria para uma situação onde a aplicação de uma função F(x) possa ser convertido em zero em vez de apenas pular a etapa, assim evitando processamento desnecessário e a explosão ou desaparecimento do gradiente. Ou seja, o modelo tem por objetivo ser treinado com um número maior de camadas e extrair o potencial máximo do algoritmo criado.

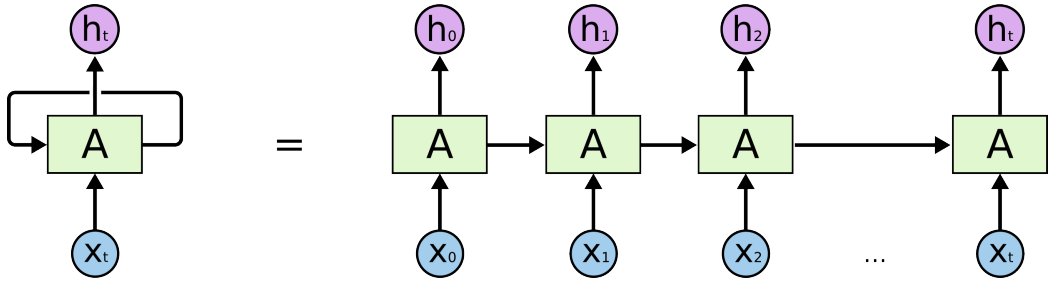
2.3.5.1 Gradiente de fuga (Vanish Gradient)

De acordo com (KIM, 2017), durante a retro-propagação nas fases de treinamento e o aumento de camadas nas redes neurais, foi possível visualizar dificuldades de convergência dos valores dos pesos graças a degradação do gradiente descendente estocástico medido, ou seja, os valores de gradientes calculados podem se tornar muito pequeno ou muito grande durante o aprendizado do algoritmo. O desaparecimento ou explosão do algoritmo atrapalha a convergência com as atualizações de parâmetros muito pequenas para gradientes mínimos e atualizações muito grandes para gradientes enormes, de modo a evitar os valores ideais para o conjunto treinado. O gradiente descendente estocástico é a métrica usada para atualizar os parâmetros da rede neural treinada, de modo a encontrar o mínimo erro global presente no problema, onde usa funções de custo com base no erro encontrado durante as épocas de treinamento para atualizar os pesos da rede neural.

2.3.6 Rede Neural Recorrente

As redes neurais recorrentes são algoritmos mais robustos comparado as redes neurais artificiais convencionais. O principal diferencial das redes recorrentes com as redes neurais básicas *feedfoward* é o fato das camadas ter uma espécie de memória, capaz de guardar os estados antigos em camadas anteriores que serve como base para ajuste de estados atuais, este cenário ilustrado na Figura 10. O uso mais comuns para essa arquitetura são em processamento de linguagem natural e texto, devido ao forte recurso em lidar com dependências, já que este usa o recurso de memória de estado para prever cenários de contexto na linguagem (ALSENAN; AL-TURAIKI; HAFEZ, 2020).

Figura 10 – Representação da arquitetura de uma rede neural recorrente.



Fonte: (OLAH, 2015)

Na figura acima é possível visualizar o formato como o neurônio recorrente é usado, no caso, em um determinado tempo t temos entradas denotadas pela variável X, onde, gera saídas h, mas como o neurônio A contém a capacidade de recorrência, logo é possível receber entradas anteriores com a x_t e a saída h_{t-1} (MOURA et al., 2018). A imagem à direita da Figura 10 exemplifica o mesmo neurônio a esquerda, mas aninhado com vários outros.

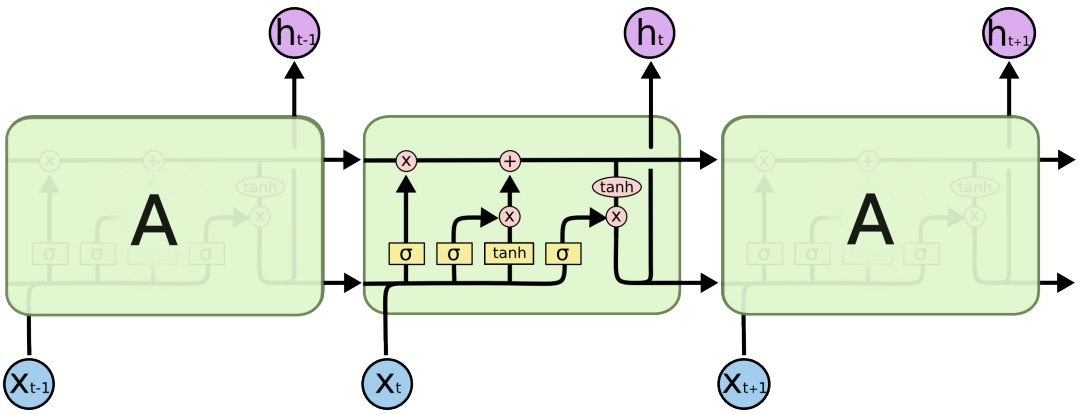
Como as RNNs tem uma grande similaridade com as DNN convencionais, há também a etapa de *backpropagation* para ajuste dos pesos da rede, mas como é uma rede que usa como base o tempo para definir seus estados, sua forma de ajuste é determinado como *backpropation through time* (BTT) ou *backpropagation* através do tempo. Assim como explicado na subseção 2.3.5.1, as RNNs ao usar a técnica de BTT também sofrem do problema de gradiente de fuga, onde atrapalha no ajuste dos pesos calculado. Para resolver esse problema foi criado uma arquitetura alternativa de rede neural recorrente denominada LSTM (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.3.6.1 Long Short Term Memory - LSTM

Para (PATEL, 2017), a arquitetura *Long Short Term Memory* é uma rede neural recorrente, onde seu diferencial consiste em uma unidade de memória, onde mantém certos dados por um longo período o quanto for necessário.

A estrutura desse tipo especial de arquitetura contém especificamente três camadas sigmoids e uma camada tanh, onde são responsáveis por atualizar o valor, descartar ou manter para a próxima iteração, para encontrar novos padrões para a solução treinada. A Figura 11 ilustra o formato do neurônio da LSTM de uma forma clara.

Figura 11 – Arquitetura meramente ilustrada da rede neural Long Short Term Memory.



Fonte: (PATEL, 2017)

Como mostrado na Figura 11, o sistema contém três camadas sigmoids, onde a primeira camada consiste em extrair as informações dos valores de h_t e x_t sendo respectivamente os estados anterior e o valor de entrada atual, para concatená-los. No próximo passo é analisado o quanto das informações capturadas resultadas dos valores de h_t e x_t serão propagados para os estados atualizado. Com esses valores resultantes, é aplicado o uso da camada tahn, em que atualizará os valores para a criação de novas informações ou padrões. Com uma nova camada de sigmoid é mesclado essas novas informações com os valores salvos na memória anteriormente. Na última etapa é calculado o valor de saída ou h_t para a iteração atual (PATEL, 2017).

2.4 Aprendizado Supervisionado usando Naive Bayes

O Naive Bayes é conhecido como o algoritmo Bayesiano mais básico de classificação. É um dos algoritmos mais usados no ramo de classificação de textos, pois tende de verificar a probabilidade de um determinado documento d, onde contém um conjunto de palavras p pertencer a uma classe c (DAI et al., 2007). Calculado notoriamente pela fórmula:

$$P(c|d) \ \alpha \ P(c) \cdot P(d|c) \tag{2.3}$$

Pode ser reescrita pela fórmula, onde é feito o produto da probabilidade de cada palavra pertencer a uma classe, onde esta palavra esta engloba o contexto do documento analisado:

$$P(c|d) \ \alpha \ P(c) \cdot \prod_{p \in d} P(p|c) \tag{2.4}$$

2.5 Processamento de Linguagem Natural

O processamento de linguagem natural é uma ferramenta no ramo de inteligência artificial cujo interesse é usar o computador como ferramenta para automatizar o entendimento da linguagem verbal humana. Onde tem por interesse a análise e geração de escrita de linguagem faladas, usada como tradutor do usuário para a máquina através da linguagem humana e não binária (MEURERS, 2012).

Para (MEURERS, 2012) a forma do ser humano se comunicar pode ser muitas vezes complexa, pois a adaptação dos algoritmos podem ser um grande desafio. Com a frequente incrementação de novos formatos de comunicação na língua (gírias, abreviações e etc...), os algoritmos precisam ser frequentemente adaptados e treinados ao contexto em que a abordagem de NPL está sendo aplicada.

O NPL pode ser aplicado em diversas atividades do cotidiano humano, como é caso da previsão de buscas em sites como o Google, aplicação de assistentes virtuais para entender e processar atividades requeridas por usuário, além de ser usada em Chatbots com o entendimento das frases para ajudar os usuários durante um determinado processo.

2.5.1 Tokenização

Nessa etapa a frase recebida é separada em diversas palavras, geralmente por espaço. Dependendo da linguagem, diferentes técnicas são usadas como é o caso do dialeto Mandarim onde o espaço não diz ser o final de uma palavra. A Figura 12 mostra um exemplo de frase “tokenizada” (SANYAL et al., 2017).

Figura 12 – Exemplo do processo de tokenização em uma frase

Fonte: Autoria própria

2.5.2 Stopwords

A remoção de *stopword* é o passo onde são removidos os elementos da frase onde não há um significado semântico para o contexto, geralmente são artigos que servem como conectores no contexto geral da frase, mas não são significativos para a análise textual (SANYAL et al., 2017).

2.5.3 Correção Ortográfica

Etapa usada para o tratamento de conjunto de dados que contém erros ortográficos em sua composição textual. Sua função é evitar a criação indevida de novos *tokens*. Para tal ação é necessário um grande corpus do idioma para que o algoritmo consiga prever a qual palavra o texto se encaixa semanticamente (SANYAL et al., 2017).

2.5.4 Stemização e Lematização

A Stemização e Lematização tem por objetivo analisar as palavras capturadas durante o *pipeline* e transformar estas em uma forma primitiva onde é possível trabalhar para o entendimento treinado, ou seja, tem por objetivo diminuir o vocabulário e abstrair o significado. Na Stemização as palavras são reorganizadas pelo seu radical, logo considerando um conjunto de famílias de palavras, como “bonecas”, “bonequinho”, “bonecos” são abreviados para “bonec”, pois é o argumento comum em ambos os itens do conjunto. Por outro lado, a Lematização abrevia todos os termos para o infinitivo, logo para o exemplo citado o seu lema seria “boneco” (SANYAL et al., 2017).

2.5.5 Métrica Bleu Score

A métrica *Bleu* (bilingual evaluation understudy) é comumente usada para validar a qualidade dos textos gerados dos modelos de *machine learning* especificamente no ramo de processamento de linguagem natural (LIN; OCH, 2004). Esta métrica é responsável

particularmente por medir a quantidade de palavras presentes durante a geração do texto, comparados com o texto que deve ser de fato predito, ou seja, tender a uma sentença mais próxima do que o ser humano possa entender. Dado que c é são as palavras geradas pelo modelo e r a verdadeira sentença, para calcular a penalidade P da fórmula é usado:

$$P = \begin{cases} 1, & \text{if } c > r \\ e^{1-\frac{r}{c}}, & \text{if } c \leq r \end{cases}$$

Para (SUTTON, 1992), seja w_n os pesos positivos, p_n a média geométrica das precisões de n-grama e N o tamanho dos valores analisados, define-se o cálculo de métrica BLEU por:

$$BLEU = P \cdot exp(\sum_{n=1}^{N} w_n \log p_n) \tag{2.5}$$

2.6 Métricas para Validação de Classificadores

2.6.1 Acurácia

De acordo com (POWERS, 2020) acurácia é a métrica que diz a respeito a quanto dos itens classificados durante o processo de treinamento foram rotulados da forma correta para cada uma de suas classes. Dado que o total de rótulos classificados como positivo (TP), rótulos classificados como negativos (TN), rótulos classificados positivamente, mas com o resultado real negativo (TFP) e rótulos classificados negativamente, mas com resultado real negativo (TFN) é possível calcular a acurácia do modelo treinado com a seguinte fórmula:

$$acurácia = \frac{TP + TN}{TP + TN + TFP + TFN} \tag{2.6}$$

Não é uma métrica muito adequada para a validação final do modelo, pois verifica apenas quanto dos itens foram classificados e não destaca os erros ocorridos em TFN e TFP.

2.6.2 Precisão

Outro modelo de métrica comumente usado para avaliação de modelos de *machine learning* e *deep learning* (POWERS, 2020). Ao contrário da acurácia a fórmula de precisão foca nos erros de falsos positivos coletados, ou seja, tende de verificar qual dos itens

positivos analisados são realmente rotulados como positivos. A fórmula de precisão é dada por:

$$precisão = \frac{TP}{TP + TFP} \tag{2.7}$$

2.6.3 Recall

O *Recall* ao contrário da métrica de Precisão tem como foco a quantidade de falso negativos coletados, onde tende de validar qual dos itens positivos foram classificados corretamente como positivos. É possível definir essa métrica pela fração de itens rotulados corretamente como positivo dividido pelo número de itens que são realmente rotulados como positivo (POWERS, 2020).

$$precisão = \frac{TP}{TP + TFN} \tag{2.8}$$

2.6.4 F1 Score

O F1 Score é a métrica que usa outras duas métricas (*Recall* e Precisão) para encontrar o melhor equilíbrio (falsos positivos e falso negativos) e verificar a qualidade apurada do modelo (POWERS, 2020).

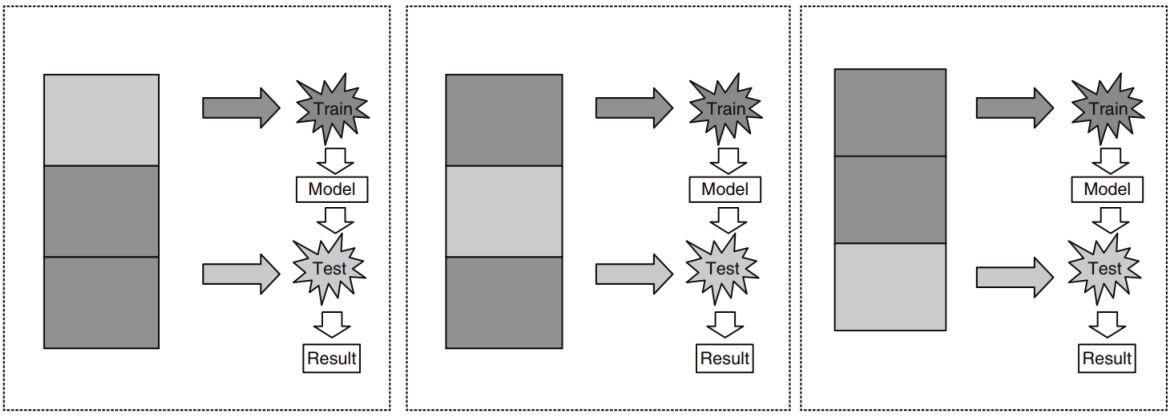
$$f1_{score} = 2 * \frac{Recall * Precisão}{Recall + Precisão} \tag{2.9}$$

2.6.5 Validação Cruzada

De acordo com (REFAEILZADEH; TANG; LIU, 2016) a validação cruzada é um método estatístico, onde avalia algoritmos baseados na divisão de dados em grupos de treinamento e validação. O reconhecimento da técnica se mostra através da iteração entre os conjuntos de treinamento e teste, onde cada uma das partes tem a oportunidade de ser usada para validar o conjunto treinado.

O algoritmo K-Fold é uma técnica da validação cruzada. O objetivo do algoritmo é dividir o conjunto de dados em K partes iguais, onde uma das partes divididas é utilizada para validação e o resto para treinamento. O número de partes divididas no algoritmo é correspondente com o número de iterações que o modelo irá treinar e validar os dados, mudando a cada iteração os dados de validação e treinamento, de modo que todas as partes em algum momento serão usadas para validar o modelo. Fluxo esse ilustrado na Figura 13.

Figura 13 – Exemplo do fluxo do algoritmo K-Fold.

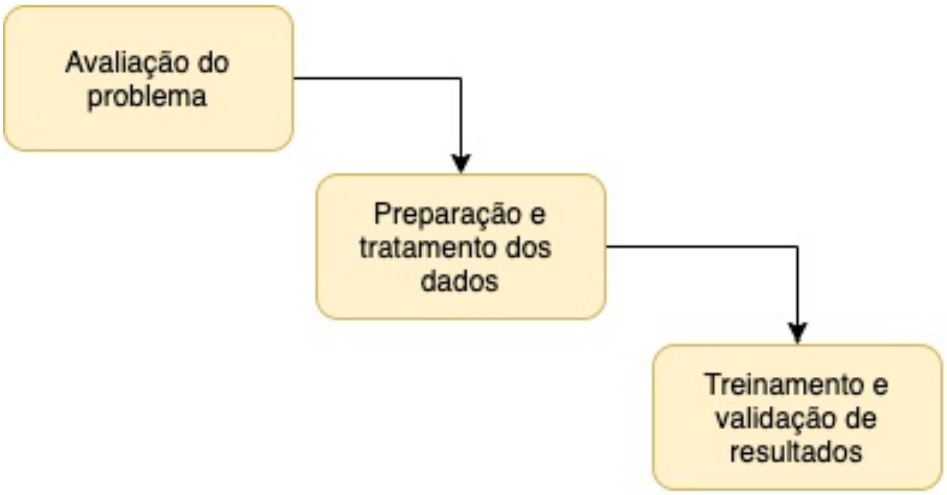


Fonte: (REFAEILZADEH; TANG; LIU, 2016)

3 Materiais e Métodos

Para o desenvolvimento deste trabalho foi definida uma estrutura metodológica para investigar as duas abordagens propostas como problema pesquisa. Essas etapas estão descritas no fluxograma da Figura 14 e que serão detalhadas a seguir nas próximas seções deste capítulo.

Figura 14 – Etapas realizadas no desenvolvimento do trabalho



Fonte: Autoria própria

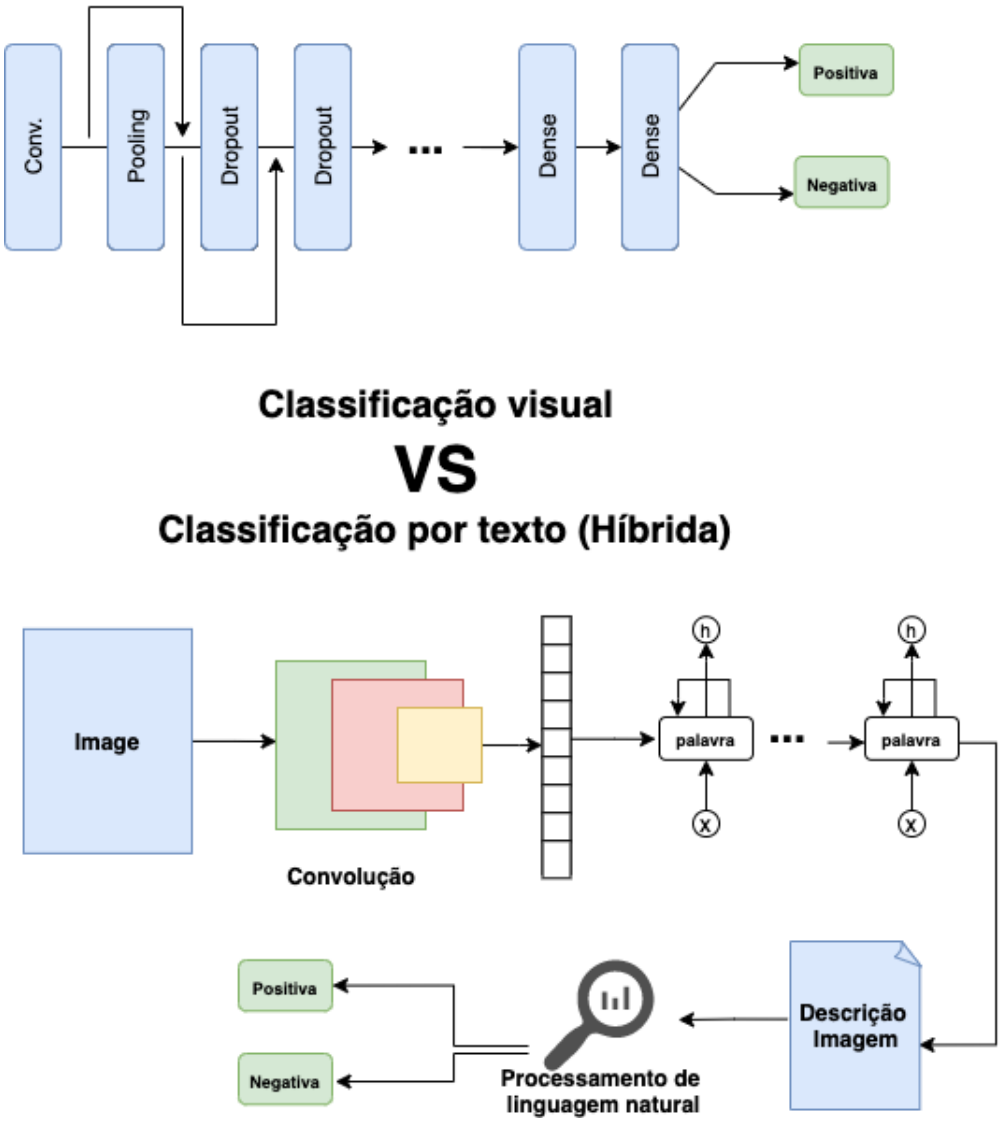
3.1 Avaliação do Problema de Pesquisa

A etapa inicial do desenvolvimento do trabalho foi avaliar e explorar o problema de pesquisa inicialmente proposto - classificação de sentimentos baseada em imagem. Esta etapa foi imprescindível para não se ter contradições durante o processo de desenvolvimento. A partir do levantamento de trabalhos correlatos na literatura, foi possível identificar as principais técnicas de implementação encontradas atualmente no ramo de *Deep Learning*, voltado para o reconhecimento de sentimentos em imagens e seus resultados.

Como o tema principal do trabalho é a classificação de sentimentos em imagens, foi pensado durante a elaboração e pesquisa de desenvolvimento, na comparação de diferentes algoritmos para realizar a mesma tarefa de classificação. Para validar as diferentes técnicas presentes atualmente, foi necessário verificar na literatura os métodos mais eficazes para realizar a classificação de imagens voltado para o reconhecimento de sentimentos. Depois de algumas pesquisas foi destacado duas arquiteturas específicas com diferentes abordagens, a primeira refere-se ao trabalho feito por (GAJARLA; GUPTA, 2015) com um

foco de classificação convencional (visual), onde é comparado diferentes arquiteturas de redes neurais para classificar sentimentos em imagens e dentre todas do *benchmark*, a que melhor se sobressaiu sobre as outras foi a arquitetura ResNet50, uma rede neural residual com cinquenta camadas com alta desempenho em treinamentos, criada pela Microsoft, logo esta foi escolhida como arquitetura principal para a abordadem #1.

Figura 15 – Esquema simplificado representando ambas as arquiteturas que foram implementadas no decorrer do projeto



Fonte: Autoria própria

A segunda arquitetura que obteve resultados interessantes, foi usada por (MINAEE; AZIMI; ABDOLRASHIDI, 2019), onde mesclou duas arquiteturas de rede neurais para gerar descrição da figura em texto, especificamente uma rede convolucional junto a arquitetura *long short term memory*, e com o uso da descrição é classificado o sentimento conforme o texto gerado. Logo, esse trabalho teve o objetivo de comparar a arquitetura recorrente ResNet50, que considera principalmente os aspectos visuais

da imagem e sua composição para classificar os itens (abordagem #1), contra uma arquitetura hibrida CNN e LSTM para fazer a descrição de imagem e posteriormente classificar o sentimento através da descrição via processamento de linguagem natural com o algoritmo Naive Bayes (abordagem #2). É importante considerar que na abordagem #2, também foi usado o modelo ResNet50 em sua composição, devido essa arquitetura já ser pré treinada para a biblioteca, além de ser um caso estudado na abordagem #1.

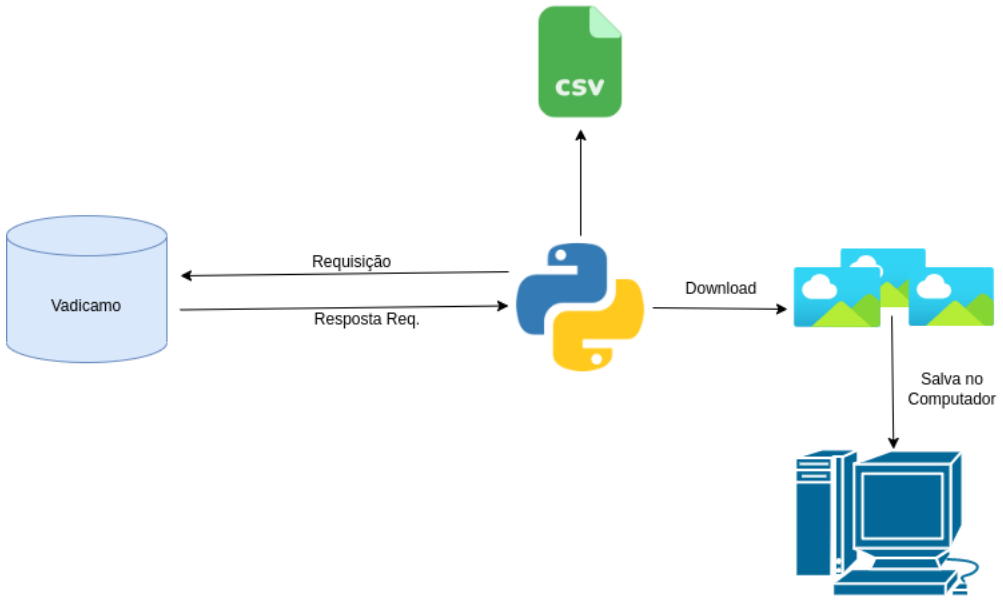
3.2 Preparação e tratamento dos *datasets*

Com o entendimento do problema proposto no projeto e a definição dos algoritmos de *Deep Learning* que seriam avaliados, a próxima etapa foi buscar por dados para as etapas de treinamento e teste dos algoritmos de classificação.

3.2.1 Dataset para abordagem #1

Após a identificação e definição das técnicas que seriam utilizadas para implementação, foi necessário procurar dados para serem usados nas fases de treinamento e testes dos algoritmos de classificação. Para isso, foram usadas bases de dados já existentes presentes na internet. As referências dos arquivos de imagem foram disponibilizados em arquivos CSV (*Comma Separeted Values*) onde foram extraídos usando *scripts* em Python para download dos mesmo para máquina onde seria realizado os treinamentos. O fluxograma da coleta dos dados da abordagem #1 esta descrito na Figura 16.

Figura 16 – Diagrama da esquema feito para criação do *dataset* final de treinamento e validação dos modelos



Fonte: Autoria própria

Para aumentar a confiabilidade do modelo durante o treinamento, foi reunido um conjunto de imagens nos diferentes contextos de sentimentos (Negativo e Positivo). Para reunir o *dataset* final, foi usado o conjunto de dados abertos fornecidos por (VADICAMO et al., 2017). Ao todo, o conjunto de dados coletados continha ao total cerca de 60 GB de imagens, entre as diferentes *labels* arquitetadas para o treinamento. Este *dataset* chamou a atenção pela forma organizada.

Durante o processo, os autores responsáveis pela elaboração usaram o modelo classificador de sentimento italiano proposto por (CIMINO; DELL'ORLETTA, 2016), mas usando a língua inglesa como base. Para criar a base de dados, os autores reuniram cerca de 4 milhões de twitters, onde foram classificados com uma mistura hibrida de uma rede neural LSTM e um classificador SVM, onde o LSTM captura as dependências de longo termo nas sentenças e o SVM entra com a classificação em uma das classes disponíveis (positivo, neutro, negativo, altamente negativo e altamente positivo).

Como não havia necessidade de usar todas as imagens para o treinamento dessa abordagem, foram separado ao todo 18 mil itens das 200 mil imagens disponíveis, escolhidas de forma aleatória igualmente para ambas categorias (Negativo e Positivo).Também é importante citar que foi usado o algoritmo K-Fold com o número de 8 *folds* para validação dos dados de uma forma acurada, onde a cada iteração e divisão entre os diferentes *folds*, foi necessário usar o método de balanceamento dos dados. Para isso foi usado a biblioteca "imbalanced learn" do Python com a função "RandomUnderSampler" na base de treino de cada passo do algoritmo, sendo este responsável por verificar a quantidade atual de cada *label* e deixá-las com o mesmo valor do conjunto com o menor número dentro da amostra.

Dado que o trabalho atuou com a classificação de imagens, foi necessário transformar os arquivos trabalhados de modo que o algoritmo entendesse o formato dos dados trabalhado. Desse modo, foi usado uma função chamada "ImageDataGenerator" presente na biblioteca *Keras*, onde sua funcionalidade principal é mudança no formato das imagens para um mais claro e padronizado, de forma que os algoritmos não tenham discrepância de valores durante o treinamento. Para este caso o trecho de código da Figura 17 importa os dados do diretório de origem, e em seguida transforma os arquivos que serão alimentados pela rede neural para o formato 224×224. Logo, todas as imagens coletadas serão transformadas para um tamanho adequado pelo arquiteto da solução, de forma que todo o *dataset* seja igualado.

3.2.2 Dataset para abordagem #2

Para a abordagem de classificação #2, foi necessário usar outro conjunto de dados para treinar o modelo de descrição de imagem para texto, diferente do modelo de classificação da abordagem #1. Durante algumas pesquisas, o *dataset* usado no trabalho

Figura 17 – Exemplo de uma função *generator*, responsável por balancear e dividir o *dataset* de forma padronizada

```
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datasetdir = '/tf/notebooks/image_classifier/dataset'
os.chdir(datasetdir)

batch_size = 15
```

```
def generators(shape, preprocessing):
    '''Create the training and validation datasets for
    a given image shape.
    '''
    imgdatagen = ImageDataGenerator(
        preprocessing_function = preprocessing,
        horizontal_flip = True,
        validation_split = 0.1,
    )

    height, width = shape

    train_dataset = imgdatagen.flow_from_directory(
        os.getcwd(),
        target_size = (height, width),
        classes = tuple(labels),
        batch_size = batch_size,
        subset = 'training',
    )

    val_dataset = imgdatagen.flow_from_directory(
        os.getcwd(),
        target_size = (height, width),
        classes = tuple(labels),
        batch_size = batch_size,
        subset = 'validation'
    )

    return train_dataset, val_dataset
```

```
train_dataset, val_dataset = generators((224,224), preprocessing=preprocess_input)
```

Fonte: Autoria própria

proposto por (HODOSH; YOUNG; HOCKENMAIER, 2013) chamou atenção por sua qualidade de dados e o fato da Universidade de Illinois deixar seus dados totalmente abertos, motivo em que ajudou este ser uma das escolhas usadas nos modelos deste trabalho ao primeiro momento, mas seu uso não trouxe resultados satisfatórios para as medições, o que motivou na sua troca por outro mais adequado.

Como a abordagem desenvolvida relaciona principalmente a classificação de sentimentos, o *dataset* que continha maior compatibilidade para trazer descrições sentimentais foi criado por (MATHEWS; XIE; HE, 2016), onde as descrições treinadas continham exclusivamente adjetivos positivos e negativos que podiam ser peças importantes no processo de categorização, como é possível visualizar na Figura 18. Para a criação desse *dataset*, os autores usaram a técnica de *crowdsorcing* junto a plataforma Amazon Mechanical Turk (AMT), o que ajudou os autores a enaltecer os pares de adjetivo/nome (ANPs) e através do SentiBank, ou seja, uma ontologia para reconhecimento de sentimentos em imagens, foi possível escolher a melhor descrição para as diversas imagens, mesmo que essas tenham mais de uma descrição envolvendo classes diferentes.

Figura 18 – Exemplos de descrições de imagens presente no *dataset* (MATHEWS; XIE; HE, 2016)

Fonte: (MATHEWS; XIE; HE, 2016)

Como essa abordagem considera textos para validar as respectivas imagens e assim gerar posteriormente descrições para novas imagens no modelo, seu passo crucial foi o tratamento e limpeza desses dados no intuito da eliminação de ruído antes do treinamento. Esse passo envolveu, a remoção de acentos, padronização de *case sensitive*, remoção de descrições sem sentido léxico suficiente e claros para a imagem analisada. É importante ressaltar também que todo o processo de tratamento de imagens como entrada dos modelos, também foi usado nessa etapa do trabalho (subseção 3.2.1).

Além do *dataset* responsável pela descrição, foi necessário encontrar outro que se adequasse a categorização para as classes de sentimentos usando o texto gerado pela primeira etapa . Para essa tarefa foi usado os dados de *reviews* de filmes e séries do IMDB como conjunto de dados de treinamento, presente na plataforma Kaggle. Além disso, foi aplicado as mesmas etapas de limpeza de texto citadas nos parágrafos anteriores.

3.3 Treinamento e validação de resultados

Com os dados devidamente tratados, foi possível treinar os algoritmos de cada abordagem com os conjuntos coletados de acordo com os requisitos codificados. Para cada teste realizado, todos os resultados foram salvos em arquivos JSON, de modo a comparar posteriormente ambos os fluxos e assim predizer qual entre as duas abordagens tem os melhores resultados. Todos os passos descritos nesta seção estão detalhados no Capítulo 4.

Para executar os códigos foi necessário a preparação do ambiente de desenvolvimento para trazer facilidade durante a produção dos resultados. Todo os códigos foram executados no seguintes requisitos citados na Tabela 1.

Tabela 1 – Ambiente computacional para execução dos testes

Especificações do Ambiente	
Sistema Operacional	Ubuntu 18.04 (Linux)
Processador	Intel Core I5 6300HQ - Quad Core 2.3 GHz com Turbo Max até 3.2 GHz – Cache 6 MB
Memória RAM	16 GB DDR3L 1600 MHz
Placa de Vídeo	NVIDIA GeForce GTX 960M até 4GB de memória dedicada

Fonte: Autoria própria

É importante ressaltar que, para dar rapidez na configuração dos ambientes, foram usados *containers* Docker com todas as dependências pré configuradas da própria documentação do *Tensorflow*. Dessa forma não houve nenhuma discrepância nas dependências da máquina enquanto configurava, pois o ambiente estava totalmente isolado. Com o uso dos *containers* foi possível habilitar a GPU de uma forma automatizada, pois toda parte de ambiente do CUDA já estava englobado na *build* da imagem disponibilizada pelo *Tensorflow* no DockerHub, o que agilizou o processo. Além disso, foi possível usar o Jupyter Notebook como IDE de desenvolvimento no próprio *container*, o que melhorou bastante a experiência de criação de código.

4 Resultados

Esse capitulo tem por objetivo mostrar os resultados obtidos durante o desenvolvimento do trabalho. Nas primeiras seções desse capítulo serão abordados os resultados referente ao treinamento/testes das abordagens #1 e #2 e por fim a comparação de seus resultados.

4.1 Classificação usando a abordagem #1

4.1.1 Treinamento e testes

Para ajudar no desenvolvimento e criação dos algoritmos foi usado especificamente a biblioteca Keras, uma das bibliotecas manutenidas pelo Google para o Python, desenvolvida para abstrair algumas funções do Tensorflow que por si são difíceis de manipular em relação a código. Como a arquitetura ResNet50 é uma rede neural já implementa no ecossistema do Keras, foi possível usar a técnica de *transfer learning* para agilizar o treinamento do modelo durante o desenvolvimento. A técnica de *transfer learning* consiste em reaproveitar um treinamento já elaborado e retreina-lo especificamente para o problema que o arquiteto da solução precisa.

Com a ajuda do *transfer learning* junto a ResNet50, é possível realizar treinamentos mais focados para a solução e evitar que a Rede Neural tenha que aprender tudo do zero. No caso do problema deste trabalho, foi usado este recurso acoplado ao modelo multicamadas treinados com o conjunto de treinamento esperado, desse modo foi possível checar os primeiros resultados que a rede neural podia encontrar nas etapas seguintes. Apesar da ajuda do pré treino, os primeiros resultados não foram tão compreensivos, pois as arquiteturas usadas não tiveram bons resultados até a arquitetura final avaliada via K-Fold igual a 8, como é possível verificar a arquitetura inicial dos testes na Figura 19.

No treinamento dos modelos, foi usado um *dataset* menor e fixo de imagens do conjunto coletado, pois o conjunto continha muitas imagens, então foi necessário criar um *subdataset* onde seria usado fixamente por todos os modelos treinados durante o processo. Nas primeiras rodadas, o algoritmo chegava a resultados perto ou igual a 50%, o que significava que o algoritmo possivelmente perderia para a aleatoriedade, o que estimulou na procura de novas formas de arquitetura, além disso, mostrou que o uso de diferentes formas de *Pooling* e um número significante de camadas conectadas poderia trazer bons resultados para o problema.

É importante destacar que no decorrer dos testes realizados, alguns parâmetros atribuídos para os modelos resultaram em *overfitting* dos resultados, onde muitas vezes

Figura 19 – Primeira arquitetura desenvolvida em código da ResNet50 no trabalho

```
shape = (IMAGE_WIDTH, IMAGE_HEIGHT, CHANNELS)

resnet = ResNet50(include_top=False, weights='imagenet', input_shape=shape)
output = resnet.layers[-1].output
output = keras.layers.Flatten()(output)
restnet = Model(resnet.input, output)

for layer in restnet.layers:
    layer.trainable = False

resnet.summary()
```

```
x = keras.layers.Flatten()(restnet.output)
x = keras.layers.Dense(100, activation='relu')(x)
predictions = keras.layers.Dense(3, activation='softmax')(x)
full_model = keras.models.Model(inputs=resnet.input, outputs=predictions)
full_model.summary()
```

Fonte: Autoria própria

Tabela 2 – Tabela com o resultado das medições durante a coleta de resultado da classificação da ResNet50

Modelo	**Acurácia** (Média)	**F1-score** (Média)
Modelo 1	0.38	0.29
Modelo 2	0.50	0.44
Modelo 3	0.63	0.51

Fonte: Autoria própria

tinham os níveis de acurácia consideráveis durante o treinamento, mas nas etapas de validação os resultados não se encontravam aproximados as faixas de valores da etapa anterior. Então quando isso tendeu a acontecer, foi usado a função "EarlyStopping" do Keras para evitar que o modelo convergisse para o *overfitting* após algumas épocas. Além disso, também foram feitos alguns testes englobando a classe neutra, mas os resultados não foram satisfatórios ficando muito abaixo de 50% de acurácia nos treinamentos, por isso não foram citados neste trabalho, pois havia um excesso de falsos positivos tendendo a esta nova classe testada.

4.1.2 Melhor configuração encontrada

Após alguns testes e ajustes de parâmetros, foi possível achar uma solução intermediária para os treinamentos realizados para a abordagem #1, este presente na Figura 20 descrito detalhadamente em código. Para achar estes resultados a partir do modelo de *transfer learning* da ResNet, foi usado o formato com a técnica de *Polling* usando como base a média de *pixels* para identificar os valores principais do conjunto de saídas, seguida de uma arquitetura de multicamada com cinco conexões Dense, onde a partir de cada camada continha uma de *Dropout*. A camada de *Dropout* é responsável pela eliminação de alguns neurônios de forma aleatoriamente durante a etapa de treinamento,

etapa crucial no desenvolvimento que evitou o *overfitting* dos dados de aprendizado.

Figura 20 – Ótima arquitetura desenvolvida em código da ResNet50 no trabalho

```
def get_model():
    shape = (224, 224, CHANNELS)
    resnet = ResNet50(
        include_top=False,
        weights='imagenet',
        input_shape=shape
    )

    for layer in resnet.layers:
        layer.trainable = False

    x = resnet.output
    x = AveragePooling2D(pool_size=(4, 4))(x)
    x = Flatten()(x)
    x = Dense(units=512, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=256, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=256, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=128, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=128, activation='relu')(x)
    x = Dropout(0.5)(x)
    # Negative or Positive
    predictions = Dense(2, activation='softmax')(x)
    model = Model(inputs=resnet.input, outputs=predictions)
    return model
```

Fonte: Autoria própria

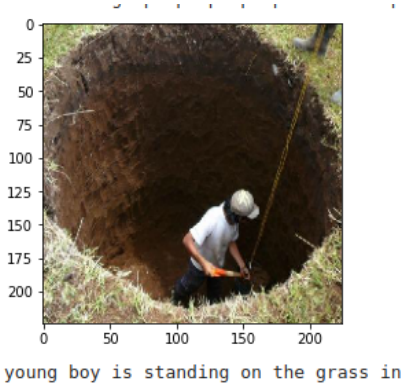
Até o momento, o melhor resultado encontrado para a abordagem #1 foi apresentado com uma acurácia de 63% com um total de 24883586 parâmetros calculados e treinados por época. O último modelo contou também com um F1 Score de 51% para um total de 8400 itens presentes no *dataset* de treinamento e balanceados usando o algoritmo de *Under Sampling* para as *labels* “Positivo” e “Negativo”, como é possível visualizar na tabela 2. Para o *dataset* de validação foram separado cerca 10% dos itens da base de treinamento para o cada *label* durante a etapa de pré-processamento.

4.2 Classificação usando a abordagem #2

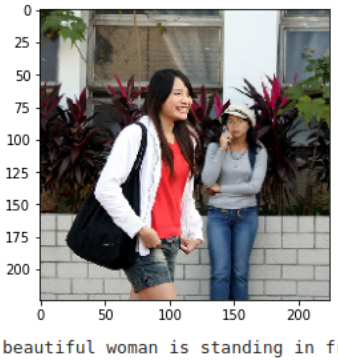
4.2.1 Rede neural híbrida para gerar descrição de imagem

Como mencionado anteriormente, o primeiro passo consistiu em descrever a imagem e isso foi criado a partir da iteração entre a rede neural ResNet50 com os itens identificados no escopo da figura e posteriormente a arquitetura LSTM usa como parâmetro os detalhes da primeira rede neural para gerar a descrição adequada, como é possível observar na Figura 21. Com a ajuda do *dataset* SentiBank usando descrições voltadas para sintaxes mais emocionais, foi possível encontrar o valor de métrica BLEU médio de 0,44 para o corpus de teste. Grande parte das imagens tinha sua descrição similar ao que estava sendo mostrado na imagem como é o caso da Figura 21. Também é importante destacar que algumas imagens de teste também não seguiram o resultado esperado, contendo descrições não condizentes, estas ilustradas na Figura 22.

Figura 21 – Imagens retiradas para teste e verificação depois do treinamento das legendas de imagens que foram condizentes com o que a imagem estava mostrando



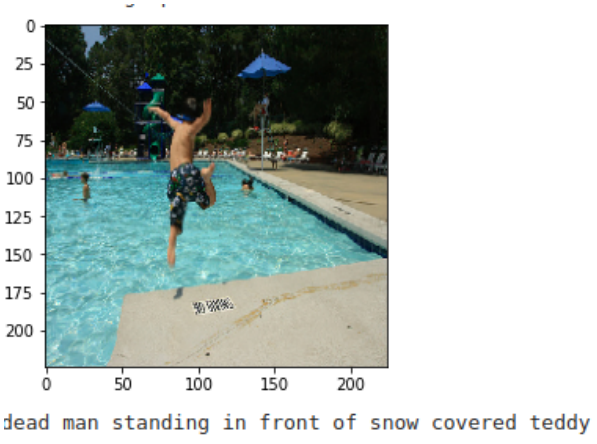
(a) Legenda de imagem após treinamento condizente.



(b) Legenda de imagem após treinamento condizente.

Fonte: Autoria própria

Figura 22 – Imagens retiradas para teste e verificação depois do treinamento da legenda de imagens que não foram condizentes com a composição imagem



(a) Legenda de imagem após treinamento no contexto errado.

(b) Legenda de imagem após treinamento no contexto errado.

Fonte: Autoria própria

Para o algoritmo Naive Bayes responsável por avaliar a descrição para a classe de sentimento designada, foram efetuados os treinamentos e testes obtendo o valor de acurácia de 86% durante o desenvolvimento, com uma taxa ótima para classificação entre ao que foi treinado.

Para avaliar os resultados da classificação após a descrição efetuada, foi usado o mesmo *dataset* da abordagem #1 com o algoritmo de K-Fold para ter uma avaliação mais precisa, já que são processos diferentes e separados em mais de uma etapa comparado com a abordagem #1. Desse modo, foi usado a iteração K-Fold com as mesmas 8 divisões, usando apenas a parte de teste de cada iteração para coleta dos resultados, de forma a assimilar o mesmo cenário da primeira abordagem com os mesmos dados de teste. A cada iteração, os dados obtidos foram guardados para calcular as métricas requeridas após o termino do processo de amostragem. Considerando a abordagem #2 como um todo, foi encontrado uma acurácia de 52% e um f-score de 45%, ou seja, é altamente provável jogar aleatoriamente e ter os mesmo valores do que usar o algoritmo pesquisado em um cenário real de classificação.

4.3 Comparação entre as abordagens #1 e #2

Com base nos experimentos realizados, foi possível verificar que a abordagem #1 se sobressaiu melhor sobre a outra abordagem com uma acurácia de 63%. Como durante todo o processo foi usado o mesmo *dataset* balanceado para treinamento, é possível comparar essa métrica contra os 52% de acurácia obtida para a abordagem #2. Logo a abordagem #1 foi melhor em relação à abordagem #2.

Apesar dos resultados encontrados serem medianos, foi encontrado alguns problemas que atrapalharam o trabalho desenvolvido. Como ambos os *datasets* coletados contem um número adequado de itens e variedades em seu meio, algumas das imagens encontradas estavam corrompidas. Dessa forma, no começo do desenvolvimento essa situação atrapalhou no andamento do trabalho, pois ao fazer o *parser* não eram identificados esses arquivos durante o treinamento. Após algumas validações, os dados dos *datasets* foram descobertos e tratados. Além deste empecilho, algumas descrições do conjunto de dados de (HODOSH; YOUNG; HOCKENMAIER, 2013) continham algumas sentenças faltando, o que foi necessário mais tratativas para melhorar a acurácia dos dados durante o processo de treinamento.

Durante o treinamento para geração de descrições das imagens, o uso do *dataset* criado por (HODOSH; YOUNG; HOCKENMAIER, 2013) não foi uma boa opção devido a suas características que não envolviam sentimento, desse modo trazendo inconsistências maiores para o que estava sendo analisado. O uso do *dataset* de (MATHEWS; XIE; HE, 2016) foi um grande divisor de águas para a segunda abordagem, pois trouxe a essência correta para validar a linha de raciocínio da abordagem #2 de uma forma mais apurada.

Outro problema encontrado durante as etapas de treinamento, foi a questão do ambiente. Nem sempre o modo GPU da máquina estava ativado dentro do *container*, devido a algumas incompatibilidades do *template* Docker configurados. Apesar da necessidade de mudança de estratégia para uma abordagem nativa no próprio sistema operacional, o ambiente criado atendia parcialmente e ajudava medianamente no levantamento de resultados. Mas no meio do desenvolvimento, esse empecilho foi mitigado e tornou-se uma importante ferramenta no processo de desenvolvimento, pois essa automatização evitou muitos problemas com as dependências do projeto.

5 Conclusão

5.1 Considerações Finais

Ao fim do trabalho desenvolvido, foram realizados treinamentos e testes com os modelos de *Deep Learning* para as diferentes abordagens estudadas (abordagem #1 e #2). Os resultados coletados durante o processo não foram ideias, mas foram suficiente para realizar as comparações e chegar no objetivo proposto inicialmente. Após alguns levantamentos inciais da banca avaliadora, novas propostas foram englobadas, assim apurando alternativas como validação cruzada e ajuste no padrão de coleta dos dados de treinamento, o que ajudou a trazer de forma mais acurada os dados finais apresentados. Apesar das declarações anteriores, houveram problemas de *overfitting* durante o treinamento dos modelos e problemas ao reconhecer a placa de vídeo pelo ambiente desenvolvido, além de problemas com o conteúdo de *Deep Learning* por falta de conhecimento, mas ambas foram contornadas durante o desenvolvimento do trabalho. É importante ressaltar que os objetivos especificados na seção 1.2 foram todos atingidos, mas com ressalvas, onde provavelmente as arquiteturas obtidas não conseguiram chegar ao seu máximo potencial por falta de tempo para novas experimentações.

A abordagem #1 obteve resultados mais satisfatórios em relação a abordagem #2. A segunda abordagem não superou a primeira devido a forma como as descrições geradas não seguiam uma linha muito tênue para destacar a qual classe o objeto de medição deveria ser classificado. Apesar da abordagem #2 ser inferior a #1, seus resultados foram encontrados com valores acima de 50%, o que para linha de base é um ponto positivo. Já a primeira abordagem se sobressaiu devido à simplicidade do seu fluxo e a ajuda de um modelo pré treinado que contribuiu na distinção do que era levantado durante os testes.

5.2 Trabalhos Futuros

Durante a realização do trabalho, foi possível trazer consigo grandes conhecimentos nas áreas de *Machine Learning*, especialmente em *Deep Learning* e com o processamento de linguagem natural, onde até o começo do trabalho não eram muito consolidados. Conhecimentos que ajudaram a comparar às duas ideias e achar uma resposta para os objetivos levantados. Apesar de não ter encontrado uma abordagem ideal para ambas alternativas, foi possível ver que o método visual se sobressaiu melhor comparado ao fluxo de classificação por descrições.

Para trabalhos futuros poderiam ser feitas novas medições para os casos levantados,

de modo a achar novas arquiteturas para obter resultados melhores, além disso, outras abordagens poderiam ser comparadas as propostas já descritas, dessa forma incrementando mais o *benchmark* medido. Além disso, testes envolvendo um ambiente sem a conteinerização poderia ser feito, de modo a verificar se o custo de virtualizar pode interferir diretamente nos resultados coletados.

Referências

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. [S.l.: s.n.], 2017. p. 1–6. Citado 2 vezes nas páginas 21 e 23.

ALSENAN, S.; AL-TURAIKI, I.; HAFEZ, A. A recurrent neural network model to predict blood–brain barrier permeability. *Computational Biology and Chemistry*, Elsevier, v. 89, p. 107377, 2020. Citado na página 25.

BEGUM, A.; FATIMA, F.; SABAHATH, A. Implementation of deep learning algorithm with perceptron using tenzorflow library. In: *2019 International Conference on Communication and Signal Processing (ICCSP)*. [S.l.: s.n.], 2019. p. 0172–0175. Citado 2 vezes nas páginas 19 e 20.

BOLLEN, J.; MAO, H.; PEPE, A. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *ArXiv*, abs/0911.1583, 2011. Citado na página 12.

BORTH, D. et al. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In: *Proceedings of the 21st ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2013. (MM '13), p. 223–232. ISBN 9781450324045. Disponível em: <https://doi.org/10.1145/2502081.2502282>. Citado 3 vezes nas páginas 12, 16 e 17.

CIMINO, A.; DELL'ORLETTA, F. Tandem lstm-svm approach for sentiment analysis. In: *CLiC-it/EVALITA*. [S.l.: s.n.], 2016. Citado na página 35.

DAI, W. et al. Transferring naive bayes classifiers for text classification. In: *AAAI*. [S.l.: s.n.], 2007. v. 7, p. 540–545. Citado na página 27.

GAJARLA, V.; GUPTA, A. Emotion detection and sentiment analysis of images. *Georgia Institute of Technology*, p. 1–4, 2015. Citado 3 vezes nas páginas 16, 18 e 32.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016. Citado na página 26.

HE, K. et al. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. Disponível em: <http://arxiv.org/abs/1512.03385>. Citado 3 vezes nas páginas 6, 23 e 24.

HODOSH, M.; YOUNG, P.; HOCKENMAIER, J. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, v. 47, p. 853–899, 05 2013. Citado 2 vezes nas páginas 36 e 44.

KIM, P. Deep learning. In: *MATLAB Deep Learning*. [S.l.]: Springer, 2017. p. 103–120. Citado na página 25.

KRIESEL, D. *A Brief Introduction to Neural Networks*. [s.n.], 2007. Disponível em: <availableathttp://www.dkriesel.com>. Citado na página 20.

KUMAR, A.; JAISWAL, A. Image sentiment analysis using convolutional neural network. In: SPRINGER. *International Conference on Intelligent Systems Design and Applications*. [S.l.], 2017. p. 464–473. Citado na página 16.

LECUN, Y. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 21.

LI, X. et al. Depression recognition using machine learning methods with different feature generation strategies. *Artificial Intelligence in Medicine*, v. 99, p. 101696, 2019. ISSN 0933-3657. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0933365719300296>. Citado na página 12.

LIN, C.-Y.; OCH, F. J. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In: *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. Geneva, Switzerland: COLING, 2004. p. 501–507. Disponível em: <https://www.aclweb.org/anthology/C04-1072>. Citado na página 28.

LOVE, B. C. Comparing supervised and unsupervised category learning. *Psychonomic bulletin & review*, Springer, v. 9, n. 4, p. 829–835, 2002. Citado na página 18.

MATHEWS, A.; XIE, L.; HE, X. Senticap: Generating image descriptions with sentiments. In: *AAAI*. [S.l.: s.n.], 2016. Citado 4 vezes nas páginas 6, 36, 37 e 44.

MEURERS, D. Natural language processing and language learning. *Encyclopedia of applied linguistics*, Wiley-Blackwell Oxford, p. 4193–4205, 2012. Citado na página 27.

MINAEE, S.; AZIMI, E.; ABDOLRASHIDI, A. Deep-sentiment: Sentiment analysis using ensemble of CNN and bi-lstm models. *CoRR*, abs/1904.04206, 2019. Disponível em: <http://arxiv.org/abs/1904.04206>. Citado na página 33.

MITTAL, N.; SHARMA, D.; JOSHI, M. L. Image sentiment analysis using deep learning. In: *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. [S.l.: s.n.], 2018. p. 684–687. Citado na página 17.

MOURA, G. B. d. et al. *Redes neurais recorrentes para a classificaçao de estruturas retóricas*. Dissertação (Mestrado) — Universidade Estadual de Maringá., 2018. Citado na página 25.

OLAH, C. *Understanding LSTM Networks*. 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Citado na página 25.

OLIVEIRA, D. J. S. et al. A aplicação da técnica de análise de sentimento em mídias sociais como instrumento para as práticas da gestão social em nível governamental. *Revista de Administração Pública*, v. 53, n. 1, p. 235–251, dez. 2018. Disponível em: <http://bibliotecadigital.fgv.br/ojs/index.php/rap/article/view/77807>. Citado na página 16.

ORTIS, A. et al. Exploiting objective text description of images for visual sentiment analysis. *Multimedia Tools and Applications*, Springer, p. 1–24, 2020. Citado na página 17.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015. Disponível em: <http://arxiv.org/abs/1511.08458>. Citado na página 22.

PATEL, M. *Detection of Maliciously Authored News Articles*. Tese (Doutorado) — Yüksek Lisans Tezi, The Cooper Union For The Advancement of Science and Art . . . , 2017. Citado na página 26.

Pixabay. *Figura Pixabay*. 2022. <https://pixabay.com/pt/photos/pessoas-faca-esfaqueamento-facada-315910/>. Online; acessado em 15 de Janeiro de 2022. Citado na página 14.

POWERS, D. M. W. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *CoRR*, abs/2010.16061, 2020. Disponível em: <https://arxiv.org/abs/2010.16061>. Citado 2 vezes nas páginas 29 e 30.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. In: _____. *Encyclopedia of Database Systems*. New York, NY: Springer New York, 2016. p. 1–7. ISBN 978-1-4899-7993-3. Disponível em: <https://doi.org/10.1007/978-1-4899-7993-3_565-2>. Citado 2 vezes nas páginas 30 e 31.

SANYAL, S. et al. Resume parser with natural language processing. *International Journal of Engineering Science*, v. 4484, 2017. Citado 2 vezes nas páginas 27 e 28.

SUTTON, R. S. Introduction: The challenge of reinforcement learning. In: *Reinforcement Learning*. [S.l.]: Springer, 1992. p. 1–3. Citado 2 vezes nas páginas 19 e 29.

TAHIR, R. et al. Bringing the kid back into youtube kids: Detecting inappropriate content on video streaming platforms. In: *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. [S.l.: s.n.], 2019. p. 464–469. Citado na página 12.

TRUONG, Q.-T.; LAUW, H. W. Visual sentiment analysis for review images with item-oriented and user-oriented cnn. In: *Proceedings of the 25th ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2017. (MM '17), p. 1274–1282. ISBN 9781450349062. Disponível em: <https://doi.org/10.1145/3123266.3123374>. Citado na página 12.

TRUONG, Q.-T.; LAUW, H. W. Visual sentiment analysis for review images with item-oriented and user-oriented cnn. In: *Proceedings of the 25th ACM international conference on Multimedia*. [S.l.: s.n.], 2017. p. 1274–1282. Citado na página 17.

VADICAMO, L. et al. Cross-media learning for image sentiment analysis in the wild. In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. [S.l.: s.n.], 2017. p. 308–317. Citado 2 vezes nas páginas 13 e 35.

VALLIANI, A. A.-A.; RANTI, D.; OERMANN, E. K. Deep learning and neurology: a systematic review. *Neurology and therapy*, Springer, v. 8, n. 2, p. 351–365, 2019. Citado na página 19.

WADHAWAN, A.; KUMAR, P. Deep learning-based sign language recognition system for static signs. *Neural Computing and Applications*, Springer, v. 32, n. 12, p. 7957–7968, 2020. Citado na página 22.

Apêndices

APÊNDICE A – Primeiro Apêndice

Figura 23 – Código para fazer download dos *datasets* e começo do tratamento de dados

Download dataset image

```
[ ] images_not_found = []
    # create main directory
    DATASET_MAIN_DIRECTORY = "dataset"
    if not os.path.exists(DATASET_MAIN_DIRECTORY):
        os.makedirs(DATASET_MAIN_DIRECTORY)

    # create label directories
    for label in y.unique():
        if not os.path.exists(f"{DATASET_MAIN_DIRECTORY}/{label}"):
            os.makedirs(
                f"{DATASET_MAIN_DIRECTORY}/{label}"
            )

    def download_image(kwargs):
        filename = kwargs['image_url'].split("/")[-1]
        try:
            urllib.request.urlretrieve(
                kwargs['image_url'],
                f"{DATASET_MAIN_DIRECTORY}/{kwargs['label']}/{filename}"
            )
        except urllib.error.HTTPError:
            print(f"Not Found: {url}")
            images_not_found.append(url)

    with concurrent.futures.ThreadPoolExecutor() as executor:
        #download images to directories
        for image_url, label_classified in zip(X, y):
            executor.submit(
                download_image,
                {
                    'image_url': image_url,
                    'label': label_classified
                }
        )

    with open('images_not_found.txt', 'w') as f:
        f.writelines(images_not_found)
```

Threat dataset from "http://www.t4sa.it/"

```
[ ] os.listdir("../b-t4sa_imgs")
```

```
[ ] labels = os.listdir("./dataset")
```

```
    label = {
        "0": "Negative",
        "1": "Neutral",
        "2": "Positive"
    }

    files = {
        "Negative": [],
        "Neutral": [],
        "Positive": []
    }

    with open('../b-t4sa_imgs/b-t4sa_all.txt') as f:
        lines = f.readlines()
        for line in lines:
            [path_img, classify] = line.replace("\n", "").split(" ")
            filename = path_img.split("/")[-1]
            shutil.move('../b-t4sa_imgs/' + path_img, f'./dataset/{label[classify]}/{filename}')
            files[label[classify]].append(filename)
```

Fonte: Autoria própria

Figura 24 – Ultima tentativa de criação e salvamento do modelo pós treinamento (abordagem #1)

```
def get_model():
    shape = (224, 224, CHANNELS)
    resnet = ResNet50(
        include_top=False,
        weights='imagenet',
        input_shape=shape
    )

    for layer in resnet.layers:
        layer.trainable = False

    x = resnet.output
    x = AveragePooling2D(pool_size=(4, 4))(x)
    x = Flatten()(x)
    x = Dense(units=512, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=256, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=256, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=128, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(units=128, activation='relu')(x)
    x = Dropout(0.5)(x)
    # Negative or Positive
    predictions = Dense(2, activation='softmax')(x)
    model = Model(inputs=resnet.input, outputs=predictions)
    return model
```

Fonte: Autoria própria

Figura 25 – Código responsável pela validação K-Fold na abordagem #1

```
fold_epoch = 8
for train, test in kfold.split(df_minify_X, df_minify_Y):

    main_df = df_dataset_minify.iloc[train]
    test_df = df_dataset_minify.iloc[test]
    gen_train, gen_valid = generators_by_dataframe(SHAPE, main_df)

    model = define_model_7th_try()

    model.compile(
        loss='categorical_crossentropy',
        optimizer=Adamax(learning_rate=0.0001),
        metrics=['accuracy']
    )

    history = model.fit(
        gen_train,
        validation_data = gen_valid,
        workers=10,
        epochs=50,
    )

    teste_df_transform = predict_images_transform(test_df)

    y_pred = model.predict(teste_df_transform)

    y_pred = transform_values_serie(y_pred)

    data_report = classification_report(
        y_pred=y_pred,
        y_true=test_df['class_target'],
        output_dict=True,
        zero_division=1
    )

    file_json = f'/tf/notebooks/image_classifier/KFold/report_fold_{fold_epoch}.json'
    fold_epoch += 1
    with open(file_json, 'w+') as f:
        json.dump(data_report, f)
        f.close()

    model.save(f'/tf/notebooks/image_classifier/Models/model-kfold-{fold_epoch}-epoch-try.h5')
```

Fonte: Autoria própria

Figura 26 – Código responsável pela limpeza dos dados textuais na etapa de treinamento para geração de descrição na abordagem descritiva.

```
id = df['image_label'].values
print('ID IMAGES SHAPE ', id.shape)

comment = df['comment']
print('COMMENT SHAPE', comment.shape)
```

```
ID IMAGES SHAPE  (8904,)
COMMENT SHAPE (8904,)
```

```
def sentence_cleaning(sentence):
    try:
        sentence = sentence.lower()
        sentence = re.sub('[^a-z]+', ' ', sentence)
        sentence = sentence.split()
        sentence = [word for word in sentence if len(word) > 1]
        sentence = ' '.join(sentence)
        return(sentence)
    except:
        return(sentence_cleaning('A dog runs a cross the grass .'))
```

```
vocab_dict = {}

for i in range(comment.shape[0]):
    if id[i] not in vocab_dict:
        vocab_dict[id[i]] = []
    sen = sentence_cleaning(comment[i])
    vocab_dict[id[i]].append(sen)

len(vocab_dict)
```

Fonte: Autoria própria