

UNIVERSIDADE FEEVALE

GUSTAVO ANDRÉ SETTI CASSEL

DETECÇÃO DE EXPRESSÕES FACIAIS COM INTELIGÊNCIA
ARTIFICIAL

Novo Hamburgo

2020

GUSTAVO ANDRÉ SETTI CASSEL

DETECÇÃO DE EXPRESSÕES FACIAIS COM INTELIGÊNCIA
ARTIFICIAL

Trabalho de Conclusão de Curso apresentado
como requisito parcial à obtenção do grau
de Bacharel em Ciência da Computação pela
Universidade Feevale

Orientador: MARTA ROSECLER BEZ

Novo Hamburgo

2020

GUSTAVO ANDRÉ SETTI CASSEL

DETECÇÃO DE EXPRESSÕES FACIAIS COM INTELIGÊNCIA
ARTIFICIAL

Trabalho de Conclusão de Curso apresentado
como requisito parcial à obtenção do grau
de Bacharel em Ciência da Computação pela
Universidade Feevale

APROVADO EM: ___ / ___ / _____

MARTA ROSECLER BEZ
Orientador – Feevale

JOÃO BATISTA MOSSMANN
Examinador interno – Feevale

PAULO RICARDO MUNIZ BARROS
Examinador interno – Feevale

Novo Hamburgo
2020

AGRADECIMENTOS

Gostaria de agradecer a todos que, de alguma forma, contribuíram para a realização desse Trabalho de Conclusão de Curso. Em especial:

Deus, pela saúde e paz para sempre seguir em frente;

Minha orientadora, Marta Rosecler Bez, por nortear meus estudos e ser uma amiga muito especial;

Minha irmã Brenda, que foi minha modelo para capturar fotos de expressões faciais;

Por último, mas não menos importante, meus pais e o querido Jairo, que me apoiaram emocionalmente e tiveram muita paciência comigo nos diferentes âmbitos da vida.

Gratidão!

RESUMO

Esta pesquisa tem como objetivo geral propor um mecanismo para detectar e classificar, com alta acurácia, as expressões faciais universais definidas por Ekman e Friesen. Atualmente, existem várias abordagens para resolver este problema, englobando técnicas de Processamento Digital de Imagens (PDI) e, recentemente, mecanismos de Inteligência Artificial. Serão avaliados os principais *datasets* para detecção de expressões e então escolhido aquele a ser usado no estudo proposto, além de comparar mecanismos já existentes de PDI e Inteligência Artificial para projetar o protótipo de software. O que justifica e motiva este estudo é a grande aplicabilidade que existe em detectar expressões faciais, desde inferir o sentimento do aluno em uma aula EAD até identificar o possível sentimento de um comprador ao manipular produtos nos corredores de um mercado. Desta forma, quanto melhor a acurácia em classificar expressões faciais, mais as áreas recém citadas se beneficiarão. A metodologia de pesquisa escolhida foi *Design Science Research* (DSR) que é propícia para integrar aspectos teóricos com o desenvolvimento de software. A literatura pesquisada teve como foco artigos, teses e dissertações dos anos de 2015 a 2020 em busca do estado da arte neste assunto. Alguns conceitos básicos de expressões faciais foram encontrados em redações antigas, como literaturas de Ekman datadas no fim dos anos 70. O resultado alcançado é um software com alta acurácia na detecção e classificação de expressões faciais definidas por Ekman e Friesen (neutra, raiva, nojo, medo, alegria, tristeza e surpresa), chegando à acurácia de 68,48% na base de dados de validação com o FER-2013 *Database*.

Palavras-chaves: Detecção de expressões faciais. Ekman e Friesen. Inteligência artificial. Redes neurais. Processamento digital de imagens.

ABSTRACT

This research aims to propose a mechanism to detect and classify, with high accuracy, the universal facial expressions defined by Ekman and Friesen. Currently, there are several approaches to solve this problem involving Digital Image Processing techniques and, recently, Artificial Intelligence mechanisms. The main datasets for expression detection will be evaluated and then the one to be used in the proposed study will be chosen, in addition to comparing existing Digital Image Processing and Artificial Intelligence mechanisms to design the software prototype. What justifies and motivates this study is the great applicability that exists in detecting facial expressions, from inferring the student's feeling in a distance learning course, to identifying the possible feeling of a buyer manipulating products in the aisles of a market. In this way, the better the accuracy in classifying facial expressions, the more the areas just mentioned will benefit. The research methodology chosen was Design Science Research (DSR) which is conducive to integrate theoretical aspects with software development. The researched literature focused on articles, theses and dissertations from the years 2015 to 2020 in search of the state of the art in this subject. Some basic concepts of facial expression were found in old essays, such as Ekman's literature dated in the late 1970s. The achieved result is a software with high accuracy in the detection and classification of facial expressions defined by Ekman and Friesen (neutral, anger, disgust, fear, joy, sadness and surprise), reaching an accuracy of 68.48% for the validation dataset with FER-2013 Database.

Key-words: Detection of facial expressions. Ekman and Friesen. Artificial intelligence. Neural networks. Digital image processing.

LISTA DE ILUSTRAÇÕES

Figura 1 – Caracterização de um artefato	15
Figura 2 – Lógica para construção de classes de problemas	16
Figura 3 – Proposta para condução de pesquisas usando DSR	17
Figura 4 – Fluxo metodológico para a presente pesquisa	19
Figura 5 – Expressões universais definidas por Ekman e Friesen	22
Figura 6 – Pontos FDP e FAP na face humana	24
Figura 7 – Pontos FDP e FAP nos olhos, nariz e boca	24
Figura 8 – Unidades FAPU na face humana	25
Figura 9 – Amostra da base de imagens JAFFE	26
Figura 10 – Amostra da base de imagens CK+	27
Figura 11 – Amostra da base de imagens AR	28
Figura 12 – Amostra da base de imagens FER-2013	28
Figura 13 – Representação de um neurônio biológico	30
Figura 14 – Representação de um neurônio artificial	32
Figura 15 – Representação gráfica da função binária	33
Figura 16 – Representação gráfica da função sinal	34
Figura 17 – Representação gráfica da função <i>Sigmóide</i>	34
Figura 18 – Representação gráfica da função tangente hiperbólica	35
Figura 19 – Representação gráfica da função ReLU	36
Figura 20 – Processo de aprendizagem supervisionado	37
Figura 21 – Processo de aprendizagem não-supervisionado	37
Figura 22 – Dígitos escritos à mão	38
Figura 23 – Representação de uma rede neural profunda	39
Figura 24 – Abstração de uma rede neural com neurônios <i>Sigmóide</i>	40
Figura 25 – Representação de uma rede neural <i>feed-forward</i>	40
Figura 26 – Representação de uma rede neural recorrente	41
Figura 27 – Relação entre redes convolucionais e demais áreas de estudo	42
Figura 28 – Funcionamento de uma rede convolucional	42
Figura 29 – <i>Kernel</i> de tamanho 2 x 2 para convolução	43
Figura 30 – Processo de convolução em uma CNN	43
Figura 31 – <i>Max pooling</i> em uma CNN	44
Figura 32 – <i>Average pooling</i> em uma CNN	45
Figura 33 – <i>Overfitting</i> durante o processo de treino	46
Figura 34 – Representação de <i>dropout</i> em uma rede neural	47
Figura 35 – Características principais de uma <i>Support Vector Machine</i>	47

Figura 36	– Rede <i>Perceptron</i> encontrando diferentes hiperplanos válidos	48
Figura 37	– Hiperplanos da rede <i>Perceptron</i> enquadrando incorretamente os dados .	49
Figura 38	– Arquitetura geral de uma CPU e de uma GPU	51
Figura 39	– Gráfico apresentando picos de humor no filme <i>Divertida Mente</i>	53
Figura 40	– Relação de rostos com níveis de expressão facial	53
Figura 41	– Visão geral da arquitetura estado da arte com FER-2013 <i>Database</i> . . .	54
Figura 42	– Camadas de <i>hardware</i> e <i>software</i> situando bibliotecas de IA	57
Figura 43	– Tempo para treinar a mesma rede neural em diferentes máquinas . . .	59
Figura 44	– Percentual de imagens da base de dados por categoria	60
Figura 45	– Estrutura de camadas da rede neural desenvolvida	63
Figura 46	– Gráficos de acurácia e custo (<i>loss</i>) da rede neural	65
Figura 47	– Matriz de Confusão para os dados de teste	66
Figura 48	– Imagens capturadas com uma modelo para verificar a eficácia da rede neural	67
Figura 49	– Extensão instalada no Visual Studio Code	68
Figura 50	– VSCode exibindo o TCC em formato LaTeX e também arquivos da nuvem	69
Figura 51	– VSCode exibindo documento em formato PDF diretamente da nuvem .	70

LISTA DE TABELAS

Tabela 1 – Definição de FAPUs	25
Tabela 2 – Bibliotecas para execução da rede neural	59
Tabela 3 – Resultados encontrados na base de validação sem <i>Image Augmentation</i>	61
Tabela 4 – Resultados encontrados na base de validação com <i>Image Augmentation</i>	62

LISTA DE ABREVIATURAS E SIGLAS

CNN	<i>Convolutional Neural Network</i>
CPU	<i>Central Processing Unit</i>
DBN	<i>Dynamic Bayesian Network</i>
DCNN	<i>Deep Convolutional Neural Network</i>
DRAM	<i>Dynamic Random Access Memory</i>
DSR	<i>Design Science Research</i>
FACS	<i>Facial Action Coding System</i>
FAP	<i>Face Animation Parameter</i>
FAPU	<i>Face Animation Parameter Units</i>
FDP	<i>Face Definition Parameter</i>
FVAE	<i>Factorized Variational Autoencoders</i>
GPU	<i>Graphics Processing Unit</i>
IA	Inteligência Artificial
LBP	<i>Local Binary Pattern</i>
LRN	<i>Local Response Normalization</i>
PDI	Processamento Digital de Imagens
ReLU	<i>Rectified Linear Unit</i>
RGB	<i>Red Green Blue</i>
SPAM	<i>Sending and Posting Advertisement in Mass</i>
SVM	<i>Support Vector Machine</i>

LISTA DE SÍMBOLOS

φ	Função de ativação
Σ	Somatório
w	Peso do neurônio artificial
b	Bias do neurônio artificial
y	Saída do neurônio artificial

SUMÁRIO

1	Introdução	13
2	Metodologia	15
2.1	<i>Design Science Research</i>	15
2.1.1	Artefatos	15
2.1.2	Classes de problemas	16
2.1.3	Etapas propostas pela <i>Design Science Research</i>	16
2.2	Fluxo metodológico para a presente pesquisa	19
3	Expressões faciais	22
3.1	Definições técnicas	22
3.2	<i>Datasets</i> com imagens de expressões faciais	26
3.2.1	JAFFE	26
3.2.2	CK+	27
3.2.3	AR <i>Face Database</i>	27
3.2.4	FER-2013 <i>Database</i>	28
4	Inteligência Artificial	30
4.1	<i>Perceptron</i>	31
4.2	Funções de ativação	33
4.2.1	Função binária	33
4.2.2	Função sinal	34
4.2.3	Função <i>Sigmóide</i>	34
4.2.4	Função tangente hiperbólica	35
4.2.5	Função ReLU	35
4.2.6	Função <i>Softmax</i>	36
4.3	Mecanismos de aprendizagem	36
4.3.1	Supervisionado	36
4.3.2	Não-supervisionado	37
4.4	Redes neurais profundas	37
4.4.1	Redes <i>feed-forward</i>	40
4.4.2	Redes recorrentes	41
4.5	Redes neurais convolucionais	41
4.5.1	Convolução	43
4.5.2	<i>Pooling</i>	44
4.5.3	<i>Overfitting</i>	46

4.5.4	<i>Dropout</i>	46
4.6	<i>Support Vector Machines</i>	47
5	Trabalhos correlatos	50
5.1	<i>Automatic Facial Expression Recognition Using DCNN</i>	50
5.2	<i>Factorized Variational Autoencoders for Modeling Audience Reactions to Movies</i>	52
5.3	<i>Local Learning with Deep and Handcrafted Features for Facial Expression Recognition</i>	53
5.4	Outros trabalhos	55
6	Software para reconhecimento de expressões faciais	56
6.1	Tecnologias	56
6.1.1	Python	56
6.1.2	TensorFlow e Keras	56
6.1.3	NumPy	57
6.1.4	Pandas	58
6.1.5	OpenCV	58
6.1.6	Preparação do ambiente	58
6.2	Base de dados utilizada	60
6.3	Redes neurais construídas e avaliadas	60
6.3.1	Arquitetura da rede neural com a melhor acurácia	62
6.3.2	<i>Callbacks</i> configurados para a rede neural	64
6.4	Resultados	64
7	Extensão desenvolvida para o Visual Studio Code	68
8	Conclusão	71
	Referências	73

1 INTRODUÇÃO

O reconhecimento de expressões faciais tem grande aplicabilidade no mundo moderno, podendo ser utilizado para beneficiar diversas áreas de conhecimento. Alhussein (2016), Amico *et al.* (2016) e Jaimes e Sebe (2007) apontam algumas das aplicações, como identificar o sentimento do aluno em uma aula EAD, a fim de que o professor tenha noção se a aula está ou não agradável; reconhecer o sentimento do consumidor enquanto este verifica e manipula os produtos de uma loja; detectar possíveis comportamentos suicidas a partir de expressões de tristeza; auxiliar o diagnóstico precoce de distúrbios psicológicos, com base nos movimentos da face; entre outras aplicações.

Nos últimos anos, a pesquisa avançou consideravelmente na área de reconhecimento de expressões faciais. Dentre os mecanismos utilizados para este fim, sabe-se que a Inteligência Artificial vem gradualmente se destacando, sendo usada para a classificação, extração de características e obtenção de padrões a partir de imagens do rosto humano. No momento em que novos mecanismos são propostos para o reconhecimento de expressões, ou mecanismos já existentes são aprimorados, diversas áreas se beneficiam com tais melhorias, enfatizando a importância da presente pesquisa.

A Inteligência Artificial pode ser entendida como um mecanismo de aprendizado de máquina onde o computador aprende de forma similar ao cérebro humano, passando por um processo prévio de treinamento. Neste processo, neurônios artificiais têm a responsabilidade de descobrir padrões e correlações nos dados de entrada, a fim de extrair suas características mais relevantes. O capítulo 4 apresentará maiores detalhes acerca de Inteligência Artificial.

Este trabalho tem como objetivo geral desenvolver e validar um protótipo de software para identificar, com alta acurácia, as expressões faciais básicas (neutra, raiva, nojo, medo, alegria, tristeza e surpresa) definidas por Ekman e Friesen (2003). Como objetivos específicos estão a escolha da linguagem de programação; escolha dos *datasets* com imagens do rosto humano, a serem usadas para treinar a rede neural proposta; projetar e desenvolver o protótipo de software em si, que detectará expressões faciais a partir dos *datasets* previamente selecionados; e, por fim, mas não menos importante, validar os resultados alcançados.

A metodologia utilizada para nortear a pesquisa é a *Design Science Research* (DSR). Esta metodologia tem como propósito colocar em prática a *Design Science*, conhecida como ciência do projeto ou ciência do artificial, que foca na solução de problemas através do uso e desenvolvimento de artefatos. O fluxo metodológico desta pesquisa científica é composto por quatro etapas principais. A primeira, Identificação do Problema,

reconhece a necessidade de aprimorar a detecção de expressões faciais; a segunda, Criação, é onde a pesquisa científica é de fato realizada e o artefato de software é desenvolvido; a terceira, Avaliação, é onde os resultados alcançados pelo software são validados e comparados com resultados alcançados por outros pesquisadores; por fim, a quarta, Disseminação, é onde a pesquisa científica e seus resultados são compartilhados com outras pessoas e organizações interessadas.

Os próximos capítulos retomarão alguns destes conceitos e aprofundarão mecanismos na área de detecção de expressões, sendo divididos da seguinte forma: no capítulo 2 é apresentada a metodologia *Design Science Research* e o fluxo metodológico que guia a presente pesquisa; o capítulo 3 apresenta conceitos e definições essenciais sobre as expressões do rosto humano, acompanhado de alguns dos principais *datasets* com imagens de expressões faciais; o capítulo 4 aborda o tema de Inteligência Artificial, apresentando desde conceitos iniciais, como *Perceptron*, até definições de redes neurais profundas e redes convolucionais; o capítulo 5 apresenta trabalhos correlatos com mecanismos propostos por outros pesquisadores no mesmo nicho de pesquisa, incluindo o estado da arte que obteve a acurácia de 75,42% com a base de dados FER-2013 *Database*; o capítulo 6 apresenta em detalhes o software desenvolvido na presente pesquisa científica, capaz de reconhecer expressões faciais com uso de Redes Neurais Convolucionais; o capítulo 7 apresenta uma extensão desenvolvida para o Visual Studio Code que integra o editor de programação com o Google Drive, sendo um subproduto da presente pesquisa científica; por fim, o capítulo 8 apresenta a conclusão e as considerações finais.

Expressões faciais não representam apenas fisionomia, mas uma forma de linguagem. Talvez, a mais primitiva e ordinária delas, mas ao mesmo tempo, a mais universal.

2 METODOLOGIA

A metodologia usada no desenvolvimento da presente pesquisa foi a *Design Science Research* (DSR). Baseada na *Design Science*, possui vários conceitos importantes que serão destacados no decorrer deste capítulo, assim como, será apresentado o fluxo metodológico construído especificamente para nortear o desenvolvimento desta pesquisa.

2.1 DESIGN SCIENCE RESEARCH

Para compreender a metodologia DSR é importante, primeiramente, ter conhecimento da ciência à qual esta se refere: a *Design Science*. De acordo com Dresch, Lacerda e Antunes (2015), também conhecida como ciência do projeto ou ciência do artificial, tem como propósito a solução de problemas, projetando e construindo novos sistemas ou aprimorando sistemas já existentes em busca de melhores resultados. O termo *artificial* é definido por Simon (1996) como tudo aquilo que não é natural, sendo tipicamente inventado ou desenvolvido pelos seres humanos.

2.1.1 Artefatos

A metodologia DSR é responsável por operacionalizar e colocar em prática a *Design Science*. Nesta metodologia, o conceito de artefato é definido por Dresch, Lacerda e Antunes (2015) como algo artificial que por si só não soluciona o problema, porém, pode auxiliar no processo de solução, interagindo com o ambiente externo e com o contexto em que o problema está situado. Algoritmos computacionais e softwares podem ser citados como exemplos de artefatos. A Figura 1 apresenta um artefato indicando sua relação com os ambientes interno e externo.

Figura 1 – Caracterização de um artefato



Fonte: Dresch, Lacerda e Antunes (2015) baseados em Lacerda *et al.* (2013)

2.1.2 Classes de problemas

Outro conceito importante da DSR é classe de problemas, representando um conjunto similar de situações, impasses e dificuldades tratadas por assuntos de um mesmo nicho de pesquisa. A solução proposta para um determinado problema pode servir de base para encontrar soluções para outros problemas da mesma classe. Desta forma, classe de problemas pode ser definida como "organização de um conjunto de problemas práticos ou teóricos que contenha artefatos úteis para a ação nas organizações" (DRESCH; LACERDA; ANTUNES, 2015, p. 104).

A Figura 2 destaca a lógica sugerida por Dresch, Lacerda e Antunes (2015) para construir e delimitar uma classe de problemas. O primeiro passo é a conscientização do problema e do contexto em que este se encontra, assim como, é o momento de definir quais objetivos o pesquisador precisa alcançar para que o problema seja solucionado. O passo seguinte é a revisão sistemática da literatura, onde o pesquisador busca, em diferentes bases de dados, pesquisas que forneçam o embasamento necessário para solucionar o problema. Em um terceiro momento, a partir da revisão sistemática, são identificados os artefatos capazes de oferecer solução ao problema abordado. Por fim, definidos os artefatos, é possível enquadrá-los dentro uma mesma classe de problemas.

Figura 2 – Lógica para construção de classes de problemas

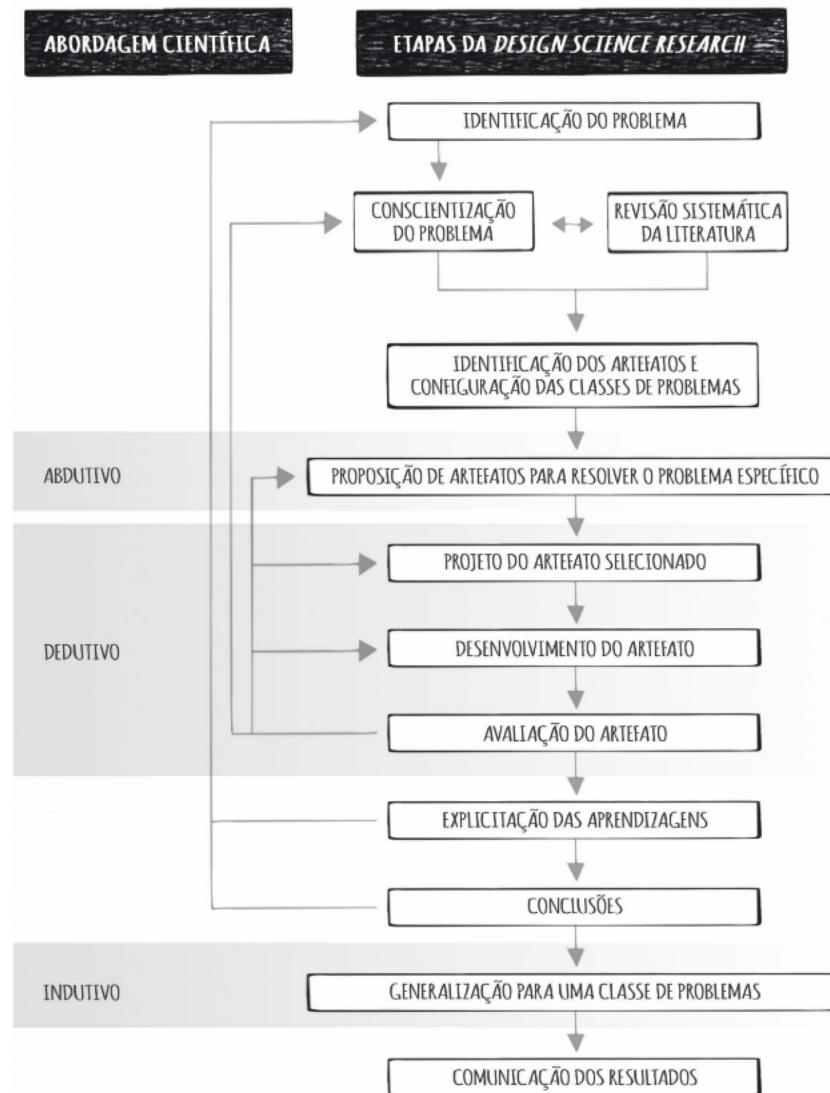


Fonte: Dresch, Lacerda e Antunes (2015) baseados em Lacerda *et al.* (2013)

2.1.3 Etapas propostas pela *Design Science Research*

Dresch, Lacerda e Antunes (2015) apresentam esta metodologia com doze etapas principais que o pesquisador deve seguir para alcançar os resultados esperados, conforme pode ser observado na Figura 3. Tais etapas são apresentadas a seguir:

Figura 3 – Proposta para condução de pesquisas usando DSR



Fonte: Dresch, Lacerda e Antunes (2015)

- **Identificação do problema:** etapa na qual o problema é de fato detectado e reconhecido, sendo também o momento em que a justificativa e os requisitos são redigidos para o estudo proposto.
- **Conscientização do problema:** momento em que o pesquisador formaliza as faces do problema e toma real ciência do contexto em que o mesmo está inserido, sendo um aprofundamento do conhecimento a respeito do problema.
- **Revisão sistemática da literatura:** complementa a etapa anterior que trata da conscientização do problema, visto que através da revisão sistemática o maior número possível de informações acerca do problema é identificado em literaturas já existentes e publicadas por outros pesquisadores.

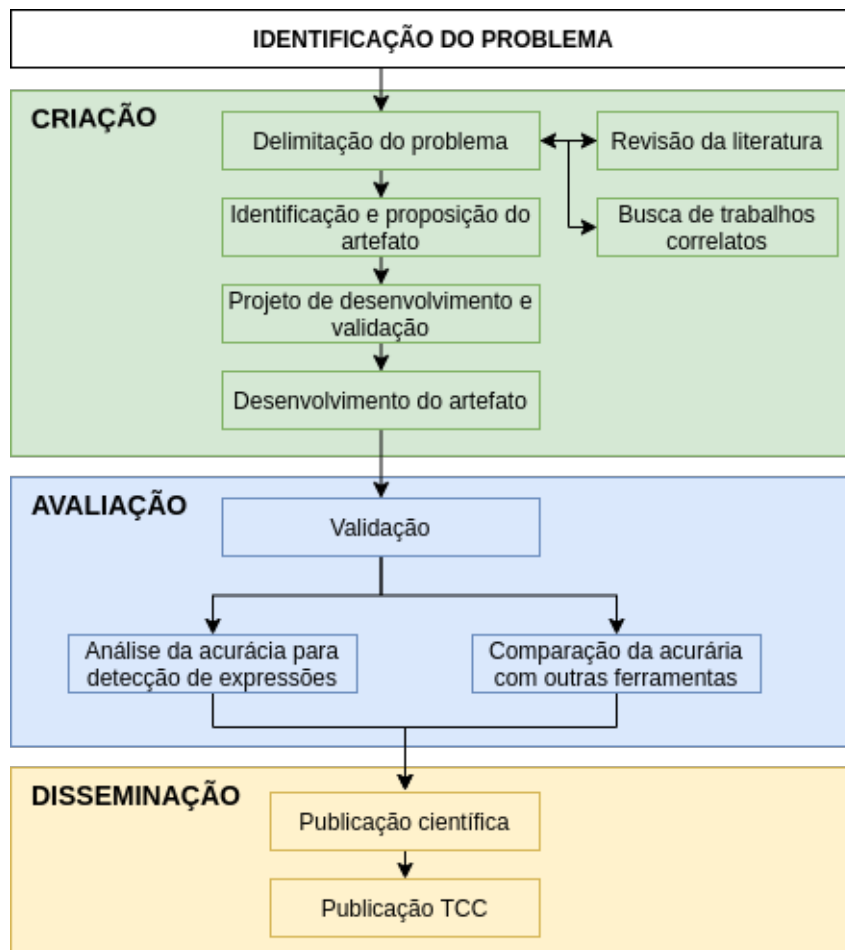
- **Identificação dos artefatos e configuração das classes de problemas:** a busca por trabalhos correlatos que sustentam a pesquisa pode, além de situar o pesquisador no contexto e na conscientização do problema, também ser útil para identificar artefatos já existentes e que podem auxiliar na resolução do problema. Além disso, também situa o problema de pesquisa dentro de possíveis classes de problemas já existentes.
- **Proposição de artefatos para resolver o problema específico:** nesta etapa o pesquisador propõe artefatos próprios para resolver seu problema específico. Muitas vezes, os artefatos encontrados na literatura servem para resolver problemas genéricos, não sendo suficientes para o problema abordado pelo pesquisador.
- **Projeto do artefato selecionado:** uma vez que artefatos já foram propostos na etapa anterior, um deles deve ser selecionado, projetado e analisado mais a fundo. Este artefato será levado adiante nas próximas etapas da pesquisa.
- **Desenvolvimento do artefato:** momento onde o artefato propriamente dito é desenvolvido, a fim de solucionar o problema abordado pelo pesquisador. O artefato pode ser um algoritmo computacional, um software, um protótipo de software, entre outros.
- **Avaliação do artefato:** na etapa de avaliação o pesquisador compara, com os resultados previstos na etapa de identificação do problema, os resultados alcançados pelo artefato desenvolvido. Caso o artefato não atinja os objetivos previstos, pode-se voltar para etapas anteriores e determinar quais pontos precisam ser aprimorados, até que a necessidade inicial seja suficientemente atendida.
- **Explicitação das aprendizagens:** neste momento o pesquisador apresenta o que foi aprendido com o desenvolvimento dos artefatos, destacando principalmente o que deu certo e o que não deu certo (pontos de sucesso e insucesso). Isto serve como base para pesquisas futuras, permitindo que pesquisadores tenham noção do que possui maior ou menor chance de funcionar.
- **Conclusões:** etapa onde o pesquisador aponta os resultados alcançados por seu artefato e sua pesquisa, complementando a explicitação das aprendizagens. Também é o momento de apresentar possíveis limitações e sugerir trabalhos futuros neste nicho de pesquisa.
- **Generalização para uma classe de problemas:** trata do levantamento de heurísticas para enquadrar e generalizar, em uma classe de problemas, o artefato desenvolvido, contribuindo para o avanço de pesquisas futuras em áreas relacionadas.

- **Comunicação dos resultados:** é a etapa final da DSR, onde a pesquisa é publicada em congressos, *journals*, seminários, entre outros, disseminando o conhecimento com pessoas e organizações interessadas no assunto.

2.2 FLUXO METODOLÓGICO PARA A PRESENTE PESQUISA

Baseado na proposta de Farias (2019), a metodologia para elaborar esta pesquisa científica foi feita com princípios da DSR. No entanto, algumas etapas foram adaptadas e abstraídas para simplificar o diagrama que contém o fluxo metodológico, o qual é apresentado na Figura 4, seguido da definição de cada uma de suas principais etapas.

Figura 4 – Fluxo metodológico para a presente pesquisa



Fonte: do autor, baseado em Farias (2019)

A primeira etapa, nomeada Identificação do Problema, resume-se a reconhecer a necessidade de aprimorar mecanismos que detectam expressões faciais, uma vez que diversas áreas podem se beneficiar de tal pesquisa e de seus resultados. Esta primeira etapa é tipicamente menos complexa do que as seguintes. Neste momento, elementos iniciais como justificativa e objetivos são levantados.

Após a identificação do problema é iniciada a segunda etapa do fluxo metodológico, denominada Criação. Nesta etapa, a pesquisa científica é de fato realizada e o artefato de software é proposto e desenvolvido. Esta etapa engloba desde a delimitação do escopo do problema, através da busca por trabalhos correlatos e revisão da literatura já existente no nicho de pesquisa, até a proposta e o desenvolvimento do artefato de software. Pode-se destacar quatro passos principais envolvidos por esta etapa:

- **Delimitação do problema:** passo onde o escopo da pesquisa é lapidado através da revisão da literatura já existente e da busca por trabalhos correlatos. Como motor de busca é utilizado o *Web of Science*, consolidando artigos e materiais provenientes de diferentes bases de dados, a fim de realizar uma revisão no tema de detecção de expressões faciais. A revisão da literatura está disponível no capítulo 4, enquanto alguns dos principais trabalhos correlatos podem ser encontrados no capítulo 5.
- **Identificação e proposição do artefato:** uma vez realizada a delimitação do problema, é possível identificar artefatos desenvolvidos por outros pesquisadores neste nicho de pesquisa. Também é o momento de propor um ou mais artefatos para a pesquisa em questão, ou seja, definir o que exatamente será desenvolvido.
- **Projeto de desenvolvimento e validação:** dados os artefatos propostos no passo anterior, um deles é selecionado para dar sequência na pesquisa. É definida pelo menos uma base de dados (*dataset*) com imagens de expressões faciais para avaliar os resultados do software assim que o mesmo for desenvolvido. Também é o momento de escolher a linguagem de programação na qual o software será implementado, além de definir como será desenvolvido. As seções 6.1 e 6.2 apresentam, respectivamente, as tecnologias selecionadas para a construção do software e a base de dados utilizada para os processos de treinamento, validação e testes.
- **Desenvolvimento do artefato:** momento em que o artefato (software) é de fato implementado. No contexto do presente Trabalho de Conclusão de Curso, um software que detecte expressões faciais através de Inteligência Artificial, capaz de reconhecer as expressões faciais básicas (neutra, raiva, nojo, medo, alegria, tristeza e surpresa) definidas por Ekman e Friesen (2003). A seção 6.3 apresenta uma visão geral das diversas arquiteturas construídas.

Uma vez desenvolvido o software, dá-se início à etapa de Avaliação, onde seus resultados e sua acurácia são mensurados com o uso de um ou mais *datasets* previamente selecionados. Além disso, a avaliação também é feita comparando os números encontrados pelo software desenvolvido, em relação aos números alcançados por ferramentas e mecanismos propostos por outros pesquisadores. A seção 6.4 apresenta em detalhes os re-

sultados alcançados pela rede neural com a melhor acurácia dentre todas as arquiteturas projetadas, testadas e avaliadas.

Por fim, mas não menos importante, chega-se à etapa de Disseminação, onde os resultados são compartilhados com a comunidade científica. Na Publicação Científica, os resultados são apresentados e compartilhados na Feira de Iniciação Científica (FIC) 2020, da Universidade Feevale. Em um segundo momento será realizada a entrega e a apresentação do presente Trabalho de Conclusão de Curso, o qual ficará disponível na biblioteca da Universidade Feevale e no site do TC Online após passar pela banca examinadora. Almeja-se, ainda, encaminhar artigos referentes ao processo do TCC em congressos ou revistas especializadas.

No capítulo atual foram apresentados diversos conceitos sobre a metodologia *Design Science Research*, seguido do fluxo metodológico usado na presente pesquisa científica. O capítulo seguinte abordará conceitos de expressões faciais, evidenciando várias definições propostas por Ekman e Friesen (2003). Por fim, também apresentará alguns dos principais *datasets* com imagens de expressões faciais, os quais são utilizados para validar a acurácia de softwares voltados à detecção de expressões faciais.

3 EXPRESSÕES FACIAIS

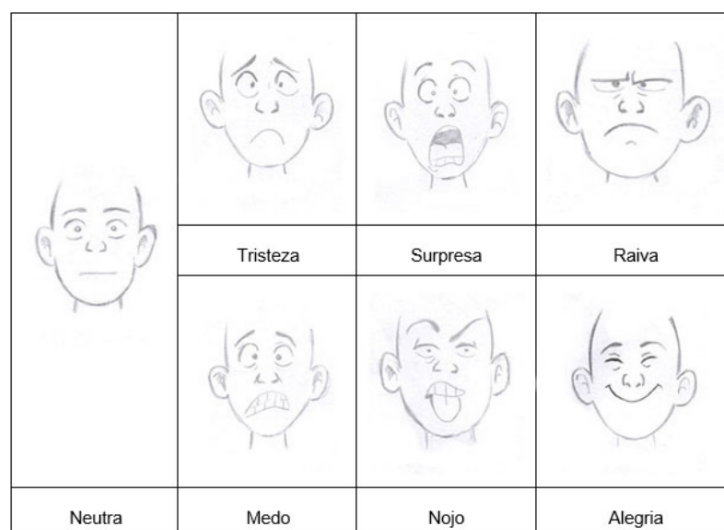
Ekman e Friesen (2003) apontam que as expressões faciais são elementos de extrema importância para a comunicação humana. Através delas, as pessoas conseguem exteriorizar seu estado emocional não somente com o uso de palavras, mas também através da fisionomia, representando uma forma de comunicação não-verbal.

O presente capítulo abordará as expressões faciais universais propostas por Ekman e Friesen, assim como elementos principais do rosto humano e termos técnicos desta área de estudo. Também serão apresentadas algumas das especificações existentes, como o *Facial Action Coding System*, seguidas de bases de dados com imagens para treinar sistemas de detecção de expressões faciais.

3.1 DEFINIÇÕES TÉCNICAS

As emoções são tipicamente identificadas pelo rosto através das expressões faciais, seguidas do restante do corpo, o qual expõe a maneira propriamente dita como as pessoas lidam com suas emoções. O sentimento de medo pode ser usado como exemplo: não há um movimento característico no corpo que sempre represente o medo, uma vez que as reações podem ser diversas e variar de pessoa para pessoa. No entanto, a expressão facial possui traços característicos que possibilitam perceber este sentimento com maior facilidade (EKMAN; FRIESEN, 2003).

Figura 5 – Expressões universais definidas por Ekman e Friesen



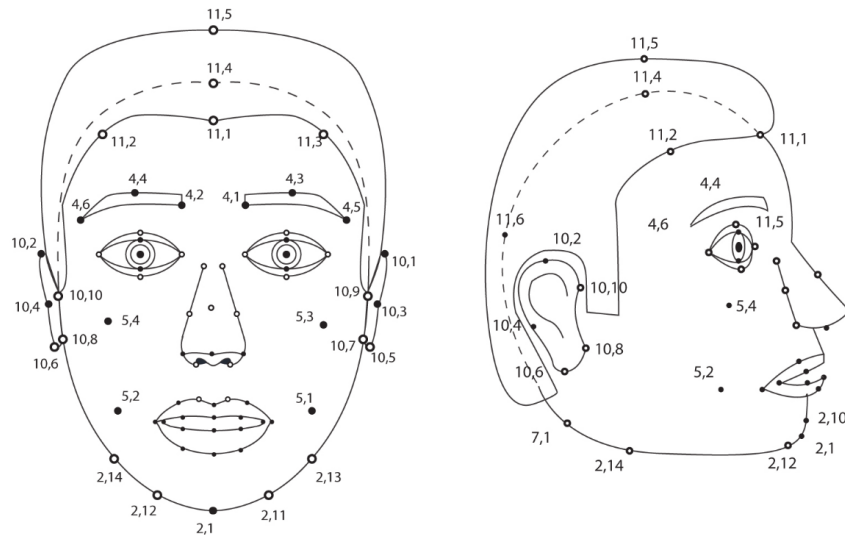
Fonte: Bez *et al.* (2017)

Ao mesmo tempo em que o reconhecimento das expressões ocorre de forma natural e automática pelo cérebro, a descrição detalhada dos movimentos, traços e característi-

cas inerentes a cada expressão pode ser uma tarefa complexa de realizar. Diante desta realidade, nos anos 70, Ekman e Friesen propuseram o *Facial Action Coding System* - FACS - definindo as principais feições do rosto humano. São categorizadas por eles as expressões universais neutra, raiva, nojo, medo, alegria, tristeza e surpresa, as quais são comuns e inerentes aos seres humanos de qualquer lugar do mundo e não possuem relação com culturas específicas (BEZ *et al.*, 2017) (EKMAN; OSTER, 1979) (LLC, 2020). De acordo com Bez *et al.* (2017), a Figura 5 apresenta as expressões universais definidas por Ekman e Friesen enquanto a descrição de cada uma delas é detalhada abaixo:

- **Neutra:** a expressão facial neutra é representada quando a face não indica nenhum sentimento nem reação específica e está em sua normalidade. É definida por sinais como boca fechada, cabeça reta e músculos relaxados.
- **Raiva:** na expressão de raiva o indivíduo tende a franzir a testa ao mesmo tempo em que o nariz é pressionado. Algumas vezes os dentes aparecem e os olhos se retraem no centro.
- **Nojo:** na expressão de nojo a testa e o nariz são tipicamente franzidos, formando rugas, que também podem ser percebidas em outras localidades do rosto. A ponta do nariz é contraída e se eleva.
- **Medo:** a expressão de medo tem como característica principal a elevação das sobrancelhas juntamente com a abertura dos olhos. Os lábios também tendem a se separar, assim como as fossas nasais se expandem e a parte superior do nariz se afina.
- **Alegria:** a alegria é tipicamente expressada pela boca esticada em formato de sorriso, normalmente com dentes expostos, ao mesmo tempo em que a distância entre a boca e o nariz diminui. Paralelo a isso, a bochecha também fica em evidência e a parte visível dos olhos tende a diminuir de tamanho pois estes são pressionados, formando algumas rugas.
- **Tristeza:** na expressão de tristeza a boca se retrai e seus cantos são muitas vezes curvados para baixo. Os dentes tendem a não aparecer e pode ser exibida uma demarcação entre o lábio inferior e o queixo. A parte interna das sobrancelhas se aproxima e as pálpebras caem.
- **Surpresa:** a expressão de surpresa é normalmente acompanhada de lábios afastados, mantendo a boca aberta e as sobrancelhas elevadas. Nesta expressão também é comum que os globos oculares fiquem mais visíveis e expostos.

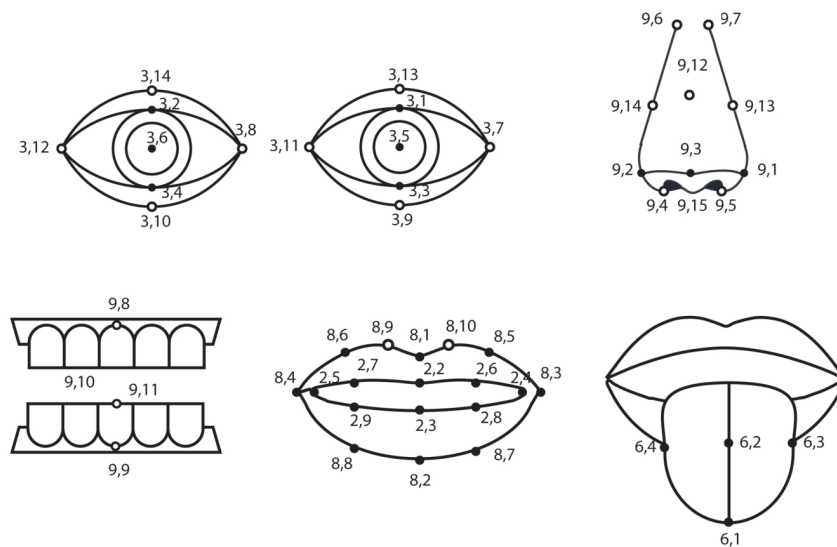
Figura 6 – Pontos FDP e FAP na face humana



Fonte: ISO/IEC JTC 1/SC 29/WG 11 (2001)

O *Facial Action Coding System* estabelece uma série de elementos, pontos chave e localizações estratégicas na face para que seja possível determinar a expressão que mais se enquadra ao rosto analisado. Estes elementos são categorizados em FDP (*Face Definition Parameter*), FAP (*Face Animation Parameter*) e FAPU (*Face Animation Parameter Units*) e podem ser vistos nas Figuras 6 e 7. Os pontos não preenchidos são os FDP, ou seja, aqueles que a face humana tipicamente não consegue movimentar. Já os pontos pretos/preenchidos são os FAP, que podem ser movimentados pelo rosto, sendo um subconjunto dos pontos FDP (ISO/IEC JTC 1/SC 29/WG 11, 2001).

Figura 7 – Pontos FDP e FAP nos olhos, nariz e boca



Fonte: ISO/IEC JTC 1/SC 29/WG 11 (2001)

Sabe-se também que cada ser humano apresenta características próprias em seu

rosto e que a disposição de seus principais elementos, como olhos, nariz, boca, orelhas, etc., pode variar de pessoa para pessoa. Pode-se supor, a título de exemplo, que para uma pessoa a distância entre os olhos seja de 1 centímetro enquanto para outra seja menor ou maior do que isto, como 9 ou 11 milímetros. Diante desta realidade são necessárias unidades de medida variáveis entre os principais pontos do rosto, conhecidas como *Face Animation Parameter Units* (FAPU) (BEZ *et al.*, 2017) (ISO/IEC JTC 1/SC 29/WG 11, 2001).

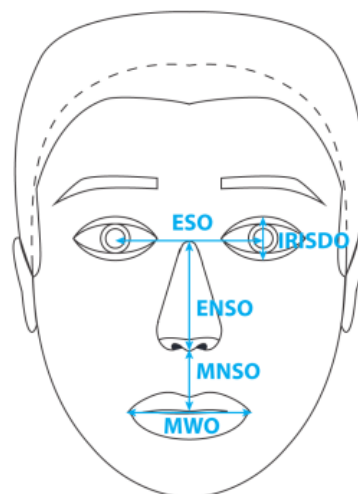
Tabela 1 – Definição de FAPUs

FAPU	FDPs	Descrição	Valor FAPU
IRISD	3.1.y – 3.3.y = 3.2.y – 3.4.y	Diâmetro da íris	$IRISD = IRISD0 / 1024$
ES	3.5.x – 3.6.x	Separação dos olhos	$ES = ES0 / 1024$
ENS	3.5.y – 9.15.y	Separação entre olhos e nariz	$ENS = ENS0 / 1024$
MNS	9.15.y – 2.2.y	Separação entre boca e nariz	$MNS = MNS0 / 1024$
MW	8.3.x – 8.4.x	Largura da boca	$MW = MW0 / 1024$
AU		Unidade angular	$AU = 10^{-5} \text{ rad}$

Fonte: ISO/IEC JTC 1/SC 29/WG 11 (2001)

Existe um total de 6 unidades FAPU no rosto humano. Dentre elas pode ser citada como exemplo a unidade ES indicando a separação dos olhos, sendo possível inferir que aproximadamente na metade desta distância se encontra o topo do nariz, conforme apresentado na Figura 8. A Tabela 1 apresenta as unidades FAPU definidas pelo *Facial Action Coding System*, enquanto a Figura 8 aponta visualmente tais unidades em um rosto humano (ISO/IEC JTC 1/SC 29/WG 11, 2001).

Figura 8 – Unidades FAPU na face humana



Fonte: ISO/IEC JTC 1/SC 29/WG 11 (2001)

De acordo com Wolf (2009) e Tian, Kanade e Cohn (2005), a detecção de expressões faciais é tipicamente feita com mecanismos que exploram a geometria do rosto ou com soluções baseadas em sua aparência. Os mecanismos de geometria buscam curvas, bordas e pontos característicos de cada elemento do rosto, como olhos, nariz e boca, para delimitá-los e contextualizá-los na conjuntura da face. A fim de realizar tal tarefa são usados pontos FDP, FAP e unidades FAPU definidos pelo *Facial Action Coding System* (FACS). Já os recursos de aparência vem nos últimos anos se sobressaindo em relação aos de geometria, baseando-se principalmente em cálculos da intensidade dos pixels e a sua correlação com imagens semelhantes, a fim de classificar a identidade da pessoa ou a expressão facial do rosto em questão.

3.2 DATASETS COM IMAGENS DE EXPRESSÕES FACIAIS

Conforme será apresentado no capítulo 5 sobre Trabalhos Correlatos, diferentes mecanismos para detectar expressões faciais já foram propostos por vários pesquisadores. Estes mecanismos requerem pelo menos um *dataset* (base de imagens) de expressões humanas para treinar e validar a eficácia dos modelos propostos. Isto permite que diferentes mecanismos sejam testados com um mesmo *dataset* a fim de comparar, de forma justa, qual mecanismo obtém a melhor acurácia na detecção das expressões. As seções a seguir apresentam alguns *datasets* comumente usados para estudo de expressões faciais.

3.2.1 JAFFE

Figura 9 – Amostra da base de imagens JAFFE



Fonte: Lyons, Kamachi e Gyoba (1998)

The Japanese Female Facial Expression (JAFFE) é um *dataset* proposto por Lyons, Kamachi e Gyoba (1998) que disponibiliza 213 imagens, em tons de cinza, de 10 mulheres japonesas, apresentando 7 expressões faciais distintas: neutra, raiva, nojo, medo,

alegria, tristeza e surpresa. A resolução de cada imagem é de 256 x 256 pixels, no formato *.tiff* sem compressão. Exemplos de seis imagens extraídas deste *dataset* são apresentados na Figura 9.

3.2.2 CK+

A base de imagens CK+ (*Extended Cohn-Kanade*), proposta em 2010, representa uma versão estendida da base de imagens CK (*Cohn-Kanade*) que foi originalmente proposta no ano de 2000. O CK+ corrige algumas imperfeições detectadas no *dataset* CK original, aplicando um processo de revisão, ajustes e validação. Sua versão atual contém imagens de 210 adultos entre 18 e 50 anos, sendo 69% do sexo feminino e 31% do sexo masculino. Além disso, 81% destes adultos são de origem euro-americana, 13% de origem afro-americana e 6% de outras origens. A Figura 10 apresenta amostra de algumas imagens deste *dataset* (LUCEY *et al.*, 2010).

Figura 10 – Amostra da base de imagens CK+



Fonte: Lucey *et al.* (2010)

3.2.3 AR Face Database

O AR *Face Database*, proposto por Martínez e Benavente (1998), é uma base de imagens contendo aproximadamente 3 mil imagens de pessoas em diferentes posições e estilos. Destaca-se por oferecer figuras com diferentes condições de iluminação e características variadas, como pessoas usando óculos de sol, cachecol, diferentes estilos de cabelo, diferentes estilos de roupa, entre outros. Para a elaboração deste banco de imagens foi necessária a contribuição de, ao todo, 116 pessoas, sendo 63 homens e 53 mulheres. Duas sessões de fotos foram feitas para a montagem desta base de imagens de forma que a segunda sessão foi feita 14 dias após a primeira. A Figura 11 apresenta a amostra de algumas imagens deste *dataset*.

Figura 11 – Amostra da base de imagens AR



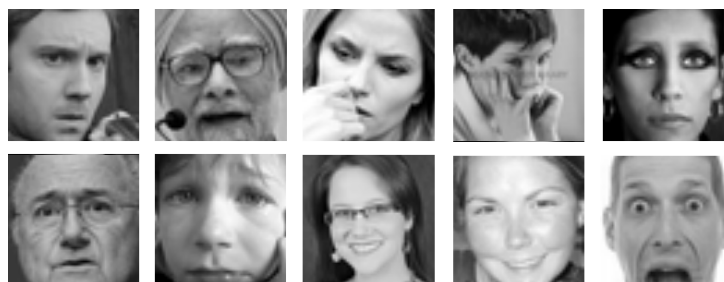
Fonte: Adaptado de Martínez e Benavente (1998)

3.2.4 FER-2013 Database

FER-2013 é um *dataset* proposto por Carrier *et al.* (2013) que contém 35.887 imagens de expressões faciais, todas rotuladas com as expressões universais definidas por Ekman e Oster (1979). Este *dataset* foi utilizado em uma competição do Kaggle, o qual sugere uso de 28.709 destas imagens para o conjunto de treinamento, 3.589 para o conjunto de testes e outras 3.589 para o conjunto de validação.

Todas as imagens são encontradas em escalas de cinza e possuem 48 x 48 pixels de tamanho, conforme pode ser visto na Figura 12. Uma vez que este *dataset* possui uma quantidade muito maior de imagens do que os demais *datasets* apresentados, nos últimos anos o FER-2013 tem sido usado em várias pesquisas para detecção de expressões faciais com Inteligência Artificial.

Figura 12 – Amostra da base de imagens FER-2013



Fonte: Carrier *et al.* (2013)

Este capítulo expôs uma visão geral sobre expressões faciais, destacando as expressões básicas apontadas por Ekman e Friesen e apresentando conceitos importantes do *Facial Action Coding System*. Também foram citados *datasets* com imagens comumente usadas em mecanismos para detectar expressões faciais, assim como detalhes dos pontos FDP/FAP e unidades FAPU. O capítulo seguinte apresentará conceitos importantes

de Inteligência Artificial usados para a construção de mecanismos de computadores que detectam expressões faciais.

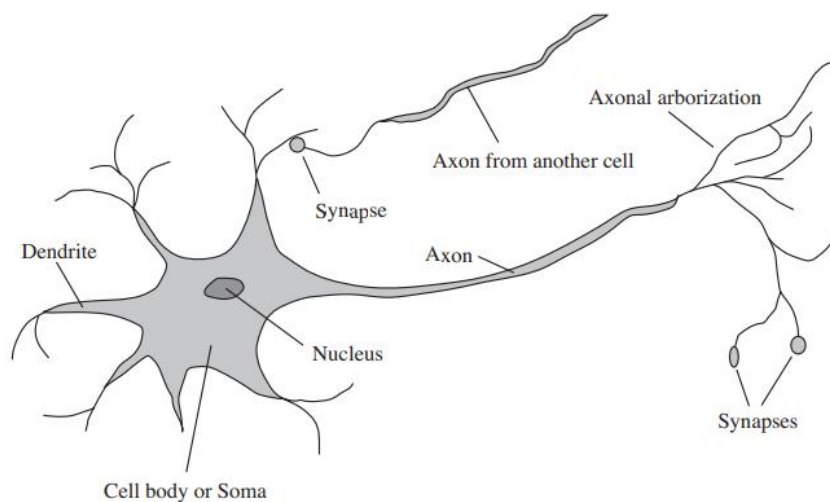
4 INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial (IA) pode ser definida de várias maneiras segundo diferentes autores. Russell e Norvig (2009), por exemplo, indicam que a Inteligência Artificial é o estudo de agentes que percebem características do ambiente externo e executam ações inteligentes a partir delas. Neste ramo de pesquisa, os cientistas tentam não somente entender o funcionamento da inteligência, mas também desenvolver entidades artificiais inteligentes, baseando-se principalmente na estrutura e no funcionamento do cérebro humano.

Diversas técnicas de IA foram desenvolvidas e aprimoradas para as mais diversas finalidades nos últimos anos. Dentre elas, podem ser citados os veículos robotizados que dirigem sozinhos; sistemas de reconhecimento de fala; planejamento autônomo; detecção automática de SPAM; planejamento logístico; robótica; detecção de padrões em imagens; diagnóstico de doenças; entre muitos outros (RUSSELL; NORVIG, 2009).

Russell e Norvig (2009) demonstram que a Inteligência Artificial tem como origem a Neurociência, responsável pelo estudo do sistema nervoso humano e principalmente o estudo do cérebro. Squire *et al.* (2008) apontam também a Neurociência como base biológica que busca explicar o comportamento dos seres vivos, sendo o cérebro composto por redes neurais com milhões de neurônios interconectados. O neurônio representa a unidade mais básica desta rede e na Figura 13 pode ser visualizada a sua estrutura geral, assim como os seus principais componentes.

Figura 13 – Representação de um neurônio biológico



Fonte: Russell e Norvig (2009)

Squire *et al.* (2008) indicam que o neurônio pode ser classificado de diferentes maneiras de acordo com o seu tamanho, formato, função, localização, entre outros. Apesar

de cada classificação ter um objetivo específico no corpo humano, os neurônios possuem vários elementos em comum, os quais são apresentados a seguir de acordo com Squire *et al.* (2008), Haykin (2001), Association *et al.* (2003) e Khan *et al.* (2018):

- **Dendritos:** estendem o corpo nervoso celular e são responsáveis por receber sinais e informações enviadas pelos axônios de neurônios vizinhos. Um dendrito pode receber contato de um, alguns ou muitos outros neurônios.
- **Corpo celular ou Soma:** é o elemento central do neurônio onde seus demais componentes estão ligados e conectados. O corpo celular recebe informações através dos dendritos, processando-as e em seguida delegando a resposta aos axônios.
- **Núcleo:** cercado pelo corpo celular, é o componente responsável por fazer as tomadas de decisão do neurônio.
- **Axônios:** são responsáveis por transmitir a informação do neurônio atual para neurônios vizinhos a ele conectados. O axônio é tipicamente ramificado na sua extremidade, formando o que é conhecido como arborização axônica, que permite a interconexão com um grande número de neurônios.
- **Sinapse:** é conhecido como sinapse ou transmissão sináptica o processo responsável por transmitir entre diferentes neurônios as informações propriamente ditas.

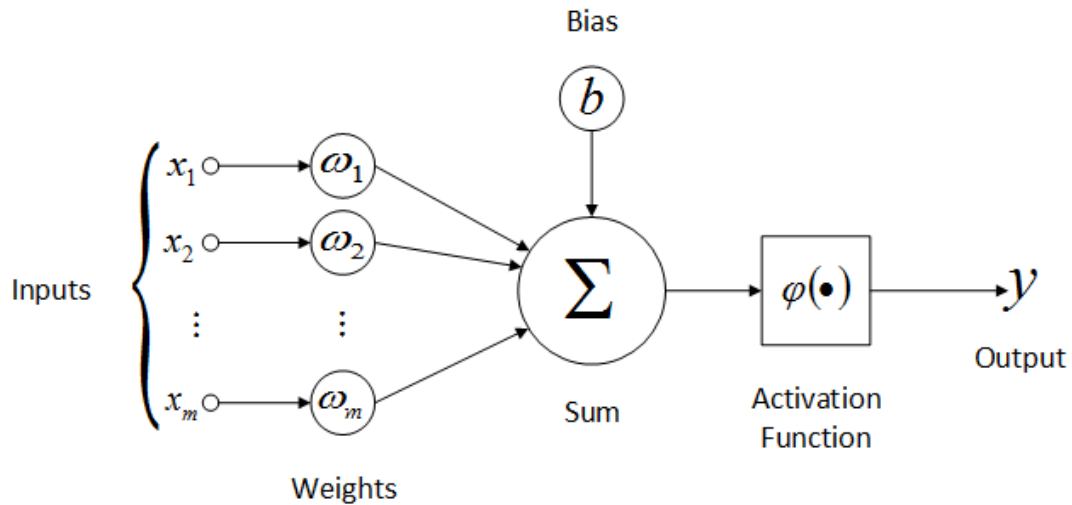
É importante ressaltar que existem diversos mecanismos de IA e cada um possui um propósito específico. O reconhecimento automático de fala, por exemplo, pode ser feito com um mecanismo completamente diferente daqueles utilizados para detecção de SPAM. Um mecanismo pode ter alta eficácia em determinada situação, mas baixa eficácia em outra. É papel do cientista de dados escolher o mecanismo que melhor se adapta ao cenário analisado.

As seções a seguir abordarão neurônios de maneira matemática e artificial, explicando a sua correlação com mecanismos de IA e apresentando algumas das inúmeras técnicas usadas nos dias atuais para o aprendizado de máquina.

4.1 *PERCEPTRON*

Segundo Haykin (2001) e Shalev-Shwartz e Ben-David (2014), as redes neurais artificiais do tipo *Perceptron* foram prototipadas no ano de 1958 por Rosenblatt, sendo a primeira espécie de rede neural apresentada. Uma rede desta natureza é consideravelmente simples de implementar e é usada para classificar dados linearmente entre duas categorias, ou seja, de forma binária. A Figura 14 apresenta um neurônio artificial com seus principais componentes, seguido da respectiva definição de cada um deles.

Figura 14 – Representação de um neurônio artificial



Fonte: Oliveira *et al.* (2017)

- **Inputs:** representam as entradas do neurônio artificial. Quando comparado a neurônios biológicos, as entradas representam os dendritos.
- **Weights:** são os pesos associados a cada entrada, indicando o impacto, força ou importância que cada entrada do neurônio terá na função de ativação.
- **Sum:** somatório das entradas do neurônio multiplicadas pelos respectivos pesos. Pode ser definida, também, como a multiplicação ponderada de cada entrada pelo peso correspondente.
- **Bias:** valor extraordinário adicionado ao somatório das entradas do neurônio multiplicadas pelos pesos, sendo um coeficiente que aumenta ou diminui o valor de entrada na função de ativação. Em outras palavras, se o bias for positivo, aumentará o resultado do somatório; já se for negativo, diminuirá tal resultado.
- **Activation Function:** função de ativação, que recebe como entrada o somatório já acrescido do bias. Seu resultado classifica esta entrada e indica se a mesma é válida ou não, ou se deve ser repassada a neurônios posteriores na rede neural. Esta função também é conhecida como função restritiva visto que restringe o valor em um intervalo pré-definido. Em neurônios do tipo *Perceptron*, o resultado da função de ativação sempre divide os dados de maneira linear em duas classes distintas.
- **Output:** saída do neurônio, que na prática representa o resultado da função de ativação.

Haykin (2001) indica que um neurônio artificial pode ser matematicamente representado pelas equações abaixo. A primeira equação (4.1) indica o somatório de cada

entrada (*input*) multiplicada pelo seu respectivo peso (*weight*). Em outras palavras, representa o somatório da multiplicação ponderada entre entradas e pesos. A segunda equação (4.2) representa de forma genérica uma função de ativação. Pode-se ver que recebe como entrada o somatório da equação recém apresentada, acrescido do bias. Em redes *Perceptron*, a saída da função de ativação é sempre binária e tipicamente $[0, 1]$, ou também representado como $[-1, 1]$.

$$u = \sum_{i=1}^m w_i x_i \quad (4.1)$$

$$y = \varphi(u + b) \quad (4.2)$$

Na seção 4.2 serão apresentados maiores detalhes sobre funções de ativação e algumas das diferentes funções existentes, cada qual com suas principais características e cenários onde são mais propícias para uso.

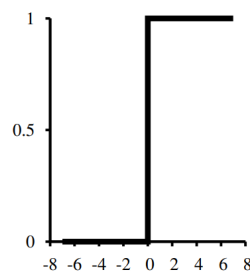
4.2 FUNÇÕES DE ATIVAÇÃO

Segundo Haykin (2001) existem várias funções de ativação que podem ser usadas a fim de gerar diferentes saídas para um neurônio artificial. Serão apresentadas algumas das principais funções seguidas de suas representações gráficas e equações.

4.2.1 Função binária

A função binária, como o próprio nome sugere, classifica a entrada em duas categorias distintas. Pode-se ver na Figura 15 e na equação 4.3 que, sempre que o valor for igual ou maior do que zero, o resultado é 1. Caso contrário, se for menor do que zero, o resultado é sempre zero (RUSSELL; NORVIG, 2009).

Figura 15 – Representação gráfica da função binária



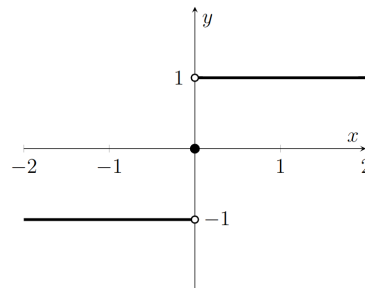
Fonte: Russell e Norvig (2009)

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases} \quad (4.3)$$

4.2.2 Função sinal

Haykin (2001) indica a *função sinal* como variação da função de ativação binária, classificando os dados de entrada em três categorias distintas. Respectivamente, classifica em 1 quando a entrada é positiva; zero, quando a entrada é igual a zero; e, por fim, classifica em -1 quando o valor de entrada é negativo. A Figura 16 representa esta função graficamente seguida de sua equação.

Figura 16 – Representação gráfica da função sinal



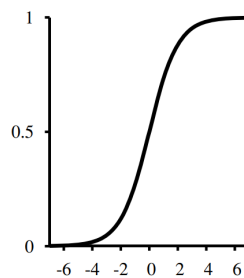
Fonte: do autor

$$\varphi(v) = \begin{cases} 1, & \text{se } v > 0 \\ 0, & \text{se } v = 0 \\ -1, & \text{se } v < 0 \end{cases} \quad (4.4)$$

4.2.3 Função Sigmóide

Segundo Russell e Norvig (2009) e Haykin (2001), a função de ativação *Sigmóide* também é conhecida como função logística e é do tipo não-linear. Esta função torna os resultados mais flexíveis do que as funções binária e sinal pois considera qualquer valor dentro da faixa de zero e 1, não necessariamente dividindo os dados em somente duas ou três categorias. A Figura 17 representa a função *Sigmóide* graficamente.

Figura 17 – Representação gráfica da função Sigmóide



Fonte: Russell e Norvig (2009)

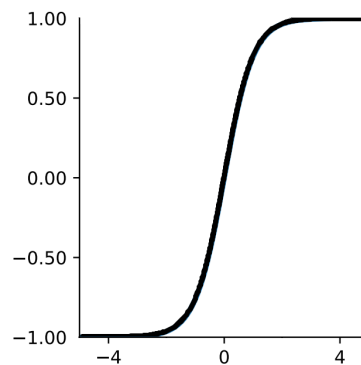
A equação abaixo (4.5) indica a representação matemática da função *Sigmóide*. Nielsen (2015) aponta que, quanto maior a entrada (v) desta função de ativação, mais próximo de 1 o resultado será. Analogamente, quanto menor e mais negativa a entrada na função de ativação, mais próximo de zero o resultado será.

$$\varphi(v) = \frac{1}{1 + e^{-v}} \quad (4.5)$$

4.2.4 Função tangente hiperbólica

De forma similar à função *Sigmóide*, a função tangente hiperbólica também é não-linear e classifica os dados em formato sigmoidal, com a diferença de que os resultados estão dentro da faixa de valores -1 e 1. A Figura 18 apresenta a representação gráfica desta função, seguida de sua respectiva representação matemática.

Figura 18 – Representação gráfica da função tangente hiperbólica



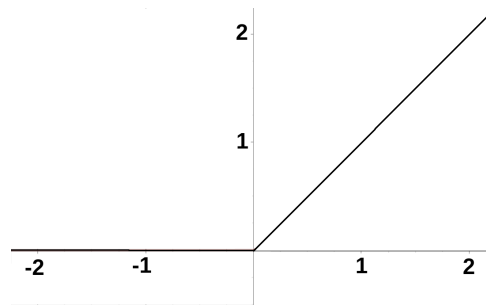
Fonte: do autor

$$\varphi(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}} \quad (4.6)$$

4.2.5 Função ReLU

De acordo com Krizhevsky, Sutskever e Hinton (2012), a função de ativação ReLU (*Rectified Linear Unit*) foi proposta com a intenção de minimizar o impacto dos valores negativos na rede neural. Esta função tende a ser mais rápida do que outras funções de ativação, visto que não apresenta nenhum cálculo complexo e basicamente transforma valores negativos em zero. Desta forma, os valores positivos permanecem inalterados. Seu gráfico é apresentado na Figura 19 enquanto sua equação é indicada na sequência.

Figura 19 – Representação gráfica da função ReLU



Fonte: Adaptado de Agarap (2018)

$$\varphi(v) = \max(0, v) \quad (4.7)$$

4.2.6 Função *Softmax*

A função *Softmax* é utilizada para gerar distribuições probabilísticas em problemas de classificação. Supondo que para um problema existam N classes ou respostas, a função *Softmax* retorna um vetor de N posições, contendo probabilidades entre 0 e 1 para cada uma destas N classes. A posição do vetor que possuir a maior probabilidade representa a classe de saída da rede neural (CHOLLET, 2017).

4.3 MECANISMOS DE APRENDIZAGEM

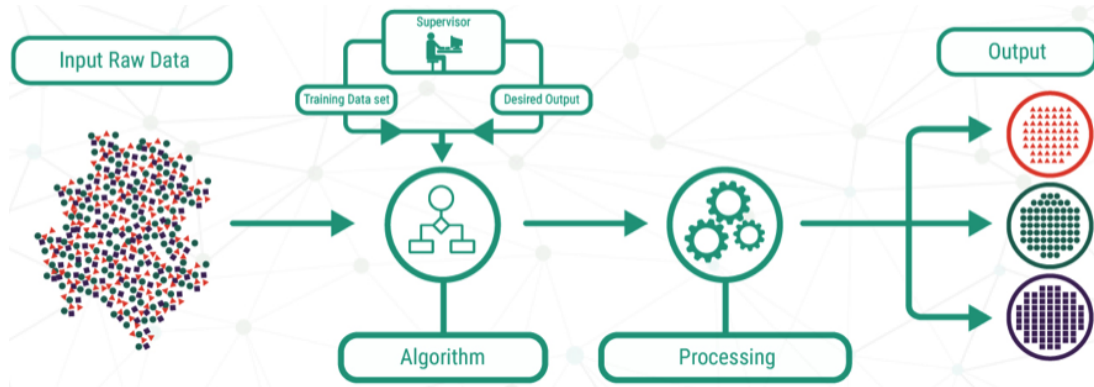
Segundo Haykin (2001), assim como no cérebro humano, a essência da Inteligência Artificial está em aumentar o conhecimento através da experiência. As atividades podem falhar ou ter sucesso e tudo é denominado experiência. Isso torna o sistema, com o passar do tempo, inteligente, permitindo a tomada de sensatas decisões após um período de aprendizagem e treinamento. O processo de aprendizagem em um sistema de IA pode ser, dentre outras classificações, supervisionado ou não-supervisionado.

4.3.1 Supervisionado

De acordo com Haykin (2001), Russell e Norvig (2009) e Mohammed, Khan e Bashier (2016), no processo de aprendizagem supervisionado a rede neural é alimentada com um conjunto de dados pré-rotulados com a resposta correta. Isto permite que a rede detecte padrões e ajuste os pesos sinápticos iterativamente, a fim de gerar saídas correspondentes às respostas rotuladas nas entradas. Ao final do treinamento, dados de teste são apresentados à rede a fim de avaliar seu desempenho e verificar se a acurácia das respostas está em um nível aceitável. O termo *supervisionado* vem justamente do acompanhamento por parte do supervisor (tipicamente o cientista de dados) nas etapas

de treinamento e teste. A Figura 20 apresenta o fluxo de um processo de aprendizagem supervisionado.

Figura 20 – Processo de aprendizagem supervisionado

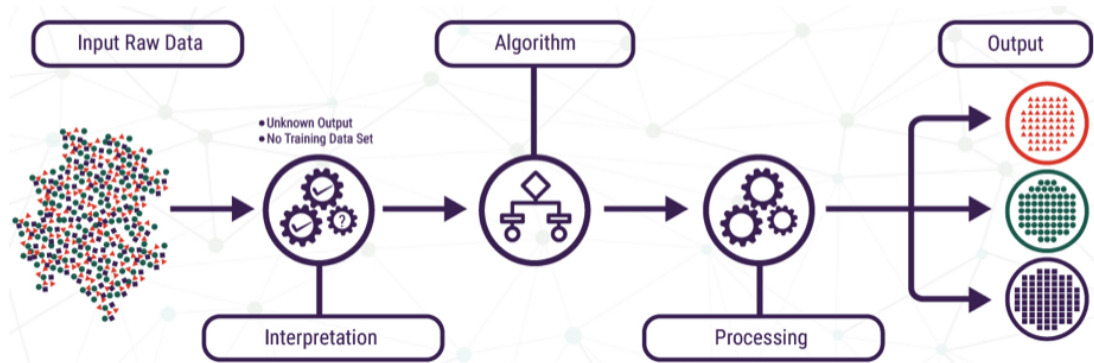


Fonte: Adaptado de Loon (2018)

4.3.2 Não-supervisionado

Segundo Russell e Norvig (2009), Loon (2018) e Mohammed, Khan e Bashier (2016), o processo não-supervisionado não faz uso de dados de treinamento e nem possui supervisores, como o próprio nome já indica. A rede neural é alimentada com dados que não estão pré-rotulados com a resposta correta. Seu propósito é encontrar padrões e correlações entre os dados sem nenhum treinamento prévio, em busca de classificações. A Figura 21 apresenta o fluxo de um processo de aprendizagem não-supervisionado.

Figura 21 – Processo de aprendizagem não-supervisionado



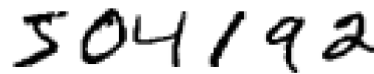
Fonte: Adaptado de Loon (2018)

4.4 REDES NEURAIAS PROFUNDAS

Nielsen (2015) indica que o cérebro humano é capaz de facilmente reconhecer e classificar objetos por meio da visão, tarefa esta que possui um altíssimo nível de complexidade para resolver programaticamente com o uso de computadores. Descrever um algoritmo que indique como reconhecer o dígito nove, por exemplo, pode parecer simples.

Para isso seria necessário detectar um círculo na parte superior, ligado a uma linha vertical (ou possivelmente curvada) na parte inferior, conforme pode ser visto na Figura 22, que apresenta alguns dígitos escritos à mão. O fato é que este círculo pode não ser completo e a linha pode ter traços e características que variam de pessoa para pessoa. Além disso, o dígito também pode estar parcialmente oculto por outra figura e mesmo assim continuar representando um nove. A quantidade de situações, regras e exceções que o algoritmo precisaria prever é altíssima, aumentando muito a complexidade do *software*. Para resolver esta tarefa podem ser usadas redes neurais profundas.

Figura 22 – Dígitos escritos à mão


 A photograph of the handwritten digits '504192' in black ink on a white background. The digits are written in a casual, slightly slanted style.

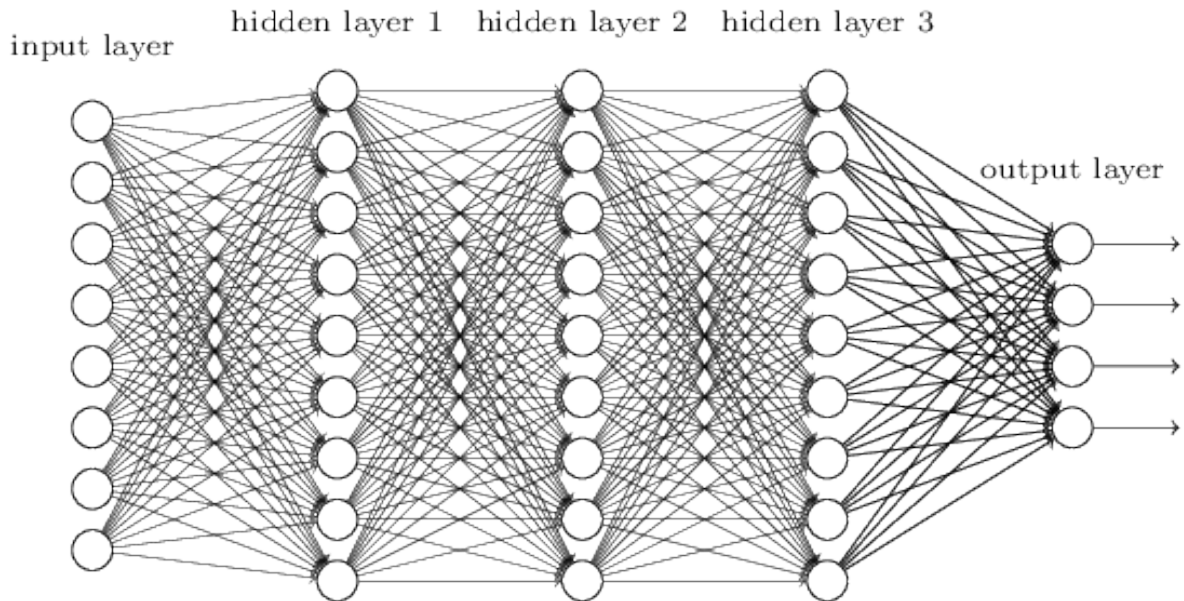
Fonte: Nielsen (2015)

De acordo com Haykin (2001) e Nielsen (2015), a principal característica de uma rede neural profunda é a existência de múltiplas camadas de neurônios para encontrar a resposta final. Tipicamente, cada camada interage apenas com as camadas predecessora e sucessora: sua entrada é alimentada pela saída da camada anterior, assim como sua saída serve de entrada para a camada seguinte. Pode-se dizer que cada camada na rede neural é responsável por decompor o problema em frações menores. O comportamento de redes neurais profundas é análogo ao funcionamento do cérebro humano, onde inúmeros neurônios são ativados e trabalham em conjunto para chegar ao resultado final.

Haykin (2001) e Nielsen (2015) também indicam que as camadas intermediárias da rede (aquelas que não são camadas de entrada e nem de saída) são denominadas ocultas. Esta denominação é feita pois as camadas ocultas são usadas para processamento interno na rede, não recebendo diretamente informações externas e nem gerando resultados finais que resolvem o problema proposto. É possível ver na Figura 23 uma rede neural com 5 camadas, onde a primeira representa a entrada (camada onde a rede é alimentada), seguida de três camadas ocultas para processamento dos dados e uma quinta e última camada de saída, responsável pela classificação final da imagem.

Uma das abordagens para resolver o problema de reconhecimento de dígitos manuscritos, com o uso de redes neurais profundas, seria alimentar a rede neural com um número consideravelmente grande (milhares, milhões ou até mesmo bilhões) de exemplos com imagens de dígitos manuscritos, juntamente com o respectivo rótulo indicando qual dígito a imagem representa. Esta modalidade é considerada como aprendizagem supervisionada (NIELSEN, 2015), conforme citado na seção 4.3.1.

Figura 23 – Representação de uma rede neural profunda

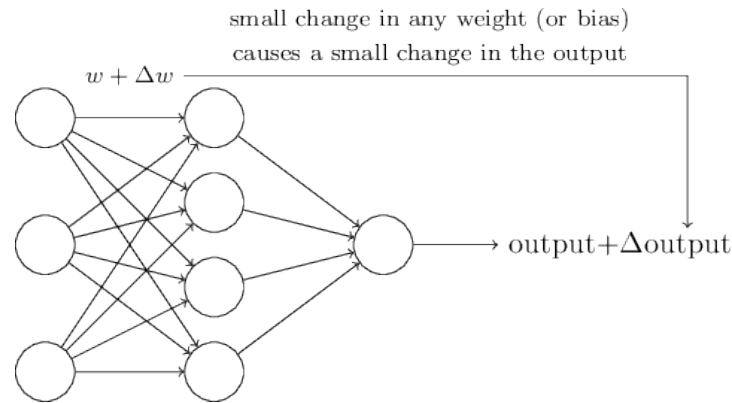


Fonte: Nielsen (2015)

Segundo Nielsen (2015), as redes neurais profundas normalmente são construídas com neurônios *Sigmóide* (ou suas variações) e não com neurônios *Perceptron*. Os neurônios *Perceptron* simplesmente classificam a resposta de forma binária, conforme visto na seção 4.1, o que não é suficiente em problemas e situações complexas operadas por redes neurais profundas. Uma pequena alteração nos pesos e bias de *Perceptrons* poderia modificar completamente o resultado da rede, enquanto a ideia é fazer sutis mudanças em tais valores para suavemente melhorar a convergência ao resultado e aumentar a acurácia da rede, calibrando a resposta.

A Figura 24, também apresentada por Nielsen (2015), representa a abstração de uma rede neural profunda composta por neurônios do tipo *Sigmóide*. Pode-se ver que a saída é suavemente modificada pelas sutis alterações Δw aplicadas nos pesos sinápticos da rede. Uma vez que neurônios *Sigmóide* geram quaisquer valores dentro da faixa de zero e 1, o processo de aprendizagem se torna muito mais flexível do que a aprendizagem com *Perceptrons*. Pode ser feita analogia ao reconhecimento de dígitos manuscritos: supondo que o dígito nove não esteja sendo detectado corretamente pela rede com o uso de *Perceptrons*, uma pequena mudança nos pesos e bias poderia ajustar a detecção de tal dígito, mas possivelmente estragar a detecção dos demais. Já com neurônios *Sigmóide*, pequenos ajustes são capazes de aprimorar a detecção do dígito nove sem interferir na detecção dos demais.

Figura 24 – Abstração de uma rede neural com neurônios *Sigmóide*



Fonte: Nielsen (2015)

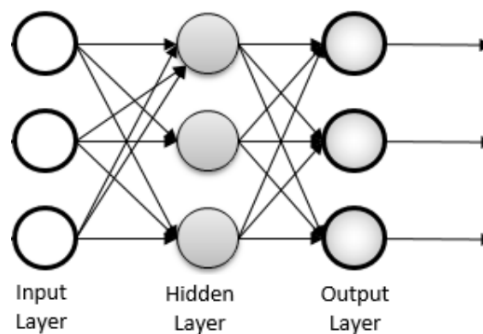
Na seção atual foi introduzido o funcionamento de redes neurais profundas e apresentadas suas principais características. Nas seções a seguir, algumas categorias de redes neurais serão detalhadas, abordando seus principais objetivos e benefícios.

4.4.1 Redes *feed-forward*

Segundo Russell e Norvig (2009), Nielsen (2015) e Du e Swamy (2014), as redes neurais profundas do tipo *feed-forward* são aquelas onde a saída dos neurônios de uma camada é a entrada para neurônios da camada seguinte, formando um grafo acíclico. Nela, as informações sempre trafegam em uma única direção, sendo uma categoria de rede consideravelmente simples pois não há *loops* onde a informação volta para camadas anteriores da rede neural.

Du e Swamy (2014) também apontam que nesta categoria de rede neural não há comunicação entre neurônios de uma mesma camada, reforçando o aspecto de comunicação unidirecional entre diferentes camadas. A Figura 25 apresenta uma rede neural *feed-forward* com uma camada de entrada, uma camada oculta e uma camada de saída.

Figura 25 – Representação de uma rede neural *feed-forward*

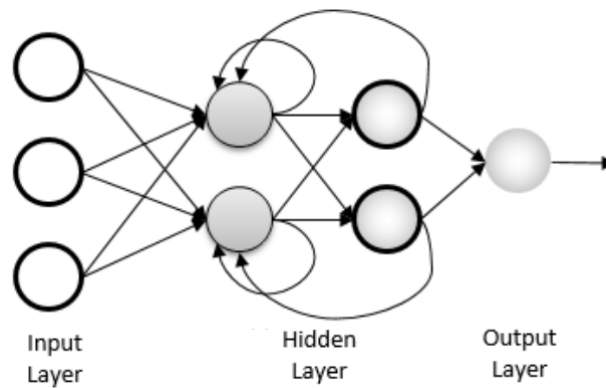


Fonte: Adaptado de Pekel e Kara (2017)

4.4.2 Redes recorrentes

Segundo Haykin (2001), Nielsen (2015), Russell e Norvig (2009) e Du e Swamy (2014), as redes neurais recorrentes são mais complexas do que redes *feed-forward* pois permitem que a saída do neurônio em uma determinada camada volte para camadas anteriores, permitindo *loops* de *feedback*. Ao mesmo tempo em que essa característica torna a rede mais complexa, também a torna mais robusta na detecção de informações e no processo de aprendizagem. As redes recorrentes também possibilitam o uso de *short-term memory* (memória de curto prazo) que armazenam informações temporárias durante o processo de aprendizagem. A Figura 26 apresenta os componentes principais de uma rede neural recorrente, reforçando o fato de que resultados podem ser enviados a neurônios de camadas anteriores dentro da rede neural.

Figura 26 – Representação de uma rede neural recorrente



Fonte: Adaptado de Pekel e Kara (2017)

4.5 REDES NEURAIAS CONVOLUCIONAIS

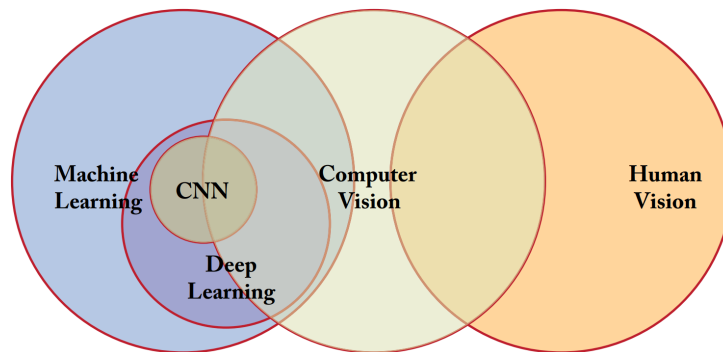
Goodfellow, Bengio e Courville (2016) e Khan *et al.* (2018) apontam que redes neurais convolucionais são focadas na detecção de padrões em estruturas de dados que possuem duas ou mais dimensões. Também conhecida como *Convolutional Neural Network* (CNN), tal categoria de rede é propícia para a manipulação de matrizes e análise de imagens, visto que uma determinada imagem i é a representação visual da respectiva matriz m . Nela, cada posição representa a intensidade RGB de um de seus *pixels*. Pode-se citar, também, o uso de redes desta categoria para a visão computacional: robótica, *drones*, medicina e carros autônomos (capazes de dirigir por conta própria) seriam apenas algumas das inúmeras aplicações.

Khan *et al.* (2018) reforçam a robótica como um dos eminentes nichos para uso de visão computacional e CNNs. Tal tecnologia permite que, em sistemas avançados, robôs detectem o ambiente onde estão inseridos e tomem decisões similares às tomadas por humanos: pular, correr, desviar, caminhar, entre outros. Além disso, também poderiam

reconhecer seres e objetos ao seu redor para interagir da forma que for mais conveniente para a situação.

A Figura 27 indica onde as CNNs se encontram dentro do nicho de pesquisa de Inteligência Artificial. Esta categoria de rede é uma subárea do aprendizado profundo (*Deep Learning*), que por sua vez é uma subárea do aprendizado de máquina (*Machine Learning*). A visão computacional faz uma intersecção entre o mundo artificial das CNNs e a visão humana biológica (KHAN *et al.*, 2018).

Figura 27 – Relação entre redes convolucionais e demais áreas de estudo

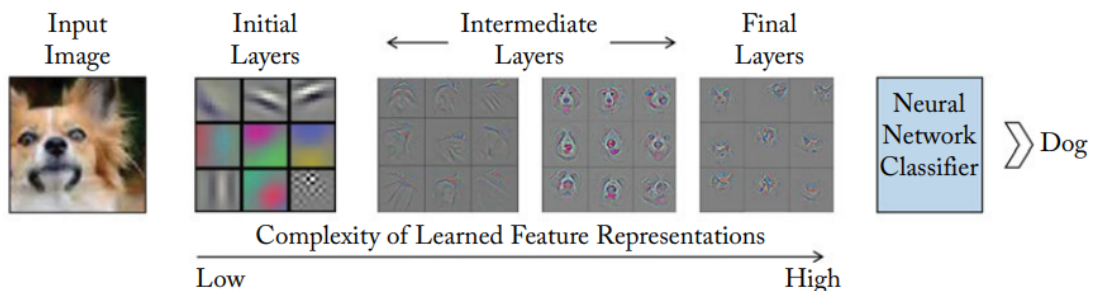


Fonte: Khan *et al.* (2018)

A convolução e o *pooling* representam dois mecanismos muito importantes em uma rede convolucional, os quais serão detalhados nas seções seguintes do presente capítulo. Existem diferentes arquiteturas de redes convolucionais propostas por diferentes autores, sendo possível que a convolução e o *pooling* se repitam inúmeras vezes dentro da rede, dependendo de como a mesma foi projetada. Algumas arquiteturas conhecidas são *LeNet*, *AlexNet*, *GoogLeNet*, *ResNet* e *FractalNet*.

A Figura 28 apresenta, de forma simplificada, o funcionamento de uma rede do tipo convolucional. Pode-se ver que em cada camada a Inteligência Artificial detecta características abstratas da imagem, iniciando pelas mais simples até as mais complexas (KHAN *et al.*, 2018).

Figura 28 – Funcionamento de uma rede convolucional



Fonte: Khan *et al.* (2018)

4.5.1 Convolução

Khan *et al.* (2018) apontam a convolução como o processo mais importante das redes neurais convolucionais, como o próprio nome da rede já indica. Neste processo, um filtro, também conhecido por *kernel*, é aplicado na matriz original para modificar seus valores a fim de destacar as características mais importantes da imagem. Por sua vez, o *kernel* também é uma matriz, que pode ter os valores inicializados na primeira iteração da rede neural e aprimorados conforme a rede é treinada. Quanto melhores os valores do *kernel*, melhor tende a ser a extração de características da imagem e sua posterior classificação na última camada da rede neural.

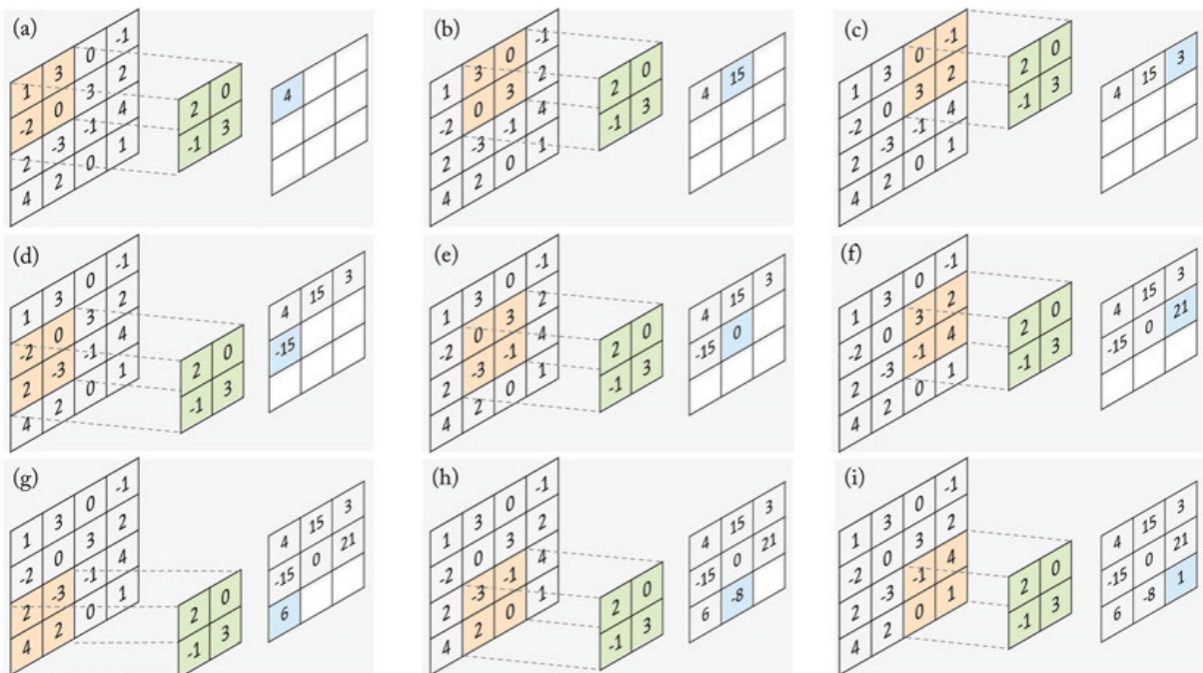
A Figura 29 apresenta um *kernel* de tamanho 2 x 2 usado para extrair características da imagem, enquanto a Figura 30 apresenta o processo completo de convolução com o *kernel* recém citado. A cada iteração os valores do *kernel* são multiplicados pelas respectivas posições em uma subárea da matriz original, sendo o somatório de tais multiplicações o resultado utilizado para construir a matriz de saída (KHAN *et al.*, 2018).

Figura 29 – *Kernel* de tamanho 2 x 2 para convolução

2	0
-1	3

Fonte: Khan *et al.* (2018)

Figura 30 – Processo de convolução em uma CNN



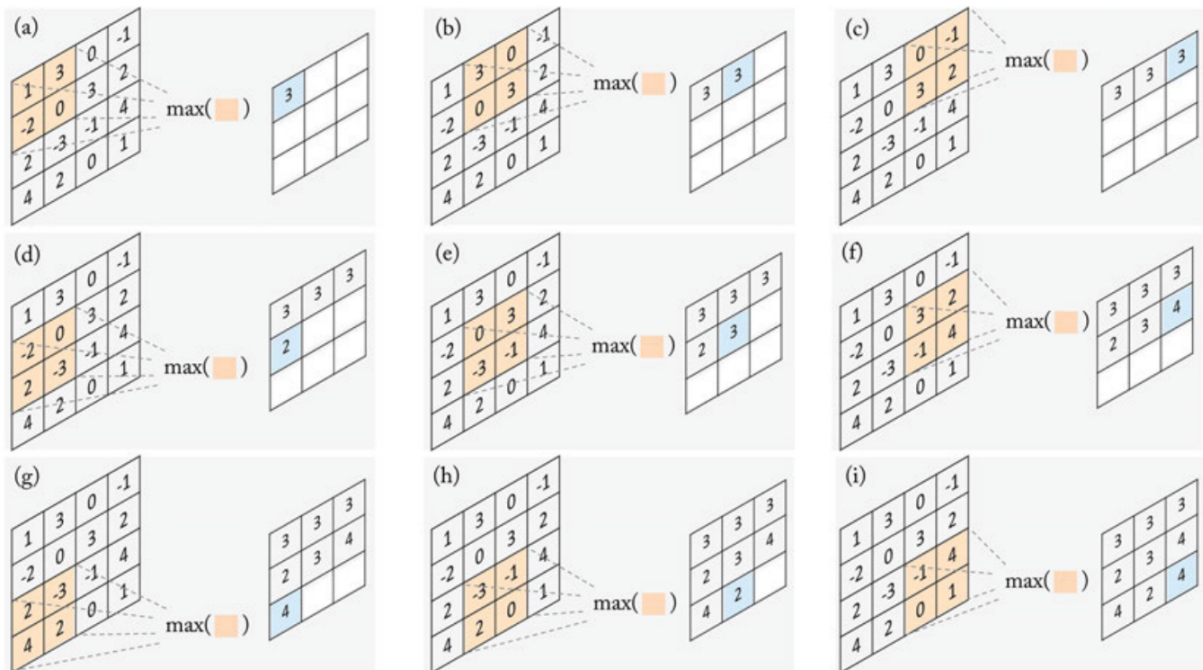
Fonte: Khan *et al.* (2018)

Pode-se ver também na Figura 30 que na primeira iteração, denominada por (a), os valores $[2, 0, -1, 3]$ do *kernel* são multiplicados pelos valores $[1, 3, -2, 0]$ da matriz convolucionada. Tais multiplicações geram os valores $[2, 0, 2, 0]$ que, somados, totalizam 4, sendo este o resultado a ser inserido na primeira posição da matriz de saída. O processo se repete iterativamente deslocando e aplicando o *kernel* nas demais posições da matriz convolucionada até que todas as posições tenham os valores recalculados, como é possível visualizar na última iteração, denominada por (i).

4.5.2 Pooling

Segundo Nielsen (2015), Khan *et al.* (2018) e Dominguez-Morales (2018), o *pooling* é um processo iterativo responsável por reduzir o tamanho da matriz convolucional aplicando *kernel* que detecta e extrai suas características mais relevantes. Por consequência, as características menos relevantes são descartadas, dado que o objetivo é diminuir o tamanho da matriz. Saha (2018) aponta que este procedimento também é importante para aprimorar o desempenho nas camadas posteriores da rede, pois uma matriz de menores dimensões representa menos dados a serem processados. As camadas de *pooling* muitas vezes sucedem camadas de convolução dentro de uma CNN.

Figura 31 – *Max pooling* em uma CNN



Fonte: Khan *et al.* (2018)

O *max pooling* ganha destaque dentre as operações de *pooling* existentes, sendo amplamente utilizado em arquiteturas modernas de redes convolucionais. O seu objetivo é encontrar o maior valor dentro da área abrangida pelo *kernel* de *pooling*. A Figura 31,

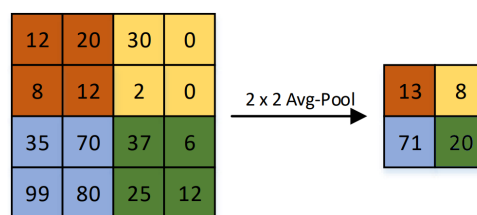
proposta por Khan *et al.* (2018), apresenta uma operação *max pooling* em funcionamento. Pode-se ver que a cada iteração o maior valor encontrado nas quatro posições do *kernel* é escolhido para alimentar a matriz de saída. O *kernel* é deslocado em uma posição e o processo é repetido até que toda a matriz convolucional seja analisada, construindo ao final a matriz de tamanho 3 x 3 apresentada (MICHELUCCI, 2018) (NIELSEN, 2015) (KHAN *et al.*, 2018).

Ainda na Figura 31 apresentada por Khan *et al.* (2018), pode-se ver que na primeira iteração (a) o maior dos valores analisados na região [1, 3, -2, 0] é o número 3, sendo este o resultado da primeira posição da matriz de saída. O processo é repetido iterativamente deslocando o *kernel* para as posições seguintes até que todas as posições da matriz passem pelo processo de *pooling*.

Segundo Dominguez-Morales (2018), o *average pooling*, por sua vez, aplica um procedimento similar ao previamente citado no *max pooling*, com a diferença de que calcula o valor médio das posições abrangidas pelo *kernel* ao invés de assumir o maior valor encontrado. Ambas as operações *max* e *average pooling* podem ser usadas em conjunto ou separadamente dentro de uma rede convolucional de acordo com a arquitetura projetada. A Figura 32 apresenta o funcionamento de uma operação *average pooling*, onde os valores de cada região 2 x 2 da matriz convolucional foram somados e divididos por 4, que é a quantidade de posições abrangidas pelo *kernel*. O resultado é a média aritmética de tais valores, exibidos na matriz apresentada à direita da imagem.

Khan *et al.* (2018) e Saha (2018) também apontam o *stride* como um parâmetro muito importante nas operações de convolução e *pooling*. Este parâmetro indica o deslocamento do *kernel* na matriz convolucional a cada iteração, que conseqüentemente interfere no tamanho da matriz construída ao final do processo. Por exemplo, um *stride* de tamanho 2 significa que o *kernel* será deslocado duas posições para a direita a cada iteração, assim como um *stride* de tamanho 1 indica deslocamento do *kernel* em apenas uma posição. Na Figura 31, previamente apresentada sobre o *max pooling*, pode-se ver que o *stride* é de tamanho 1 pois o *kernel* foi deslocado em uma única posição, assim como na Figura 32 o *stride* é de tamanho 2.

Figura 32 – Average pooling em uma CNN

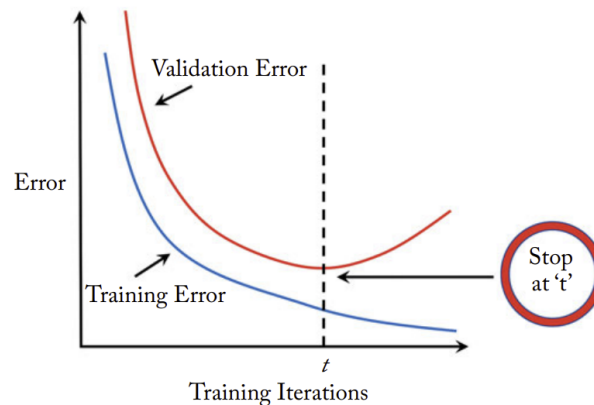


Fonte: Adaptado de Dominguez-Morales (2018)

4.5.3 *Overfitting*

O *overfitting* é um problema muito comum que ocorre durante o treinamento de redes neurais profundas. De acordo com Khan *et al.* (2018) e Chollet (2017), ao invés da rede neural de fato extrair as características das imagens a fim de obter um excelente nível de generalização, a rede acaba por memorizar as imagens usadas no treinamento e desta forma não classifica adequadamente imagens que não estiveram presentes no treinamento.

Figura 33 – *Overfitting* durante o processo de treino



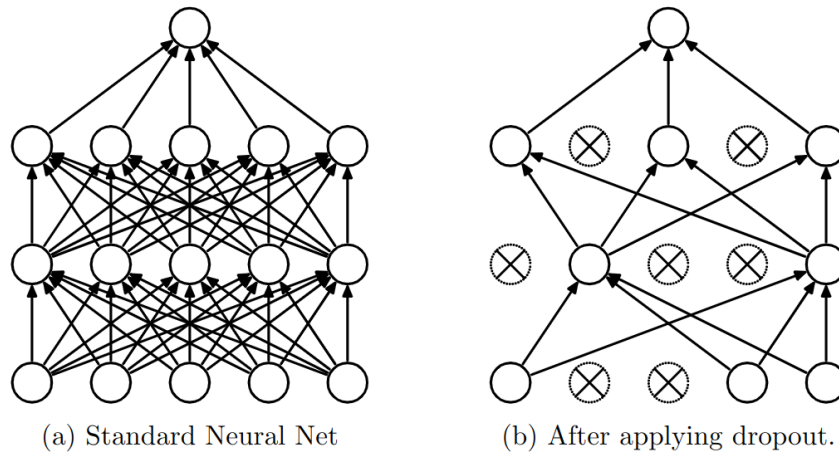
Fonte: Khan *et al.* (2018)

A Figura 33 exibe um gráfico evidenciando que, a partir de uma determinada época ou iteração t , a taxa de erro começa a aumentar consideravelmente para a base de dados de validação enquanto diminui para a base de dados de treino. Isto significa que a rede está perdendo sua capacidade de generalização e começando a memorizar a base de dados de treino, errando respostas para imagens até o momento desconhecidas.

4.5.4 *Dropout*

As camadas *Dropout* são responsáveis por desativar aleatoriamente um determinado percentual de neurônios da rede neural, com o objetivo de evitar ou minimizar o *overfitting* (SRIVASTAVA *et al.*, 2014). Desta forma, os neurônios artificiais que permanecerem ativos precisam trabalhar seus pesos em busca da resposta correta mesmo com um total diferente de neurônios, evitando que os pesos permaneçam iguais durante épocas do treinamento e consequentemente evitando a memorização de imagens. A Figura 34 apresenta uma rede neural antes e depois de aplicar o mecanismo de *dropout*.

Figura 34 – Representação de *dropout* em uma rede neural

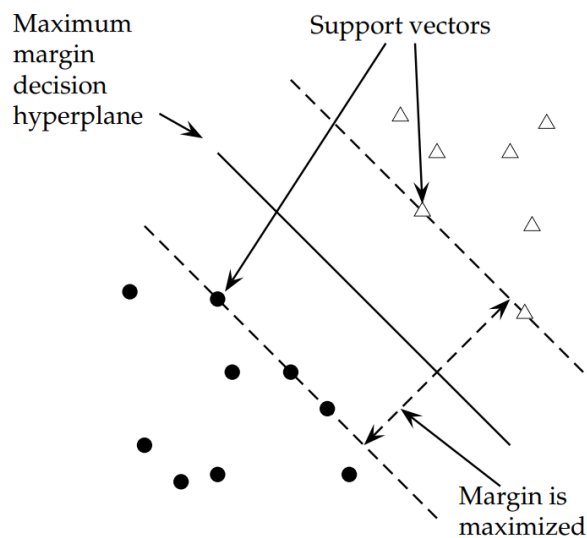


Fonte: Srivastava *et al.* (2014)

4.6 SUPPORT VECTOR MACHINES

De acordo com Kowalczyk (2017) e Deng, Tian e Zhang (2012), *Support Vector Machines* (SVM) foram inicialmente introduzidas por Vapnik na década de 90 e buscam a melhor predição possível para dados futuros, analisando de forma inteligente os dados disponíveis no momento. Em outras palavras, não buscam simplesmente um hiperplano que seja adequado aos dados atuais, mas sim um hiperplano que também consiga enquadrar adequadamente os dados futuros (que no momento ainda são desconhecidos). As SVMs são usadas para predições em diversas áreas, como previsão do tempo, categorização de texto e reconhecimento de fala.

Figura 35 – Características principais de uma *Support Vector Machine*



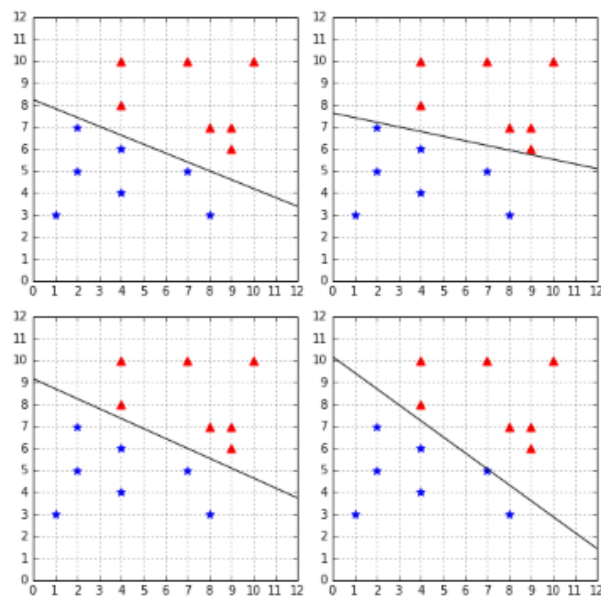
Fonte: Manning, Raghavan e Schütze (2009)

Como pode ser visto na Figura 35, uma *Support Vector Machine* tenta maximizar

a margem entre os diferentes conjuntos de dados analisados. O hiperplano mais adequado é aquele que se diferencia igualmente entre os dois conjuntos de dados, ou seja, aquele em que a margem fica igual para ambos os lados. Os pontos mais próximos da linha separadora (*hyperplane*) são conhecidos como *support vectors* e são os mais importantes para o cálculo das margens e localização da linha separadora (MANNING; RAGHAVAN; SCHÜTZE, 2009).

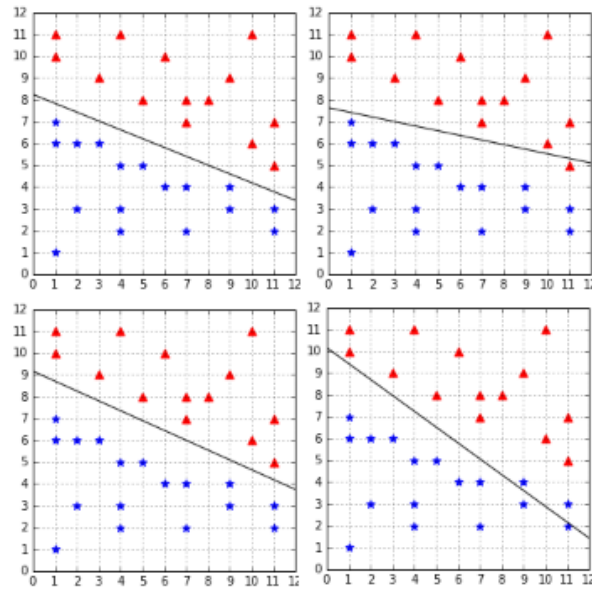
Uma de suas diferenças em relação à rede neural do tipo *Perceptron* é que esta última busca meramente um hiperplano válido capaz de enquadrar os dados atualmente apresentados à rede, sem aplicar técnicas de maximização em busca do melhor hiperplano possível para potencializar a predição e melhor enquadrar dados futuros. A Figura 36 apresenta um exemplo de rede *Perceptron* que gerou quatro hiperplanos válidos para os dados apresentados. Estes hiperplanos não foram maximizados, porém, classificam adequadamente os dados atuais. Já a Figura 37 mostra um exemplo onde, ao apresentar novos dados à esta mesma rede neural, apenas dois destes quatro hiperplanos continuam enquadrando todos os dados adequadamente pelo fato dos hiperplanos não terem sido maximizados (KOWALCZYK, 2017) (MANNING; RAGHAVAN; SCHÜTZE, 2009).

Figura 36 – Rede *Perceptron* encontrando diferentes hiperplanos válidos



Fonte: Kowalczyk (2017)

Figura 37 – Hiperplanos da rede *Perceptron* enquadrando incorretamente os dados



Fonte: Kowalczyk (2017)

Neste capítulo foram apresentados diversos conceitos de Inteligência Artificial, desde sua inspiração na Neurociência até mecanismos de aprendizagem com *Deep Learning*. Outros assuntos importantes dentro deste nicho de estudo também foram abordados: processos de aprendizagem supervisionado e não-supervisionado, redes *Perceptron*, funções de ativação, redes *feed-forward*, redes recorrentes, redes neurais convolucionais e, por fim, *Support Vector Machines*. O capítulo seguinte dará continuidade em alguns destes conceitos, apresentando trabalhos propostos por outros pesquisadores que buscam o estado da arte na detecção de expressões faciais.

5 TRABALHOS CORRELATOS

Várias pesquisas já foram efetuadas na área que estuda e trabalha a detecção automática de expressões faciais. Os mecanismos propostos pelos pesquisadores são variados, contemplando desde a análise de pixels com Processamento Digital de Imagens (PDI) até reconhecimento de padrões mesclando Inteligência Artificial com *Support Vector Machines*. O presente capítulo detalha alguns trabalhos realizados por outros pesquisadores neste mesmo nicho de estudo, expondo o funcionamento geral de cada mecanismo proposto e relatando os resultados alcançados por cada um deles.

5.1 AUTOMATIC FACIAL EXPRESSION RECOGNITION USING DCNN

O rosto humano indica e sugere diversas características sobre a pessoa, como idade, sexo e até mesmo possíveis emoções e estado mental. O artigo *Automatic Facial Expression Recognition Using DCNN*, proposto por Mayya, Pai e M. (2016) apresenta um modelo de rede neural convolucional profunda para análise de expressões faciais. O estudo, baseado em aparência e não em geometria, é feito com o *framework* Caffe com dois *datasets* de imagem: *Japanese Female Facial Expression* (JAFFE) e CK+.

Segundo Jia *et al.* (2014), *Convolutional Architecture for Fast Feature Embedding* (Caffe) é um *framework* para criação de algoritmos para Inteligência Artificial com aprendizado de máquina. Este *framework* permite construir a rede neural e formar cada uma de suas camadas a partir de arquivos de configuração, não sendo obrigatório o desenvolvimento de código-fonte para elaborar a rede neural em questão.

Existem diversas técnicas que podem ser aplicadas e que vem evoluindo com o passar dos anos no campo de pesquisa sobre soluções baseadas em aparência. Dentre as técnicas existentes, algumas podem ser citadas, como *Support Vector Machines* (SVM), *Deep Convolutional Neural Network* (DCNN), *Dynamic Bayesian Network* (DBN) e *Local Binary Pattern* (LBP). Tais técnicas podem ser usadas em conjunto para usufruir do melhor que cada uma oferece. O mecanismo proposto por Mayya, Pai e M. (2016) é composto por três etapas principais:

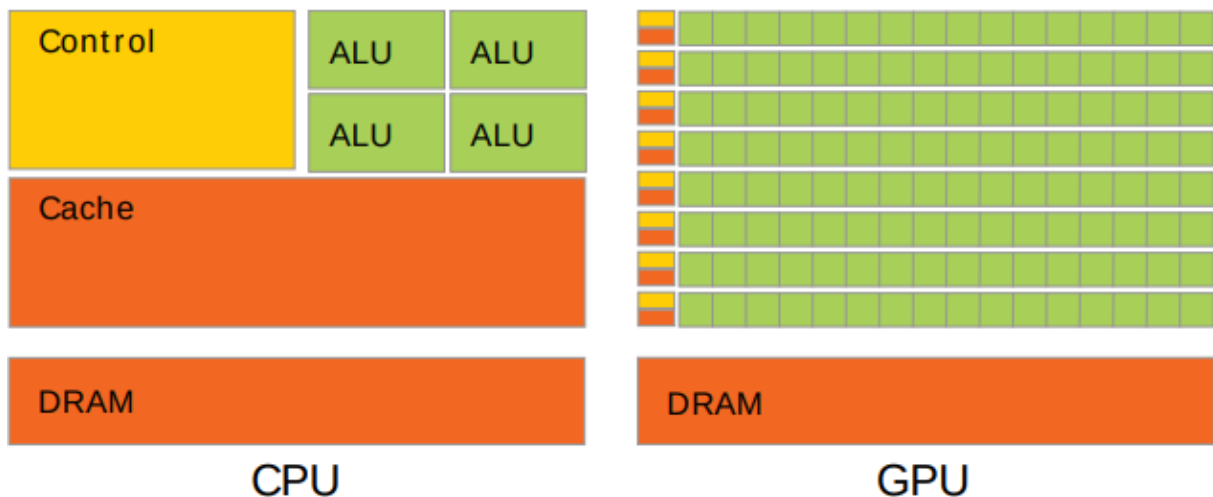
- **Pré-processamento:** as imagens são cortadas a fim de remover elementos desnecessários nas etapas seguintes, evidenciando e registrando somente o rosto a ser analisado. Isto é feito através da biblioteca OpenCV, uma biblioteca de código aberto com inúmeras funções para visão computacional.
- **Treinamento:** através de uma *Deep Convolutional Neural Network* composta por 5 camadas é efetuado o treinamento da rede, gerando, ao final, uma imagem de

pequenas dimensões que será usada na etapa de classificação.

- **Classificação:** através de uma *Support Vector Machine* a imagem é finalmente classificada em uma das expressões universais básicas definidas por Ekman e Friesen (2003).

De acordo com Kirk e Hwu (2016) e NVIDIA (2019), o processamento de dados na etapa de treinamento é realizado através da GPU (*Graphics Processing Unit*) buscando alta performance com o paralelismo oferecido por este *hardware*. Tal característica torna a GPU muito eficaz para cálculos matriciais e, por consequência, adequada para a análise de imagens com Inteligência Artificial. As GPUs disponibilizam inúmeros microprocessadores que executam tarefas e *threads* simultaneamente, como pode ser visto na Figura 38, que compara a arquitetura geral de uma CPU (*Central Processing Unit*) com uma GPU.

Figura 38 – Arquitetura geral de uma CPU e de uma GPU



Fonte: NVIDIA (2019)

Ambas as arquiteturas disponibilizam memória DRAM (*Dynamic Random Access Memory*) como armazenamento principal de dados temporários. O foco de cada arquitetura, no entanto, é diferente: a CPU é focada em execução de código sequencial que faz uso de intenso controle lógico, dispondo de um *cache* de alta performance para minimizar acessos à DRAM e consequentemente diminuir a latência; já a GPU tem como foco principal o cálculo paralelo de pontos flutuantes e não um intenso controle lógico. O número de microprocessadores é consideravelmente maior na GPU (COELHO *et al.*, 2016) (AMOR, 2016) (COELHO *et al.*, 2017).

A etapa de treinamento é feita por uma rede neural composta por 5 camadas que envolvem: convolução, aplicando 96 filtros de tamanho 11 x 11 x 3; função de ativação ReLU a fim de evidenciar as propriedades não-lineares da rede; *max-pooling* de tamanho 3

x 3 para diminuir e suavizar a imagem aplicando cálculos de acordo com a tonalidade e intensidade dos pixels vizinhos; normalização de dados através do algoritmo *Local Response Normalization* (LRN); e, por fim, a classificação propriamente dita.

A normalização com LRN é consideravelmente usada em Inteligência Artificial antes de acionar a função de ativação, a fim de aprimorar os dados analisados e aumentar a taxa de classificação. Com ela também é possível eliminar possíveis ruídos e/ou ajustar pixels fora da curva que poderiam distorcer os resultados. Ao final da rede neural, na etapa de classificação, o tamanho de cada área é de 6 x 6 x 256 pixels, sendo a nível de performance um tamanho aceitável para a classificação. Estes pixels são delegados a uma *Support Vector Machine* (SVM) responsável por fazer a classificação final das expressões faciais.

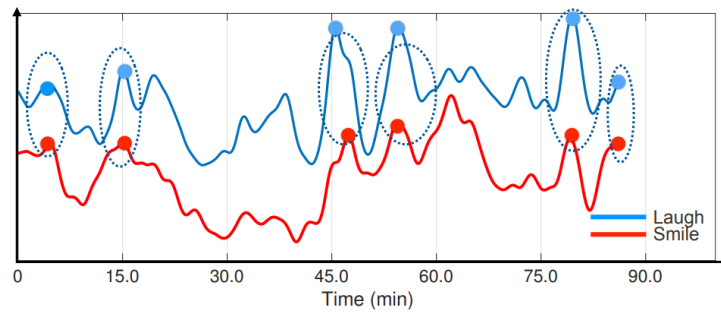
Os resultados alcançados por este trabalho correlato são promissores para as expressões neutra, raiva, nojo, medo, alegria, tristeza e surpresa, chegando a 98,12% de acurácia com o *dataset* JAFFE. Com CK+ os resultados também foram muito bons e chegaram a 96,02% contemplando expressões de raiva, desprezo, nojo, medo, alegria, tristeza e surpresa.

5.2 FACTORIZED VARIATIONAL AUTOENCODERS FOR MODELING AUDIENCE REACTIONS TO MOVIES

O estudo proposto por Deng *et al.* (2017) apresenta o mecanismo *Factorized Variational Autoencoders* (FVAE) a fim de analisar reações da audiência durante a exibição de filmes. Utilizado e validado pela *Disney Research*, tal método permite reconhecer diferentes expressões faciais e identificar se as pessoas estão rindo, sorrindo ou com expressão neutra nos mais variados momentos do filme. Este estudo também permite que outras possíveis expressões sejam detectadas no decorrer do filme se assim desejado, possibilitando mensurar o quão próximas estão as reações das pessoas em relação às reações esperadas no momento de produzir o filme.

A eficácia do mecanismo foi validada por um período de doze meses em um teatro com quatrocentos assentos, acompanhado de quatro câmeras e iluminadores infravermelhos, onde vários filmes foram visualizados. Com este mecanismo também é possível, muitas vezes, predizer com sucesso dados ainda não coletados e antecipar as futuras reações do espectador no decorrer do filme a partir dos 5% primeiros dados observados. A Figura 39 apresenta um gráfico correlacionando momentos de sorriso (*smile*) e risada (*laugh*) com o uso de FVAE para o filme *Divertida Mente*. Pode-se ver que existem picos de humor durante o filme e que em alguns momentos as reações de sorriso e de risada são comuns.

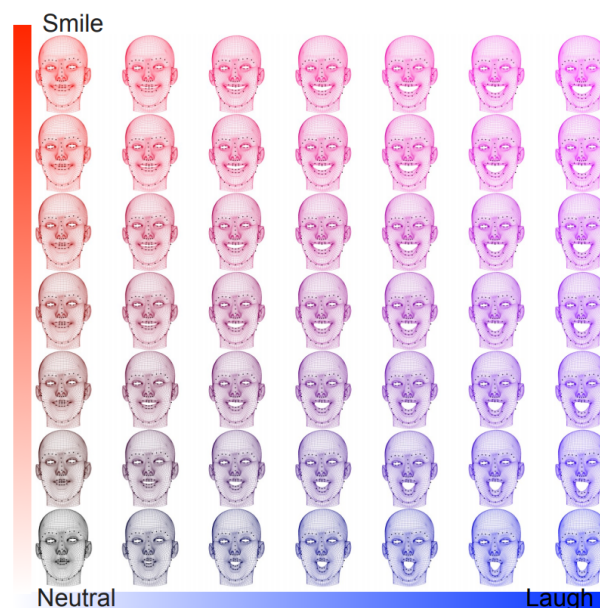
Figura 39 – Gráfico apresentando picos de humor no filme *Divertida Mente*



Fonte: Deng *et al.* (2017)

O mecanismo *Factorized Variational Autoencoders*, apresentado neste trabalho correlato, possui benefícios em relação a métodos de fatoração convencionais. Dentre suas características, destaca-se a flexibilidade similar à encontrada em sistemas de *deep learning* para descobrir relação entre dados não-lineares. A Figura 40 apresenta uma matriz relacionando o rosto humano com o respectivo nível da expressão facial, variando entre neutra, com sorriso e, por último, rindo.

Figura 40 – Relação de rostos com níveis de expressão facial



Fonte: Deng *et al.* (2017)

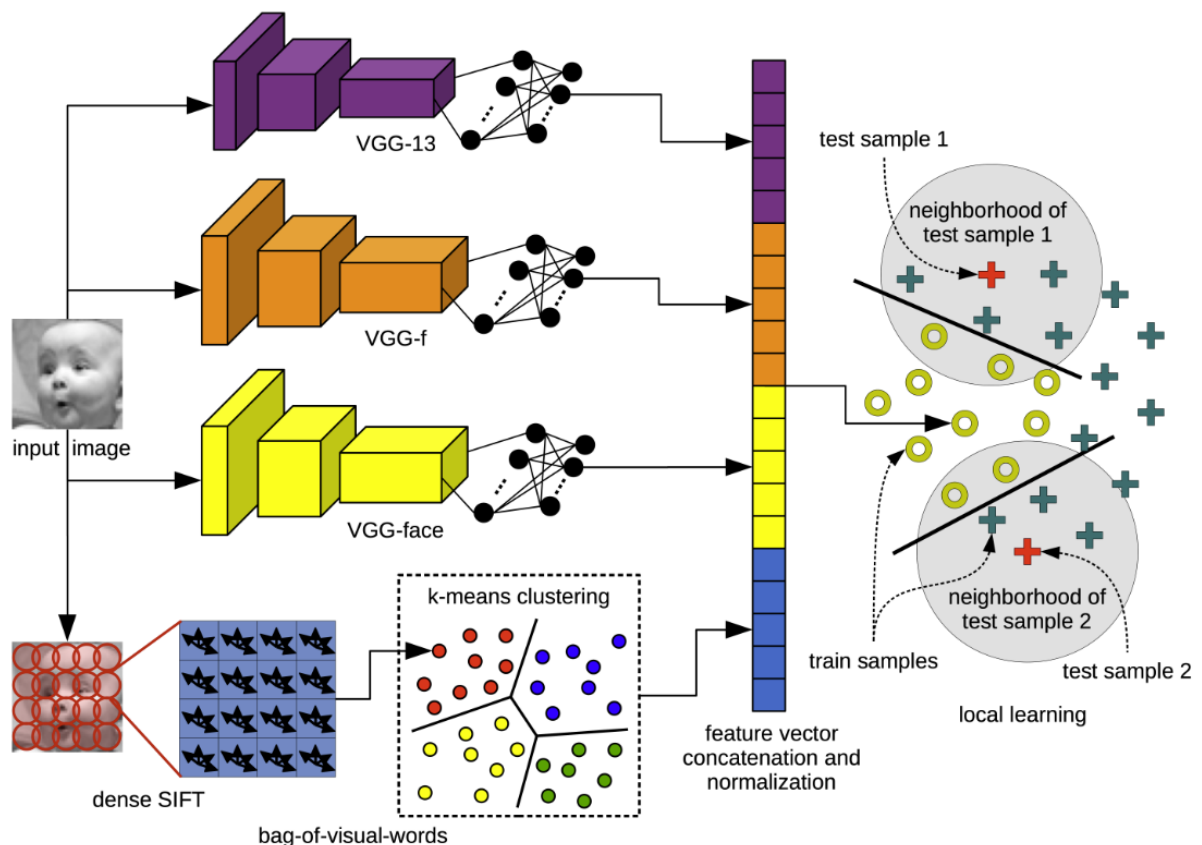
5.3 LOCAL LEARNING WITH DEEP AND HANDCRAFTED FEATURES FOR FACIAL EXPRESSION RECOGNITION

A pesquisa realizada por Georgescu, Ionescu e Popescu (2019) apresenta um mecanismo avançado que obtém 75,42% de acurácia com a base de dados FER-2013. Dentre os trabalhos correlatos pesquisados, considerando somente os que utilizam a base

de dados FER-2013, este é o que oferece os melhores resultados, com uma avançada arquitetura de rede neural que combina processos de convolução e *Support Vector Machine* (SVM) para efetuar a classificação final das expressões faciais. A Figura 41 apresenta uma visão geral da arquitetura desenvolvida por estes pesquisadores. Pode-se ver que esta arquitetura é composta por outras três arquiteturas de rede neural convolucional já apresentadas por outros pesquisadores. São elas: VVG-13, VGG-f e VGG-face, que foram suavemente modificadas para melhor se adequarem à pesquisa realizada por Georgescu, Ionescu e Popescu (2019).

Paralelo a isso, são aplicados mecanismos para extração de características com *handcrafted features*, que mesmo não fazendo uso de inteligência artificial geram resultados igualmente positivos. Ao final do processamento, as características encontradas por todos estes mecanismos são consolidadas e submetidas a uma *Support Vector Machine* para efetuar a classificação final, em busca do resultado estado da arte na detecção de expressões faciais com FER-2013.

Figura 41 – Visão geral da arquitetura estado da arte com FER-2013 Database



Fonte: Georgescu, Ionescu e Popescu (2019)

5.4 OUTROS TRABALHOS

Outros trabalhos também podem ser encontrados na área de detecção de expressões faciais. O estudo *A method of facial expression recognition based on Gabor and NMF*, proposto por Zhou *et al.* (2016), também é feito com base no *dataset* JAFFE. Tal trabalho chegou ao acerto de 98,1% das classificações através de métodos *Gabor Wavelet Transform*, fatoração de matrizes não-negativas e um classificador de duas camadas.

De forma similar, existe o estudo *Automatic facial emotion recognition using weber local descriptor for e-Healthcare system*, proposto por Alhussein (2016). Este estudo, assim como alguns dos demais anteriormente apresentados, também é feito com o *dataset* de expressões faciais JAFFE, além de também utilizar o *dataset* *Cohn–Kanade* (CK). É investigado principalmente o uso do algoritmo *Weber Local Descriptors* (WLD) que divide a imagem em inúmeros blocos, os quais são posteriormente delegados a uma *Support Vector Machine* (SVM) para a classificação final. A acurácia dos resultados alcançados para os *datasets* JAFFE e CK são, respectivamente, 97% e 98,82%.

O presente capítulo abordou e apresentou alguns dos trabalhos correlatos existentes, propostos por outros pesquisadores, na área que estuda e aprimora a detecção automática de expressões faciais. O próximo capítulo abordará o software de reconhecimento de expressões faciais desenvolvido para a presente pesquisa científica, assim como as tecnologias envolvidas e os resultados alcançados.

6 SOFTWARE PARA RECONHECIMENTO DE EXPRESSÕES FACIAIS

O presente capítulo apresentará em detalhes a rede neural construída para o reconhecimento de expressões faciais. Esta rede neural foi desenvolvida na linguagem Python fazendo uso das imagens contidas na base de dados FER-2013 para o processo de treinamento. Das bibliotecas utilizadas, Pandas é responsável por carregar as informações desta base de dados; NumPy, efetuar operações matemáticas nos pixels das imagens; por fim, TensorFlow e Keras, para efetuar o treinamento da rede neural propriamente dita. Este capítulo entrará em detalhes nestas tecnologias, assim como também apresentará como a estrutura de camadas da rede neural e os resultados alcançados.

6.1 TECNOLOGIAS

Esta seção apresentará as tecnologias usadas para a construção da rede neural. Dentre elas, destaca-se a linguagem Python e as bibliotecas TensorFlow, Keras, Pandas e Numpy.

6.1.1 Python

De acordo com Sarkar, Bali e Sharma (2017) e Chollet (2017), Python é uma linguagem de programação interpretada e baseada em indentação, criada no ano de 1991 por Guido van Rossum. Focada na simplicidade para a escrita do código-fonte, evoluiu muito com o passar dos anos e atualmente é enriquecida por um ecossistema de bibliotecas de alto-nível, que permitem manipular diferentes estruturas de dados e trabalhar com vários mecanismos de Inteligência Artificial. No entanto, o fato de ser interpretada a torna um pouco mais lenta do que outras linguagens, embora na prática isso não represente um problema. A razão disso é o fato de várias bibliotecas terem o *core* escrito em C e C++ e simplesmente disponibilizarem uma API para uso com Python, permitindo que seja obtido o melhor dos dois mundos: a simplicidade de escrita com Python e a alta performance obtida com o *core* de bibliotecas escritas em linguagens de mais baixo-nível.

6.1.2 TensorFlow e Keras

TensorFlow é uma biblioteca *open source* inicialmente proposta e desenvolvida por pesquisadores da *Google Research* para treinamento de modelos com IA. Amplamente conhecida e divulgada, esta biblioteca também disponibiliza *wrappers* e APIs que possibilitam seu uso através de outras linguagens, como Python, JavaScript e Swift. A biblioteca TensorFlow também é capaz de abstrair o uso da tecnologia NVIDIA CUDA® para fazer

uso intenso da placa de vídeo a fim de efetuar cálculos matemáticos de forma paralela e com alta performance (ABADI *et al.*, 2015) (NVIDIA, 2019).

Keras, por sua vez, é um *framework* de *Deep Learning* construído sob o próprio TensorFlow com o intuito de oferecer uma versão simplificada de sua API, tornando mais ágil a experimentação de modelos e diminuindo a curva de aprendizagem para desenvolvedores iniciantes. Este *framework* também permite o uso de outras bibliotecas de IA por meio de sua mesma API minimalista, como é o caso das bibliotecas Theano e CNTK, classificadas como bibliotecas de *Backend*. Isto significa que é possível trocar a implementação *Backend* sem necessariamente alterar o código que faz uso da API oferecida pelo Keras (CHOLLET, 2017). Algumas das empresas que fazem uso destas tecnologias são NASA, Coca-Cola, DeepMind, Intel, Uber, entre várias outras (CHOLLET *et al.*, 2015). A Figura 42 apresenta camadas para a construção de um software com Keras e TensorFlow. Nela, destaca-se o fato de que Keras faz uso de um *Backend* para fazer o processamento da Inteligência Artificial, que pode usar bibliotecas baixo-nível para interação direta com GPU e CPU (CHOLLET, 2017).

Figura 42 – Camadas de *hardware* e *software* situando bibliotecas de IA



Fonte: Chollet (2017)

6.1.3 NumPy

De acordo com Oliphant (2006), a biblioteca NumPy tem como foco a computação científica visando simplificar operações matemáticas em Python. Além de aplicações para Inteligência Artificial, que é o caso da presente pesquisa, esta biblioteca também pode ser usada para várias outras finalidades que exijam manipulação de expressões numéricas. Seu site oficial aponta que NumPy foi inclusive usada para grandes descobertas científicas, como a geração da primeira imagem de um buraco negro (NUMPY, 2020). A API pública das bibliotecas Keras e TensorFlow já espera dados em formato NumPy. Desta forma, para trabalhar com estas bibliotecas de Inteligência Artificial, é necessário que os dados tenham sido pré-processados e convertidos para objetos NumPy.

6.1.4 Pandas

Pandas é uma biblioteca para manipulação de *datasets* em Python, oferecendo uma API de alto-nível para solucionar problemas práticos do dia a dia relacionados à modelos estatísticos. Pandas oferece diversos métodos e funções para carregar dados em memória, de forma que para a presente pesquisa seja usado para carregar o arquivo em formato CSV (*Comma-separated values*) com pixels de rostos humanos para o treinamento da rede neural. Esta biblioteca traz à linguagem Python diversas facilidades encontradas na linguagem R, aumentando a popularidade de Python com pesquisas relacionadas à *Machine Learning* (MCKINNEY, 2010).

6.1.5 OpenCV

De acordo com Bradski (2000), OpenCV é uma biblioteca *open source* para visão computacional que oferece centenas de funções para a manipulação de imagens em 2D e 3D. Esta biblioteca tem a maior parte do seu *core* escrito em C++ e oferece APIs e *wrappers* para outras linguagens, como Python e Java. Esta biblioteca é muitas vezes utilizada empregando um classificador pré-treinado do tipo *Haar Cascade* que detecta regiões de objetos em imagens. Estas regiões, denominadas regiões de interesse ou ROI (*Regions of Interest*), representam a área na qual a rede neural efetivamente fará a análise, pois em um software de detecção de expressões faciais somente a área do rosto humano deve ser considerada. Para a presente pesquisa, a biblioteca OpenCV foi utilizada para fins de teste, exibindo algumas das imagens do *dataset* antes de efetivamente iniciar o processo de treinamento da rede neural. Isto garante que as mesmas sejam adequadamente carregadas para a memória RAM e que estão no formado correto para o processamento em etapas seguintes.

6.1.6 Preparação do ambiente

Algumas dificuldades foram enfrentadas ao preparar o ambiente de execução da rede neural com as tecnologias apresentadas, principalmente no que diz respeito ao versionamento das bibliotecas Keras e TensorFlow. Para o correto funcionamento do código-fonte escrito em Python, sugere-se a instalação das bibliotecas apresentadas na Tabela 2, exatamente com as versões indicadas. O código-fonte está disponível no GitHub¹. A versão da linguagem Python utilizada foi a 3.8.2, assim como as dependências necessárias para a criação da rede neural foram instaladas com o gerenciador de dependências *pip3*. Esta versão de Python já vem pré-instalada no sistema operacional Ubuntu 20.04.1 LTS. A instalação do *pip3* pode ser feita no Ubuntu com o comando *sudo apt-get install python3-pip*, ao mesmo tempo em que a instalação das bibliotecas pode ser feita através de comandos *pip3 install <biblioteca>==<versão>*.

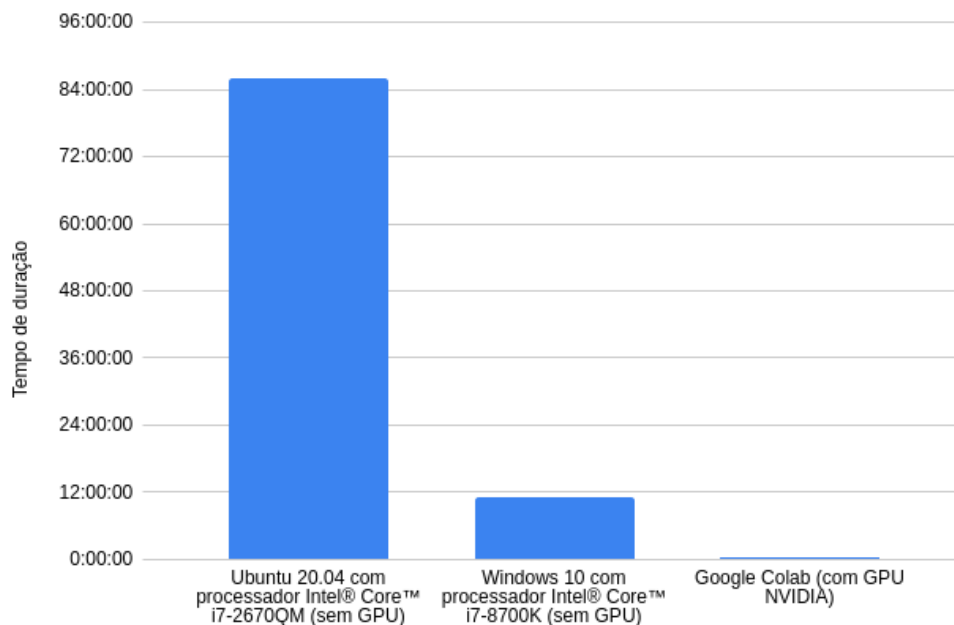
¹ Repositório pode ser acessado no link: github.com/GustavoASC/facial-expression-cnn

Tabela 2 – Bibliotecas para execução da rede neural

Biblioteca	Versão
tensorflow	2.3.0
numpy	1.19.1
Keras	2.0.0
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.2
matplotlib	3.2.2
scikit-learn	0.22.2.post1
pandas	1.0.5
opencv-python	4.1.2.30

Fonte: do autor

No entanto, mesmo com a correta configuração das versões das bibliotecas, o tempo para treinar a rede neural ainda era muito alto e levava inúmeras horas. Era inviável realizar muitos testes sem uma placa de vídeo NVIDIA com suporte à tecnologia CUDA. Diante disso, após várias buscas, foi encontrada a ferramenta Google Colab, que permite treinar na nuvem, com altíssima performance, *scripts* Python com bibliotecas Keras e TensorFlow. O tempo levado pelos testes diminuíram de várias horas para simples minutos, visto que o Google Colab oferece suporte à GPUs NVIDIA. De acordo com Google (2020a), a ferramenta Colabority ou Colab "não requer nenhuma configuração para usar e oferece acesso gratuito a recursos de computação como GPUs".

Figura 43 – Tempo para treinar a mesma rede neural em diferentes máquinas

Fonte: do autor

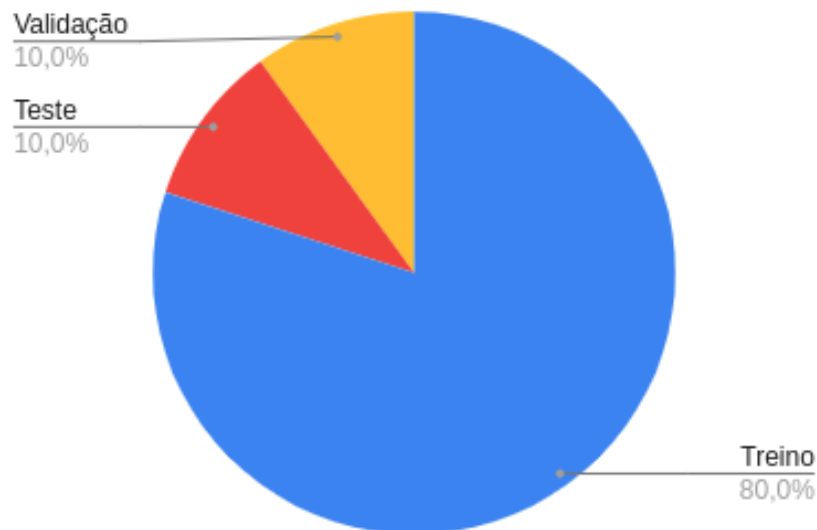
A Figura 43 compara o tempo de duração para treinar a rede neural em diferentes

máquinas. Quanto menor o tempo, melhor. A primeira coluna, da esquerda, indica o tempo de processamento em uma máquina que não faz uso de GPU e possui um processador menos potente do que as outras máquinas, resultando em um tempo de processamento muito maior. A segunda máquina, apresentada pela coluna intermediária, também não faz uso de GPU mas possui um processador mais moderno, diminuindo consideravelmente o tempo de processamento. Por fim, com a ferramenta Google Colab o tempo é incrivelmente menor, visto que faz uso de GPU NVIDIA e, portanto, consegue treinar a rede neural com muita performance e de forma paralela.

6.2 BASE DE DADOS UTILIZADA

A base de dados utilizada para o treinamento da rede neural foi a FER-2013 *Database*, previamente apresentada no capítulo 3.2.4. Esta base de dados foi a escolhida pelo fato de oferecer um número muito maior de imagens do que as demais bases apresentadas, chegando a aproximadamente 35 mil imagens com expressões faciais já rotuladas. Desta base de dados, 80% das imagens foram usadas para o treinamento, 10% foram usadas para teste e 10% foram utilizadas para validação, conforme pode ser visto na Figura 44.

Figura 44 – Percentual de imagens da base de dados por categoria



Fonte: do autor

6.3 REDES NEURAIIS CONSTRUÍDAS E AVALIADAS

Diversos testes foram realizados a fim de alcançar resultados positivos na detecção de expressões faciais. Para isso, várias arquiteturas de rede neural foram projetadas e testadas, assim como alguns mecanismos de pré-processamento no brilho das imagens foram avaliados. A Tabela 3 apresenta a visão geral de algumas destas redes e relaciona com a respectiva acurácia obtida na base de dados de validação. Nesta tabela, pode-se ver

que ainda não foram aplicados mecanismos de *Image Augmentation* para gerar variações das imagens de entrada.

Tabela 3 – Resultados encontrados na base de validação sem *Image Augmentation*

Acurácia	Visão geral da rede
64,92%	Rede com 8 camadas convolucionais e função de ativação <i>relu</i> , intercaladas por camadas <i>BatchNormalization</i> , <i>MaxPooling2D</i> e <i>Dropout</i> de 50% dos neurônios ativos.
60,40%	Rede com 4 camadas convolucionais e função de ativação <i>relu</i> , com as duas primeiras extraindo 64 características das imagens e as duas últimas camadas convolucionais extraindo 128 características.
25,94%	Rede com 3 camadas convolucionais e função de ativação <i>relu</i> , sendo que a primeira camada extrai 64 características das imagens e as duas seguintes extraem 512 características.
66,22%	Rede com 8 camadas convolucionais e função de ativação <i>relu</i> , aumentando gradativamente o número de características detectadas em cada camada. O número de características inicia em 64 e aumenta até chegar em 512 características na última camada convolucional. Também são intercaladas por camadas <i>BatchNormalization</i> , <i>MaxPooling2D</i> e <i>Dropout</i> de 50% dos neurônios ativos. Todos os pixels da imagem passam por um pré-processamento que diminui o brilho em um valor de 30.
66,13%	Rede com 7 camadas convolucionais e função de ativação <i>relu</i> , aplicando <i>Dropout</i> de 75% dos neurônios.
61,18%	Rede com 7 camadas convolucionais e função de ativação <i>relu</i> . O tamanho do <i>kernel</i> na primeira camada é de 7 x 7, diferentemente das redes anteriores, que iniciam com <i>kernel</i> de tamanho 3 x 3. Também é aplicado <i>Dropout</i> de 40% dos neurônios, seguido de um segundo <i>Dropout</i> de 75% após algumas camadas.
65,85%	Muito parecida com a rede anterior, com a diferença de que possui um total de 10 camadas de convolução.

Fonte: do autor

Já os dados apresentados na Tabela 4 oferecem resultados promissores e melhores do que os resultados apresentados na tabela anterior. Pode-se ver que além de alguns ajustes nas camadas da rede neural também é utilizada técnica de *Image Augmentation*, a fim de aumentar o número de imagens utilizadas no processo de treinamento. O melhor resultado obtido dentre todos os testes foi de 68,48% com a FER-2013 *Database*.

Tabela 4 – Resultados encontrados na base de validação com *Image Augmentation*

Acurácia	Visão geral da rede
66,53%	Introduz a aplicação de <i>Image Augmentation</i> para gerar variações nas imagens de entrada e possui um total de 10 camadas convolucionais. Diferencia-se no tamanho do <i>kernel</i> , iniciando com um <i>kernel</i> de tamanho 7 x 7 que gradativamente diminui para 4 x 4, 3 x 3 e 2 x 2.
68,11%	Rede com 5 camadas convolucionais e função de ativação <i>relu</i> . Também faz uso de <i>Image Augmentation</i> . A primeira camada convolucional possui um <i>kernel</i> de tamanho 7 x 7 e as demais camadas possuem um <i>kernel</i> de tamanho 4 x 4. É aplicado um <i>Dropout</i> de 20% dos neurônios.
68,48%	Muito parecida com a rede anterior, possuindo o mesmo número de camadas. A principal diferença está na quantidade de características extraídas em cada camada, sendo respectivamente: 32, 50, 100, 150 e 200 características. É a rede neural que apresentou os melhores resultados dentre todos os testes realizados. Também foram feitos testes passando as imagens do FER-2013 <i>Database</i> por uma etapa adicional de pré-processamento, com o intuito de ajustar o brilho das imagens. No entanto, percebeu-se que este pré-processamento não surtiu o efeito desejado, sendo descartado para simplificar a arquitetura. Este pré-processamento tornava mais claros os pixels maiores do que um <i>threshold</i> (limiar) de 127, o qual foi escolhido após uma série de testes. Da mesma forma, tornava mais escuros os pixels menores do que este mesmo <i>threshold</i> .

Fonte: do autor

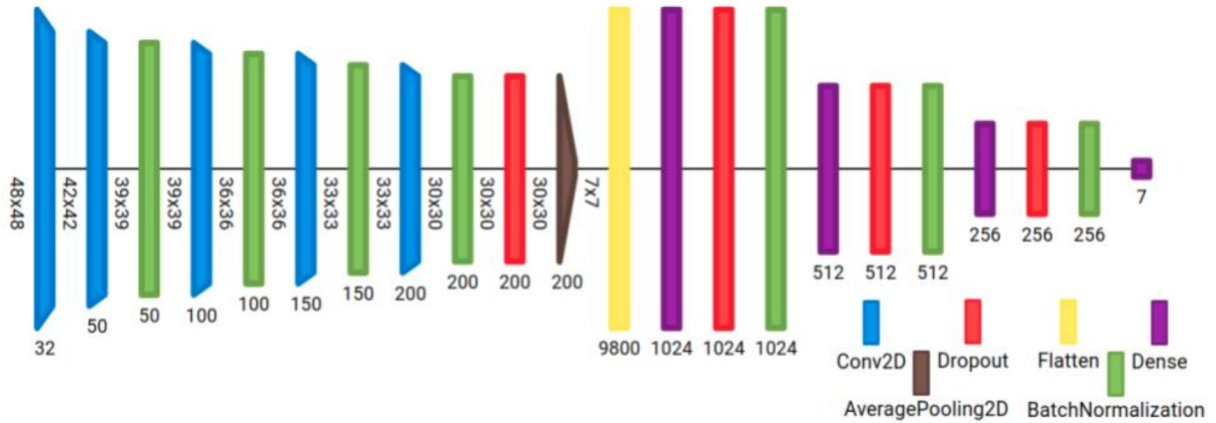
6.3.1 Arquitetura da rede neural com a melhor acurácia

Dadas as redes neurais apresentadas anteriormente, a presente subseção detalhará a arquitetura da rede que obteve a melhor acurácia, de 68,48%, com a FER-2013 *Database*. Técnicas de *Image Augmentation* também foram utilizadas para aumentar o número final de imagens utilizadas no processo de aprendizagem. Estas técnicas geram novas imagens a partir das já existentes na base de dados, aplicando mecanismos de rotação para que fiquem em ângulos diferentes, além da aplicação de *zoom* para que aplique foco em diferentes regiões das imagens.

A rede neural construída contém 11.531.019 neurônios e um total de 22 camadas, conforme pode ser visto na Figura 45. A representação desta figura foi gerada com a ferramenta Net2Vis, proposta por Bäuerle e Ropinski (2019), capaz de gerar representações visuais das camadas de uma rede neural convolucional a partir de código com API Keras. O primeiro grupo de camadas consiste em uma sucessão de 5 convoluções, responsáveis por extrair as características mais básicas das imagens, intercaladas por camadas de *Batch Normalization* que normalizam os dados e evitam o *overfitting*. A camada de entrada na rede convolucional recebe as imagens no formato original encontrado na base de dados, ou

seja, em matrizes de 48 x 48 posições, além de ser a única que não é seguida diretamente de uma camada *Batch Normalization*.

Figura 45 – Estrutura de camadas da rede neural desenvolvida



Fonte: do autor

Uma vez realizado o processamento neste primeiro grupo de camadas é acionada a camada *Dropout*, destacada na cor vermelha, que zera os valores e pesos de 20% dos neurônios já treinados até então. Esta camada também tem o objetivo de minimizar a memorização dos dados de treinamento e assim evitar o *overfitting*.

Em um terceiro momento é aplicada a camada *AveragePooling2D* para diminuir as dimensões da matriz, de 30 x 30 para 7 x 7, a fim de minimizar o processamento nas camadas seguintes. Esta etapa gera um conjunto condensado de informações das imagens. Após esta camada é iniciada a *Flatten*, que transforma as matrizes em vetores de dimensão única. Isto permite que os dados sejam classificados por camadas totalmente conectadas, ou *Fully Connected Layers*. Uma vez que os dados passam pela *Flatten*, são encontradas três sucessões de camadas *Dense*, *Dropout* e *Batch Normalization* para treinar os neurônios de forma tradicional com os dados já em uma única dimensão. Neste momento não são usadas técnicas convolucionais.

Até o presente momento, tanto as camadas convolucionais e camadas *Dense* fizeram uso da função de ativação ReLU, que demonstrou um resultado positivo durante o treinamento. A última camada, também de categoria *Dense*, é a única que usa a função de ativação *Softmax* ao invés de ReLU, a fim de encontrar a resposta (expressão facial) que possui a maior probabilidade de ser a correta. Esta camada possui um número de 7 neurônios, uma vez que a rede tem por objetivo classificar a imagem entre uma de 7 expressões faciais.

6.3.2 *Callbacks* configurados para a rede neural

A API do Keras permite configurar *callbacks* executados ao fim de cada época do treinamento. Estes *callbacks* fazem um monitoramento dos valores calculados para a rede neural, possibilitando alterações na taxa de aprendizagem ou então que o processo de treino seja finalizado antecipadamente, por exemplo. Existem diversos *callbacks* que podem ser configurados e os destacados a seguir foram utilizados no software desenvolvido.

- **ReduceLRonPlateau:** também conhecido como *Reduce Learning Rate on Plateau*, é responsável por modificar a taxa de aprendizagem caso os resultados não obtenham evolução após um número pré-determinado de épocas, monitorando o erro na base de dados de validação. A taxa de aprendizagem é suavemente alterada caso o valor do erro não diminua após X épocas consecutivas. O número de épocas que deve aguardar até que o ajuste seja aplicado é definido no parâmetro *patience*.
- **EarlyStopping:** permite que a rede neural finalize antecipadamente o processo de treinamento caso a taxa de erro não diminua após um número pré-determinado de épocas, também definido pelo parâmetro *patience*. Este *callback* é útil para economizar processamento e também para evitar *overfitting*.
- **ModelCheckpoint:** este *callback* permite salvar em disco o melhor estado encontrado pela rede neural através do monitoramento da taxa de erro na base de validação. Sempre que esta taxa diminuir o resultado será salvo em disco, o que na prática significa que a rede neural ficou mais inteligente e está convergindo para melhores resultados. Existe, também, o parâmetro *save_best_only* para sobrescrever o arquivo sempre que um resultado melhor for encontrado, ao invés de salvar uma nova cópia com nome diferente.

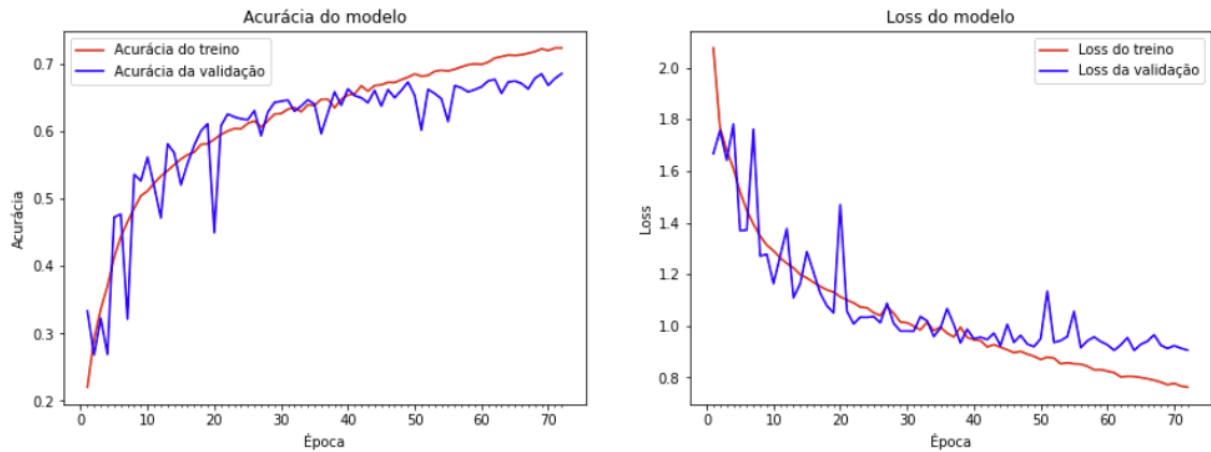
6.4 RESULTADOS

A presente seção apresenta resultados detalhados referente ao processo de treinamento da rede neural com a FER-2013 *Database*. É importante frisar que somente os resultados da rede neural com a melhor acurácia foram avaliados em detalhes e serão apresentados nos parágrafos seguintes.

Os gráficos apresentados na Figura 46 indicam, respectivamente, a taxa de crescimento da acurácia e a taxa de redução do custo (*loss*), tanto na base de treino quanto na base de validação. Percebe-se, no gráfico da esquerda, que a acurácia na base de treino aumenta suavemente sem picos nem quedas nos valores. Por outro lado, a acurácia na base de validação é diferente: em vários momentos sofre quedas e picos marcantes ao longo das 72 épocas. A melhor acurácia para a base de validação foi encontrada na última época, alcançando o valor máximo de 68,48%. De forma similar, o gráfico da direita apresenta o

custo da rede neural durante as mesmas 72 épocas de treinamento. Nas últimas épocas é possível verificar o *overfitting*, ou seja, um estado em que o custo na base de treino continua diminuindo enquanto o custo na base de validação não diminui. O treinamento é encerrado tão logo quanto o algoritmo detecta o *overfitting*. O menor custo para a base de validação foi encontrado na época 64, com um valor mínimo de 0.90507.

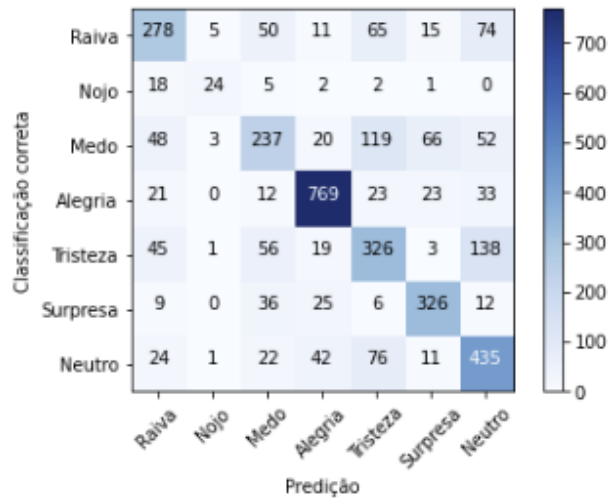
Figura 46 – Gráficos de acurácia e custo (*loss*) da rede neural



Fonte: do autor

Já a Figura 47 apresenta a Matriz de Confusão para a base de dados de teste, ou seja, 10% da base de dados original que não foi utilizada no processo de treinamento, representando um total de 3.589 imagens. A Matriz de Confusão indica quantos acertos e erros a rede neural teve para cada expressão facial, deixando explícito qual é a expressão que mais precisa ser aprimorada e qual já está apresentando resultados satisfatórios. Pode-se ver que a expressão de alegria teve um resultado muito positivo, visto que praticamente não houve erro.

Figura 47 – Matriz de Confusão para os dados de teste

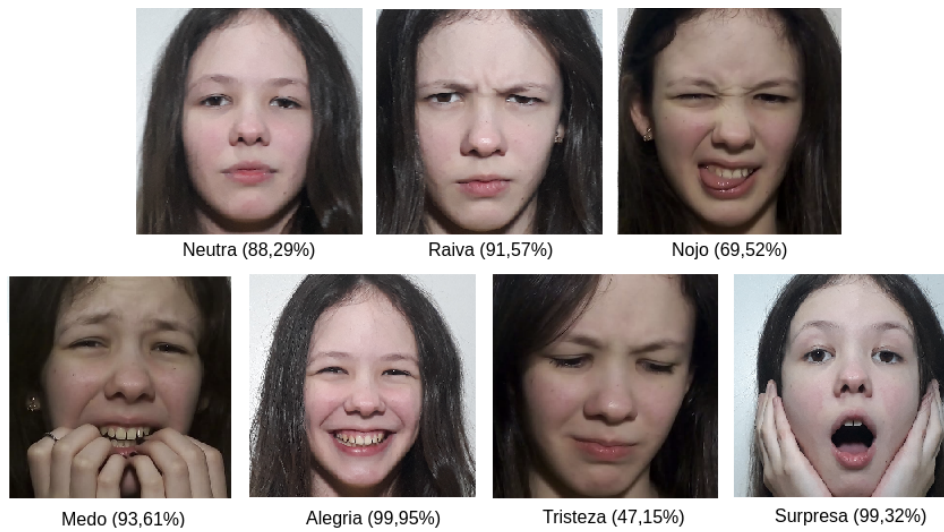


Fonte: do autor

O mesmo não se aplica para a expressão facial de medo, que foi bastante confundida com as expressões de raiva e tristeza. O inverso também é verdade: as expressões de raiva e tristeza também se confundiram entre si, indicando que a arquitetura da rede neural ainda precisa ser aprimorada e/ou são necessárias mais imagens no processo de treinamento para que a rede neural consiga detectar adequadamente as características destas expressões.

A Figura 48 apresenta um conjunto de imagens adicionais que foram especialmente capturadas com uma modelo e submetidas à análise da rede neural. Pode-se ver que as expressões de alegria e surpresa foram detectadas com a altíssima acurácia de aproximadamente 99%. Outras, como neutra, raiva e medo, também foram detectadas com a alta acurácia de aproximadamente 90%, embora menor do que as duas expressões previamente citadas. A expressão de nojo também alcançou o resultado positivo de 69,52% de acerto, embora a rede neural tenha se confundido com a expressão de raiva. Acredita-se que isto ocorra por conta das rugas entre os olhos, o que é muito característico da expressão de raiva. A expressão de tristeza foi identificada corretamente mas com uma taxa de acerto de apenas 47,15%, pelo fato da rede neural também distribuir o resultado entre as demais expressões faciais e apontar 34,85% como chance de ser a expressão de medo, estando de acordo com os resultados apresentados na Matriz de Confusão.

Figura 48 – Imagens capturadas com uma modelo para verificar a eficácia da rede neural



Fonte: do autor

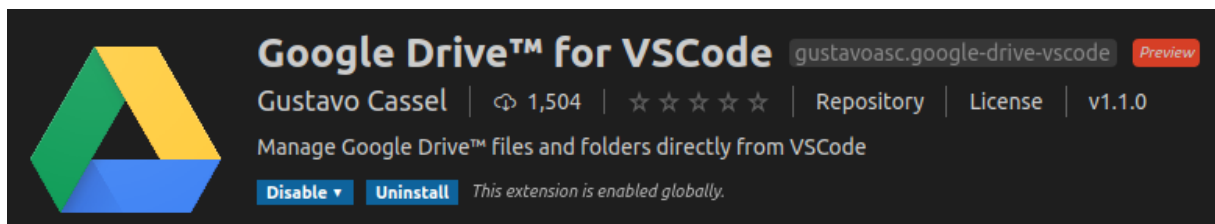
O presente capítulo apresentou em detalhes as tecnologias utilizadas para a construção da rede neural, assim como detalhes da arquitetura que obteve os melhores resultados e uma visão geral de outras arquiteturas que também foram testadas ao longo das semanas. O atual estado da arte, proposto por Georgescu, Ionescu e Popescu (2019), alcançou 75,42% de acurácia com a mesma base de dados FER-2013. Não foi possível ultrapassar este resultado com a rede neural proposta, que chegou ao resultado de 68,48% na base de dados de validação, conforme apresentado nos parágrafos anteriores. O capítulo seguinte apresentará uma extensão desenvolvida para o editor de programação Visual Studio Code no decorrer deste Trabalho de Conclusão de Curso.

7 EXTENSÃO DESENVOLVIDA PARA O VISUAL STUDIO CODE

O Visual Studio Code, comumente conhecido como VSCode, é um editor de programação de código aberto, mantido pela empresa Microsoft com a licença MIT no GitHub¹. Este editor recebe ativamente muitas contribuições espontâneas da comunidade pelo fato de ter o código-fonte aberto. De acordo com a pesquisa realizada pelo website Stack Overflow no ano de 2019, foi o editor de programação mais utilizado dentre as pessoas que responderam o questionário, chegando ao percentual de 50,7% (OVERFLOW, 2019). Este editor também oferece uma API (*Application Programming Interface*) que permite expandir suas funcionalidades e possibilitou o desenvolvimento de um grande ecossistema de extensões (CODE, 2020).

Durante o desenvolvimento deste Trabalho de Conclusão de Curso, além do software para reconhecimento de expressões faciais, também foi criada a extensão *Google Drive™ for VSCode* que permite integrar o editor de programação VSCode ao Google Drive. Tal extensão já possui mais de 1.500 *downloads* no Marketplace, sendo usada por diversas pessoas ao redor do mundo. O que motivou seu desenvolvimento foi a necessidade de efetuar *backups* de maneira simples, rápida e fácil de todos os documentos, imagens e arquivos de código-fonte relacionados ao Trabalho de Conclusão de Curso. A Figura 49 apresenta a capa desta extensão quando instalada no VSCode, assim como o número total de instalações que a extensão já teve até o presente momento.

Figura 49 – Extensão instalada no Visual Studio Code



Fonte: do autor

Por sua vez, a Google (2020b) aponta que o Google Drive é um serviço web que oferece armazenamento de arquivos e pastas na nuvem, permitindo que os dados do usuário sejam acessados por diferentes máquinas e dispositivos conectados na mesma conta Google. Este serviço também oferece uma API REST que permite a aplicativos de terceiros acessarem e manipularem os dados do usuário diretamente na nuvem, desde que o usuário conceda as permissões necessárias ao aplicativo. A empresa Google também

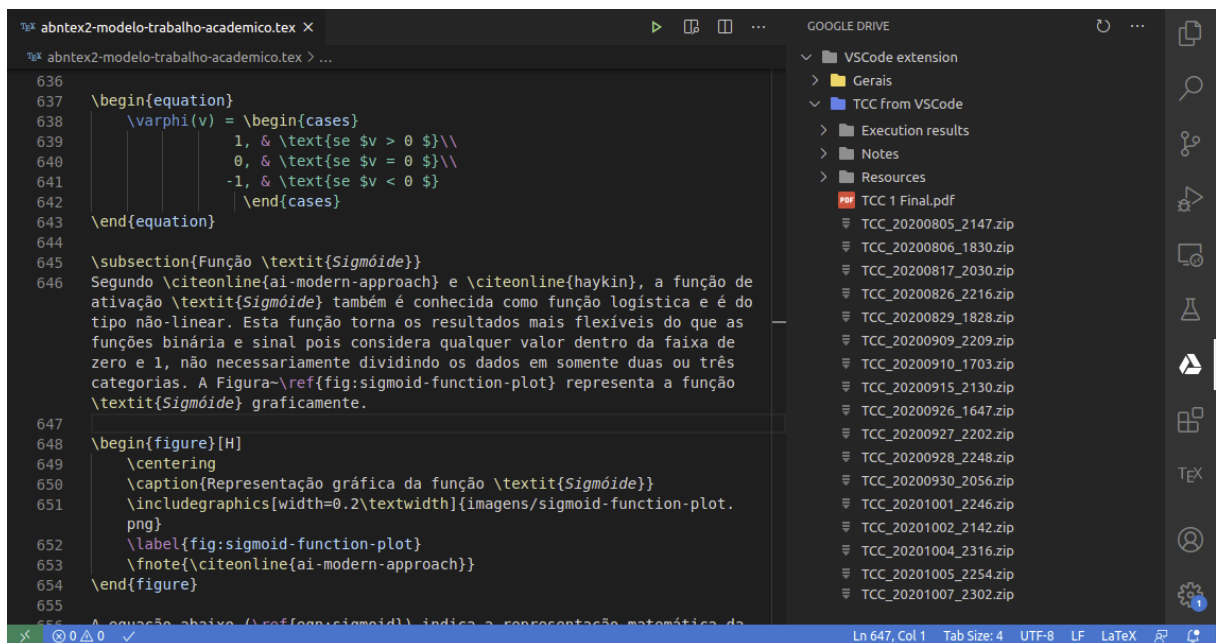
¹ Código-fonte do VSCode disponível em: github.com/microsoft/vscode

oferece implementações Java, Node.JS e Python que simplificam o uso desta API REST, abstraindo as requisições HTTP em classes, métodos e funções.

A extensão *Google Drive™ for VSCode* foi escrita com a linguagem de programação TypeScript e faz uso da implementação Node.JS da Google Drive API, além de também disponibilizar todo o seu código-fonte no GitHub². Isto possibilita que a comunidade contribua com novos recursos e correção de possíveis problemas. A extensão tem como objetivo oferecer experiência similar à obtida através do website do Google Drive, porém de forma integrada ao VSCode. No momento, alguns dos recursos disponíveis são: pré-visualização do conteúdo de arquivos na nuvem, *upload* e *download* de arquivos, criação de pastas remotas e possibilidade de renomear arquivos remotos. Outro recurso interessante é que arquivos em formato texto são pré-visualizados com colorização e tema configurados pelo usuário, herdando recursos como *syntax highlight* caso seja um arquivo de código-fonte.

A extensão também é capaz de compactar todo o *workspace* aberto no editor de programação e enviá-lo para a nuvem, considerando recursivamente todas as subpastas e seus respectivos arquivos filhos. A data/hora atual é concatenada ao nome deste arquivo compactado, permitindo saber exatamente quando o *upload* foi feito. Desta maneira, pode-se facilmente ter um histórico de todas as versões salvas do Trabalho de Conclusão de Curso com todos os arquivos relacionados, além de permitir que ele seja facilmente acessado em mais de uma máquina pelo fato de estar na nuvem.

Figura 50 – VSCode exibindo o TCC em formato LaTeX e também arquivos da nuvem



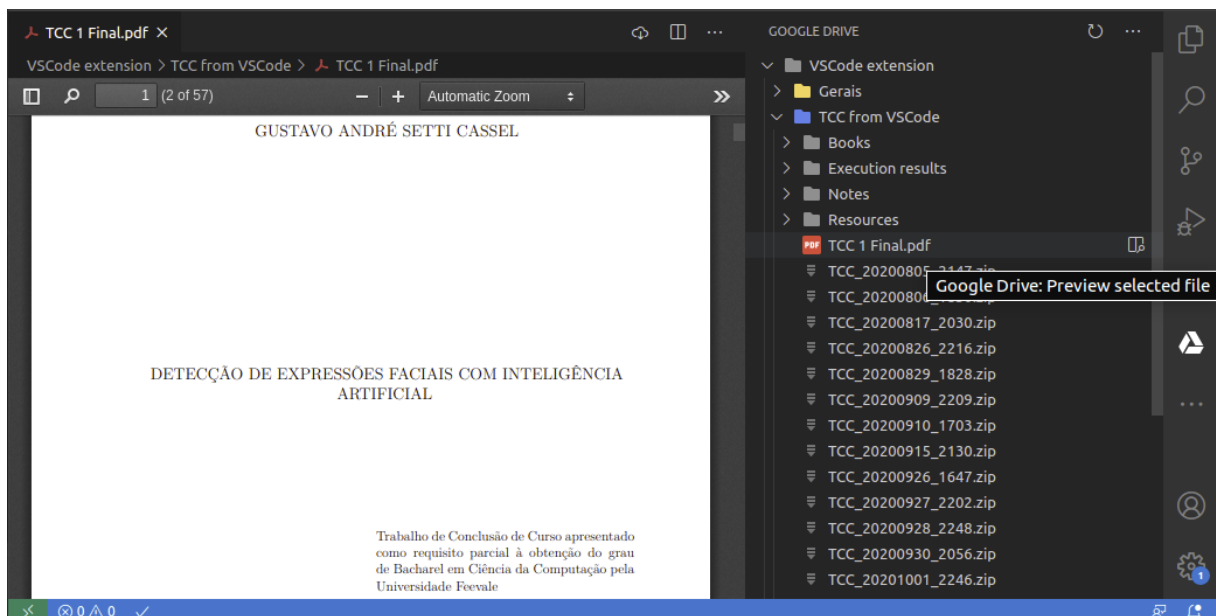
Fonte: do autor

² Código-fonte da extensão disponível em: github.com/GustavoASC/google-drive-vscode

A Figura 50 apresenta o VSCode sendo utilizado para escrever este Trabalho de Conclusão de Curso. Pode-se ver que, na parte esquerda da imagem, é apresentado o documento deste TCC em formato LaTeX. Ao mesmo tempo, na parte direita da imagem, a extensão apresenta em formato de árvore os arquivos e pastas encontrados na nuvem do Google Drive.

A extensão também permite pré-visualizar o conteúdo de arquivos diretamente na nuvem, sem a necessidade de baixá-los fisicamente no computador. Este recurso torna a experiência de programação mais ágil, integrada e agradável. Supondo que o desenvolvedor de software queira visualizar o conteúdo de um arquivo PDF, basta clicar no ícone de lupa a fim de que o arquivo seja visualizado no editor de programação. Este comportamento é apresentado na Figura 51, onde um documento em formato PDF é pré-visualizado diretamente na nuvem.

Figura 51 – VSCode exibindo documento em formato PDF diretamente da nuvem



Fonte: do autor

Este capítulo abordou uma visão geral do editor de programação VSCode, assim como apresentou a extensão *Google DriveTM for VSCode* e detalhou alguns aspectos da API de integração com o Google Drive. O capítulo seguinte apresentará a conclusão final deste Trabalho de Conclusão de Curso.

8 CONCLUSÃO

A Inteligência Artificial é um nicho de pesquisa promissor que pode ser usado em diversos ramos da computação; em especial, aqueles que abordam predição de classes e detecção de padrões. Este ramo de pesquisa engloba não somente a computação, mas também outras áreas, como Neurociência, evidenciando a importância da interdisciplinaridade. Como pôde ser visto na presente pesquisa, foi possível alcançar a acurácia de 68,48% na detecção de expressões faciais, através de uma arquitetura de rede neural convolucional com a base de dados FER-2013. Evidentemente, existem diversos aspectos a serem aprimorados e aperfeiçoados, mas os resultados já são muito animadores mesmo não ultrapassando o estado da arte de 75,42% dentre as pesquisas que utilizam a mesma base de dados FER-2013.

O uso da metodologia *Design Science Research* (DSR) se mostrou muito conveniente para a organização deste Trabalho de Conclusão de Curso. Com ela, foi possível nortear a busca por trabalhos correlatos e formalizar as etapas principais da pesquisa, além de auxiliar na definição do escopo do software desenvolvido. Esta metodologia também permitiu refletir acerca da divulgação dos resultados para a comunidade científica, dando abertura para a publicação na Feira de Iniciação Científica 2020 da Universidade Feevale.

Também foi muito instigante conhecer algumas das diferentes bases de dados voltadas à detecção de expressões do rosto humano. Algumas delas focam em pessoas de uma determinada região geográfica, como por exemplo, o Japão; outras, apresentam faces de pessoas de diferentes partes do mundo. Isto permitiu aprofundar conhecimentos na linguagem Python envolvendo diferentes maneiras de carregar *datasets* para a memória RAM. Pôde-se ver que alguns *datasets* tratam cada imagem como um arquivo físico diferente no sistema operacional, ao mesmo tempo em que outros *datasets*, como o FER-2013, disponibilizam todos os pixels das imagens em um único arquivo CSV (*Comma-separated values*).

Quanto ao processo de treinamento, pode-se concluir que é surpreendentemente possível treinar redes neurais complexas mesmo com máquinas modestas que não possuam placa de vídeo (GPU). Isto é possível graças à ferramenta Google Colab, que oferece recursos em nuvem para execução de *scripts* Python com bibliotecas de Inteligência Artificial, sem depender de recursos da máquina do pesquisador. Utilizar esta ferramenta foi importantíssimo para o bom andamento deste Trabalho de Conclusão de Curso. Caso contrário, treinar a rede neural levaria muito mais tempo e provavelmente não seria possível chegar aos mesmos resultados, devido ao prazo limitado de entrega.

Por fim, para pesquisas futuras, indica-se o uso de base de dados com número ainda maior de imagens, principalmente nojo e tristeza. Também sugere-se o uso de mais imagens de medo no processo de treinamento, onde a pessoa não esteja com a mão na boca, para que seja possível alcançar melhores resultados em cenários como esse. É possível, também, fazer uma análise dos resultados através da curva ROC (*Receiver Operating Characteristics*) e da métrica AUC (*Area Under the Curve*).

A presente pesquisa científica contribuiu para aprimorar os conhecimentos nos ramos de Inteligência Artificial e *Deep Learning*. Espera-se que futuras pesquisas façam proveito dessas conclusões e que pesquisadores tomem conhecimento da ferramenta Google Colab, a fim de agilizar o treinamento de suas redes neurais.

REFERÊNCIAS

- ABADI, M. *et al.* *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: <<https://www.tensorflow.org/>>. Citado na página 57.
- AGARAP, A. F. Deep learning using rectified linear units (relu). *CoRR*, abs/1803.08375, 2018. Disponível em: <<http://arxiv.org/abs/1803.08375>>. Citado na página 36.
- ALHUSSEIN, M. Automatic facial emotion recognition using weber local descriptor for e-healthcare system. *Cluster Computing*, v. 19, jan. 2016. Citado 2 vezes nas páginas 13 e 55.
- AMICO, F. *et al.* Multimodal validation of facial expression detection software for real-time monitoring of affect in patients with suicidal intent. *European Psychiatry*, v. 33, p. 596, mar. 2016. Citado na página 13.
- AMOR, N. B. Towards a dynamic deployment strategy of wheelchair command applications on heterogeneous architecture. *Journal of Information Assurance and Security*, v. 11, p. 117–125, mar. 2016. Citado na página 51.
- ASSOCIATION, B. N. *et al.* *Neuroscience: Science of the Brain: an Introduction for Young Students*. Liverpool, UK: British Neuroscience Association, 2003. ISBN 9780954520403. Citado na página 31.
- BÄUERLE, A.; ROPINSKI, T. Net2vis: Transforming deep convolutional networks into publication-ready visualizations. *arXiv preprint arXiv:1902.04394*, 2019. Citado na página 62.
- BEZ, M. *et al.* *Reconhecimento Facial e Micro Impressões*. 3. ed. São Paulo, SP: Murof, 2017. Citado 3 vezes nas páginas 22, 23 e 25.
- BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. Citado na página 58.
- CARRIER, P.-L. *et al.* Fer-2013 face database. *Universit de Montral*, 2013. Citado na página 28.
- CHOLLET, F. *Deep Learning with Python*. 1. ed. USA: Manning Publications Co., 2017. ISBN 1617294438. Citado 4 vezes nas páginas 36, 46, 56 e 57.
- CHOLLET, F. *et al.* *Keras*. 2015. <https://keras.io>. Citado na página 57.
- CODE, V. S. *VSCoDe API*. 2020. Disponível em: <<https://code.visualstudio.com/api/references/vscode-api>>. Acesso em: 12 out. 2020. Citado na página 68.
- COELHO, I. M. *et al.* A gpu deep learning metaheuristic based model for time series forecasting. *Applied Energy*, v. 201, p. 412 – 418, 2017. ISSN 0306-2619. Citado na página 51.

COELHO, V. N. *et al.* A hybrid deep learning forecasting model using gpu disaggregated function evaluations applied for household electricity demand forecasting. *Energy Procedia*, v. 103, p. 280 – 285, 2016. ISSN 1876-6102. Renewable Energy Integration with Mini/Microgrid – Proceedings of REM2016. Citado na página 51.

DENG, N.; TIAN, Y.; ZHANG, C. *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*. 1. ed. Minneapolis, Minnesota, EUA: Chapman and Hall/CRC, 2012. ISBN 9781439857922. Citado na página 47.

DENG, Z. *et al.* Factorized variational autoencoders for modeling audience reactions to movies. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA: IEEE, 2017. p. 6014–6023. Citado 2 vezes nas páginas 52 e 53.

DOMINGUEZ-MORALES, J. P. *Neuromorphic audio processing through real-time embedded spiking neural networks*. Tese (Doutorado), dez. 2018. Citado 2 vezes nas páginas 44 e 45.

DRESCH, A.; LACERDA, D.; ANTUNES, J. *Design Science Research: Método de Pesquisa para Avanço da Ciência e Tecnologia*. Porto Alegre, RS: Bookman Editora, 2015. ISBN 9788582602980. Citado 3 vezes nas páginas 15, 16 e 17.

DU, K.-L.; SWAMY, M. N. S. Recurrent neural networks. *Neural Networks and Statistical Learning*. London: Springer London, 2014. p. 337–353. ISBN 9781447155713. Citado 2 vezes nas páginas 40 e 41.

EKMAN, P.; FRIESEN, W. *Unmasking the Face: A Guide to Recognizing Emotions from Facial Clues*. Malor Books, 2003. (Spectrum book). ISBN 9781883536367. Disponível em: <<https://books.google.com.br/books?id=TukNoJDgMTUC>>. Citado 5 vezes nas páginas 13, 20, 21, 22 e 51.

EKMAN, P.; OSTER, H. Facial expressions of emotion. *Annual Review of Psychology*, v. 30, n. 1, p. 527–554, 1979. Citado 2 vezes nas páginas 23 e 28.

FARIAS, L. F. V. *Estudo, Desenvolvimento e Validação de um Dispositivo para Medição de Pressão de Contato*. 2019. Citado na página 19.

GEORGESCU, M.; IONESCU, R. T.; POPESCU, M. Local learning with deep and handcrafted features for facial expression recognition. *IEEE Access*, PP, 05 2019. Citado 3 vezes nas páginas 53, 54 e 67.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. Disponível em: <<http://www.deeplearningbook.org>>. Acesso em: 22 abr. 2020. Citado na página 41.

GOOGLE. *Colaboratory: Frequently Asked Questions*. 2020. Disponível em: <<https://research.google.com/colaboratory/intl/pt-BR/faq.html>>. Acesso em: 01 out. 2020. Citado na página 59.

GOOGLE. *Introduction to Google Drive API*. 2020. Disponível em: <<https://developers.google.com/drive/api/v3/about-sdk>>. Acesso em: 12 out. 2020. Citado na página 68.

HAYKIN, S. *Redes Neurais: Princípios e Prática*. 2. ed. Porto Alegre, RS: Bookman, 2001. ISBN 9788573077186. Citado 7 vezes nas páginas 31, 32, 33, 34, 36, 38 e 41.

- ISO/IEC JTC 1/SC 29/WG 11. *ISO/IEC 14496-2:2001: Information technology — Coding of audio-visual objects — Part 2: Visual*. Geneva, Switzerland: International Organization for Standardization, 2001. 517 p. Available in English only. Citado 2 vezes nas páginas 24 e 25.
- JAIMES, A.; SEBE, N. Multimodal human computer interaction: A survey. *Computer Vision and Image Understanding*, v. 108, p. 116–134, out. 2007. Citado na página 13.
- JIA, Y. *et al.* Caffe: Convolutional architecture for fast feature embedding. *MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia*, jun. 2014. Citado na página 50.
- KHAN, S. *et al.* *A Guide to Convolutional Neural Networks for Computer Vision*. Australia: Morgan & Claypool Publishers, 2018. (Synthesis Lectures on Computer Vision). ISBN 9781681730226. Citado 7 vezes nas páginas 31, 41, 42, 43, 44, 45 e 46.
- KIRK, D. B.; HWU, W.-m. W. *Programming Massively Parallel Processors, Third Edition: A Hands-on Approach*. 3. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. ISBN 9780128119860. Citado na página 51.
- KOWALCZYK, A. *Support Vector Machines Succinctly*. 1. ed. Morrisville, Carolina do Norte, EUA: Syncfusion Inc., 2017. Citado 3 vezes nas páginas 47, 48 e 49.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. PEREIRA, F. *et al.* (Ed.). *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012. p. 1097 – 1105. Disponível em: <<http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>>. Citado na página 35.
- LACERDA, D. *et al.* Design science research: A research method to production engineering. *Gestão & Produção*, v. 20, p. 741–761, 2013. Citado 2 vezes nas páginas 15 e 16.
- LLC, P. E. G. *Facial Action Coding System*. 2020. Disponível em: <<https://www.paulekman.com/facial-action-coding-system>>. Acesso em: 4 abr. 2020. Citado na página 23.
- LOON, R. V. *Machine learning explained: Understanding supervised, unsupervised, and reinforcement learning*. 2018. Disponível em: <<https://bigdata-madesimple.com/machine-learning-explained-understanding-supervised-unsupervised-and-reinforcement-learning/>>. Acesso em: 17 abr. 2020. Citado na página 37.
- LUCEY, P. *et al.* The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*. San Francisco, CA: IEEE, 2010. p. 94–101. Citado na página 27.
- LYONS, M.; KAMACHI, M.; GYOBA, J. *The Japanese Female Facial Expression (JAFFE) Database*. Zenodo, 1998. Disponível em: <<https://doi.org/10.5281/zenodo.3451524>>. Acesso em: 29 abr. 2020. Citado na página 26.

- MANNING, C. D.; RAGHAVAN, P.; SCHÜTZ, H. Support vector machines and machine learning on documents. *Introduction to Information Retrieval*. Cambridge, Reino Unido: Cambridge University Press, 2009. p. 319–348. Citado 2 vezes nas páginas 47 e 48.
- MARTÍNEZ, A.; BENAVENTE, R. The ar face database. *CVC Technical Report 24*, p. 8, jun. 1998. Citado 2 vezes nas páginas 27 e 28.
- MAYYA, V.; PAI, R.; M., M. P. M. Automatic facial expression recognition using dcnn. *Procedia Computer Science*, v. 93, p. 453–461, dez. 2016. Citado na página 50.
- MCKINNEY Wes. Data Structures for Statistical Computing in Python. WALT Stéfan van der; MILLMAN Jarrod (Ed.). *Proceedings of the 9th Python in Science Conference*. [S.l.: s.n.], 2010. p. 56 – 61. Citado na página 58.
- MICHELUCCI, U. *Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks*. 1. ed. USA: Apress, 2018. ISBN 9781484237892. Citado na página 45.
- MOHAMMED, M.; KHAN, M.; BASHIER, E. *Machine Learning: Algorithms and Applications*. Boca Raton, Flórida, EUA: CRC Press, 2016. ISBN 9781498705387. Citado 2 vezes nas páginas 36 e 37.
- NIELSEN, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015. Disponível em: <<http://neuralnetworksanddeeplearning.com/>>. Acesso em: 12 abr. 2020. Citado 8 vezes nas páginas 35, 37, 38, 39, 40, 41, 44 e 45.
- NUMPY. *Case Study: The First Image of a Black Hole*. 2020. Disponível em: <<https://numpy.org/case-studies/blackhole-image/>>. Acesso em: 29 aug. 2020. Citado na página 57.
- NVIDIA. *CUDA C++ Programming Guide Version 10.2*. 2019. Disponível em: <https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf>. Acesso em: 18 mar. 2020. Citado 2 vezes nas páginas 51 e 57.
- OLIPHANT, T. E. *A guide to NumPy*. [S.l.]: Trelgol Publishing USA, 2006. Citado na página 57.
- OLIVEIRA, R. D. *et al.* A system based on artificial neural networks for automatic classification of hydro-generator stator windings partial discharges. *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, v. 16, p. 628–645, set. 2017. Citado na página 32.
- OVERFLOW, S. *Developer Survey Results 2019 - Most Popular Development Environments*. 2019. Disponível em: <<https://insights.stackoverflow.com/survey/2019>>. Acesso em: 12 out. 2020. Citado na página 68.
- PEKEL, E.; KARA, S. A comprehensive review for artificial neural network application to public transportation. *Sigma Journal of Engineering and Natural Sciences*, v. 35, p. 157–179, mar. 2017. Citado 2 vezes nas páginas 40 e 41.
- RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. USA: Prentice Hall Press, 2009. ISBN 9780136042594. Citado 7 vezes nas páginas 30, 33, 34, 36, 37, 40 e 41.

- SAHA, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. 2018. Disponível em: <<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>>. Acesso em: 23 abr. 2020. Citado 2 vezes nas páginas 44 e 45.
- SARKAR, D.; BALI, R.; SHARMA, T. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*. 1. ed. USA: Apress, 2017. ISBN 1484232062. Citado na página 56.
- SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press, 2014. ISBN 9781107057135. Citado na página 31.
- SIMON, H. A. *The Sciences of the Artificial*. Cambridge, Massachusetts: MIT Press, 1996. (The MIT Press). ISBN 9780262264495. Citado na página 15.
- SQUIRE, L. *et al. Fundamental Neuroscience*. 3. ed. New York: Elsevier Science, 2008. ISBN 9780123740199. Citado 2 vezes nas páginas 30 e 31.
- SRIVASTAVA, N. *et al.* Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, JMLR.org, v. 15, n. 1, p. 1929–1958, jan. 2014. ISSN 1532-4435. Citado 2 vezes nas páginas 46 e 47.
- TIAN, Y.-L.; KANADE, T.; COHN, J. F. Facial expression analysis. *Handbook of Face Recognition*. New York, NY: Springer New York, 2005. p. 247–275. ISBN 9780387272573. Citado na página 26.
- WOLF, L. Face recognition, geometric vs. appearance-based. *Encyclopedia of Biometrics*. Boston, MA: Springer US, 2009. p. 347–352. ISBN 9780387730035. Citado na página 26.
- ZHOU, J. *et al.* A method of facial expression recognition based on gabor and nmf. *Pattern Recognition and Image Analysis*, v. 26, p. 119–124, jan. 2016. Citado na página 55.