UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

ALDOIR GRANDO

ANÁLISE FACIAL DE EMOÇÕES UTILIZANDO REDES NEURAIS NO CONTEXTO DE UMA SALA DE AULA INTELIGENTE

CAXIAS DO SUL

2020

ALDOIR GRANDO

ANÁLISE FACIAL DE EMOÇÕES UTILIZANDO REDES NEURAIS NO CONTEXTO DE UMA SALA DE AULA INTELIGENTE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Profa. Dra. Carine Geltrudes Webber

CAXIAS DO SUL

2020

ALDOIR GRANDO

ANÁLISE FACIAL DE EMOÇÕES UTILIZANDO REDES NEURAIS NO CONTEXTO DE UMA SALA DE AULA INTELIGENTE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 02/07/2020

BANCA EXAMINADORA

Profa. Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

Profa. Dra. Maria de Fatima Webber do Prado Lima
Universidade de Caxias do Sul - UCS

Profa. Ma. Iraci Cristina da Silveira De Carli
Universidade de Caxias do Sul - UCS

Aos meus pais, por sempre me apoiarem.

AGRADECIMENTOS

Gostaria de agradecer em especial aos meus pais e minha família, que sempre me apoiaram em todos os momentos para que eu pudesse seguir e chegar até aqui.

À minha orientadora profª. Dra. Carine Geltrudes Webber, por ter me dado as direções, por me incentivar e por sempre demostrar disposição para as dúvidas e questionamentos.

À meus amigos por estarem juntos no momentos felizes e tristes. E a todos que direta ou indiretamente contribuíram para que este trabalho fosse possível. Meu mais sincero obrigado!

"*A felicidade às vezes é uma bênção - mas geralmente é uma conquista.*"

Paulo Coelho

RESUMO

As salas de aula inteligentes (termo oriundo da língua inglesa, *smart class*) surgem como uma inovação na área da educação, visando melhorar o ensino e auxiliar tanto professores como estudantes em salas de aula. Ao equipar espaços educacionais com sensores e inteligência artificial é possível analisar um ambiente e promover *feedback* sobre a motivação e o interesse do estudante na aula. As informações produzidas podem então ser utilizadas pelo professor para entender sobre o andamento de suas aulas, e assim, melhor estruturá-las. Dentro deste contexto, este trabalho apresenta os conceitos de sala de aula inteligente, visão computacional e redes neurais, necessários para a construção de um *software* capaz de analisar as imagens de estudantes em sala de aula a fim de identificar padrões de emoções. Para poder realizar o desenvolvimento de um *software* foram estudados os processos de aquisição e processamento de imagens, definição e implementação de uma rede neural, culminando na realização de testes. Neste contexto, este trabalho apresenta a proposta de aplicação de uma rede neural convolucional para analisar imagens de expressões faciais de estudantes em um ambiente de ensino e reconhecer as emoções que estão sendo expressas. Testes realizados confirmaram a viabilidade do uso de redes neurais convolucionais para o reconhecimento de emoções nas expressões faciais, conforme estudos relacionados apontaram previamente. Entende-se que, desta forma, torna-se-á possível integrar o *software* proposto a um sistema IoT (*internet of things*) que constituirá componente relevante a uma sala de aula, adicionando assim um nível preliminar de inteligência a ela.

Palavras-chaves: Análise de Emoções, Inteligência Artificial, Visão Computacional, Redes Neurais, Sala de Aula Inteligente

ABSTRACT

Smart classrooms emerge as an innovation in the field of education, aiming to improve teaching and assisting both teachers and students in classrooms. By equipping educational spaces with sensors and artificial intelligence it is possible to analyze an environment and promote feedback on the student's motivation and interest in the class. The information produced can then be used by the teacher to understand about the progress of his or her classes, and thus, better structure them. Within this context, this work presents the concepts of intelligent classroom, computer vision and neural networks, necessary for the construction of software capable of analyzing the images of students in the classroom in order to identify patterns of emotions. In order to carry out the development of a software, the processes of image acquisition and processing, definition and implementation of a neural network were studied, culminating in the performance of tests. In this context, this work presents the proposal of applying a convolutional neural network to analyze images of students' facial expressions in a teaching environment and recognize the emotions that are being expressed. Tests carried out confirmed the feasibility of using convolutional neural networks to recognize emotions in facial expressions, as related studies previously pointed out. It is understood that, in this way, it will be possible to integrate the proposed software to an IoT (internet of things) system that will constitute a relevant component to a classroom, thus adding a preliminary level of intelligence to it.

Keywords: Emotion Analysis, Artificial Intelligence, Computer Vision, Neural Networks, Smart Class

LISTA DE ALGORITMOS

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A alta velocidade com que dispositivos inteligentes passaram a ser desenvolvidos, e testados em novos ambientes, tem elevado a relação homem-máquina a um maior nível de complexidade. Em cenários e ambientes de sala de aula já se começa a observar as mesmas tendências (HERRERA; NÚNEZ, 2018). Neste contexto, tecnologias como a internet das coisas (*IoT*, do inglês *Internet of Things*) pode tornar possível a inserção de novos artefatos de hardware e *software* nas salas de aula, apoiando diretamente o professor.

A ideia principal do *IoT* é equipar os objetos físicos do nosso cotidiano com tecnologia embarcada para que possam se conectar e comunicar através da internet (SINGH; CHITRANSH; TANWAR, 2016). O conceito de inteligência do ambiente (AmI, do inglês *Ambiente Intelligence*), utiliza tecnologias de *IoT* para criar um ambiente que auxilie o usuário em suas tarefas do dia a dia de maneira sensível e eficiente (AUGUSTO, 2007). Cada novo dispositivo, neste contexto, se interliga a outros de uma maneira que passa despercebida aos olhos do usuário (FISCHER *et al.*, 2019). Isso deve ocorrer sem requerer o uso de nenhum tipo de equipamento especial por parte dos usuários, sensores e dispositivos devem estar instalados e serem capazes de coletar e processar informações de forma transparente. Tais funcionalidades são possíveis graças aos avanços da computação pervasiva e ubíqua que buscam integrar dispositivos inteligentes em objetos do dia a dia, como móveis, roupas, brinquedos e outros (NAKASHIMA; AGHAJAN; AUGUSTO, 2010).

Os conceitos de *IoT* e *AmI*, combinados, possibilitam a construção de uma *Smart Class* (sala de aula inteligente), um espaço de estudo equipado com dispositivos e sensores capazes de proporcionar aos estudantes uma aprendizagem melhor e mais eficiente (FISCHER *et al.*, 2019). Porém, não basta integrar tecnologias preexistentes, é necessário desenvolver tecnologias a partir das existentes.

Ao interligar *hardware* e *software*, aliados com os conceitos de visão computacional e redes neurais, pode-se extrair comportamentos e padrões por meio da captura de imagens de estudantes em sala de aula. Alguns estudos têm demonstrado que, ao analisar-se a atenção e o comportamentos, através de traços faciais e corporais durante as aulas, é possível associá-los com métricas de produtividade, interação social e até de aprendizagem (ZALETELJ; KOŠIR, 2017).

A computação se ocupa em desenvolver métodos para lidar com problemas que possam ser formalizados e estruturados lógica e matematicamente. Por outro lado, a área da educação desenvolve teorias sobre como os seres humanos aprendem a fim de desenvolver recursos didáticos e pedagógicos que favoreçam as aprendizagens. Interligar as duas áreas é um desafio, pois significa que se deseja inferir aprendizagens, por exemplo, a partir de dados observados. Dentro deste contexto, nota-se a necessidade de recursos e artefatos que auxiliem e realizem tarefas de análise e identificação da motivação e do interesse de estudantes em sala de aula.

1.1 OBJETIVOS

Este trabalho tem por objetivo analisar imagens de estudantes em sala de aula, a fim de identificar e classificar as emoções de acordo com as suas expressões faciais. Busca-se, por meio deste processo, auxiliar o professor no acompanhamento das atividades e do interesse dos estudantes em sala de aula.

1.1.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

a) Identificar técnicas de visão computacional para uso em imagens capturadas de uma sala de aula, que tragam benefícios aos processos de ensino e aprendizagem;

b) Desenvolver um sistema de processamento de imagens que permita identificar as faces dos estudantes para posterior tratamento;

c) Realizar testes e experimentos com o sistema desenvolvido, a fim de avaliar seu desempenho e adequação da proposta.

1.2 ORGANIZAÇÃO DO TRABALHO

Para o melhor entendimento, este trabalho está em dividido em 4 capítulos. O capítulo dois apresenta a conceituação de visão computacional, redes neurais, e sala de aula inteligente. Nele também são apresentadas algumas das ferramentas de desenvolvimento existentes, além de abordar os trabalhos relacionados na área de estudo. No capítulo três são abordados o planejamento e as atividades desenvolvidas, e como acontecerão as fases de implementação e os testes do *software*. No capítulo quatro são apresentadas as considerações finais e discutidos os pontos de interesse do trabalho.

1.3 LIMITES DO TRABALHO

Para apresentar um trabalho que esteja de acordo com os objetivos estabelecidos, esse projeto busca conceituar e apresentar diferentes áreas, como visão computacional, redes neurais e sala de aula inteligente. Para tanto, é preciso elaborar uma implementação para que possa ser integrada a dispositivos *IoT* de maneira a serem utilizados para conferir um certo nível de inteligência a uma sala de aula, com possíveis implicações no processo de aprendizagem.

2 VISÃO COMPUTACIONAL PARA UMA SALA DE AULA INTELIGENTE

Smart class, que em português quer dizer sala de aula inteligente, é um conceito que surgiu recentemente para denominar a sala de aula que integra recursos de computação pervasiva, ubíqua e inteligência artificial (FISCHER *et al.*, 2019). Dentre os métodos e técnicas computacionais mais utilizados na sala de aula inteligente está a visão computacional. Ela permite o reconhecimento em tempo real de objetos e pessoas, tarefa que pode ser de grande utilidade no contexto educacional para inferências de aprendizagem, por exemplo. Neste capítulo são abordados os conceitos gerais da visão computacional, sendo apresentadas as redes neurais convolucionais. Em seguida, são apresentados trabalhos relacionados, que aplicam as redes neurais convolucionais no reconhecimento de padrões comportamentais. Posteriormente, o mesmo apresenta as principais ferramentas disponíveis para a aplicação de visão computacional e redes neurais convolucionais.

2.1 VISÃO COMPUTACIONAL

No dia a dia a visão humana é utilizada para observar os objetos ao nosso redor e obter informações importantes sobre eles, apesar de ser uma habilidade básica para os seres humanos o mesmo princípio torna-se complexo quando realizado por uma máquina (RAUTARAY; AGRAWAL, 2012). A visão computacional procura auxiliar na resolução de problemas complexos, simulando a visão humana, capturando, processando e analisando imagens.

A areá de visão computacional tenta imitar a habilidade do ser humano de tomar decisões de acordo com as informações obtidas. A visão computacional é responsável pela forma com que o computador consegue interpretar o ambiente que está ao seu redor, extraindo essas informações através de câmeras de vídeos e sensores, o que permite ao computador reconhecer e manipular os objetos que compõem as imagens capturadas (LI; SHI, 2018).

Conforme as tecnologias vão evoluindo, novas áreas acabam se beneficiando da integração de processamento de imagens com a inteligência artificial. A visão computacional tem suas aplicações presentes em diversos segmentos que envolvem a análise de imagens, reconhecimento de padrões e controles inteligentes, abrangendo áreas de conhecimento como agronomia, biologia, medicina, indústria, segurança e educação (NEVES; NETO; GONZAGA, 2012).

A visão computacional atua de maneira conjunta com o processamento de imagens. Para que possa funcionar adequadamente são necessárias as seguintes etapas que devem ser seguidas (Figura 1) (GONZALEZ; WOODS, 2000):

a) **Aquisição:** a aquisição é o primeiro passo no processamento digital de imagens. Ela

é geralmente composta por uma câmera digital fotográfica ou de vídeo que captura a imagem real e a transforma em uma imagem digital. Dependendo do dispositivo utilizado para a aquisição da imagem, esta pode variar entre uma imagem bidimensional ou uma tridimensional;

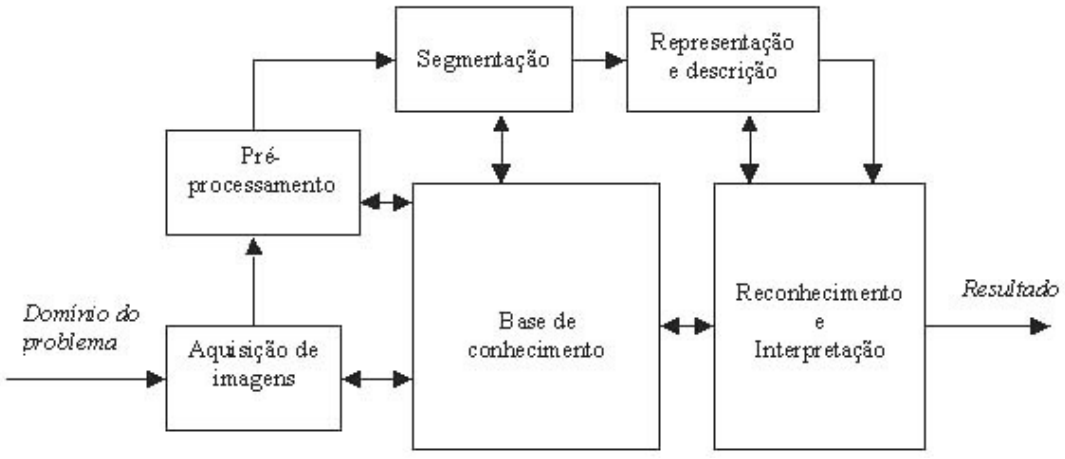
b) **Pré-Processamento:** o pré-processamento da imagem é o passo seguinte a aquisição. Antes de ser aplicado um método de visão computacional é necessário fazer um pré-processamento para melhorar a imagem, de maneira a atenuar ou suavizar algumas das características, como contraste ou ruídos existentes. Esta etapa é realizada conforme a necessidade específica de cada aplicação;

c) **Segmentação:** após realizar o pré-processamento das imagens é necessário fazer a segmentação das mesmas. Segmentar é dividir a imagem nos objetos que a compõem, selecionando assim as partes que interessam da imagem (ESQUEF, 2002). A etapa da segmentação é considerada uma das mais importantes do processamento, pois, é nela que são definidas quais serão as áreas e os objetos utilizados e analisados nas próximas etapas;

d) **Representação e Descrição:** na etapa de representação e descrição são extraídas as informações úteis da imagem, para poder realizar a classificação entre as possíveis classes de objetos. Existem duas formas para se representar os dados: por fronteira, como o número de objetos ou por região, como a forma de cada objeto (GONZALEZ; WOODS, 2000);

e) **Reconhecimento e Interpretação:** a última etapa de processamento é a etapa de reconhecimento e interpretação. Nesta etapa são atribuídos sentidos aos resultados, são analisadas todas as informações contidas na imagem, fazendo o reconhecimento e a classificação de objetos e seus padrões, atribuindo assim um significado ao conjunto de dados.

Figura 1 – Fluxograma de um sistema de visão computacional



Fonte: (GONZALEZ; WOODS, 2000)

2.2 REDES NEURAIS

Os seres humanos sempre procuraram meios de criar máquinas que os ajudassem a realizar as tarefas do dia a dia, seja para aperfeiçoar sua força ou aumentar o rendimento e a velocidade com que realizam as atividades, muitas dessas vezes inspiradas no próprio corpo humano.

As primeiras informações sobre redes neurais surgiram em 1943 em artigos escritos por McCulloch e Pitts. Ao descreverem como deveria ser o funcionamento de uma máquina baseada no cérebro humano, foi possível então modelar suas ideias criando uma rede neural simples, construída através de circuitos elétricos (FURTADO, 2019).

A partir deste modelo de rede neural surgiram dois novos paradigmas na área de inteligência artificial: simbólica e conexionista. A inteligência artificial simbólica busca simular o comportamento de inteligência humana sem uma inspiração biológica para realizá-la. A inteligência artificial conexionista busca simular a inteligência através de um sistema que possua uma estrutura parecida com a do cérebro humano, onde este sistema seria capaz de apresentar inteligência, por meio da aprendizagem e do erro (FURTADO, 2019).

O cérebro humano é uma máquina extremamente poderosa e capaz de realizar o processamento de inúmeras informações em um tempo mínimo. Pode-se dizer que seu principal componente é o neurônio, por meio do qual as informações são processadas e transmitidas, sendo capazes de responder às alterações do meio em que se encontra (HAYKIN, 2007). Pode-se definir um neurônio como sendo composto por três partes principais (ARBIB, 2002), conforme mostra a Figura 2:

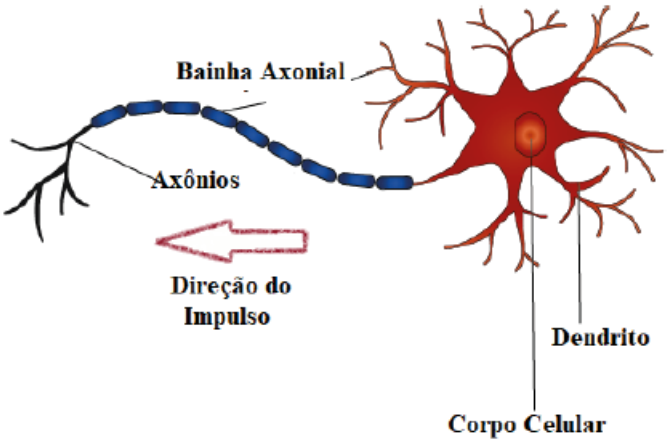
a) **Dendritos:** são os responsáveis por captar as informações do ambiente ou através da ligação com outros neurônios;

b) **Corpo Celular/Soma:** é onde fica localizado o núcleo da célula, sendo a parte responsável pelo processamento das informações recebidas;

c) **Axônio:** responsável por distribuir as informações processadas para os outros neurônios.

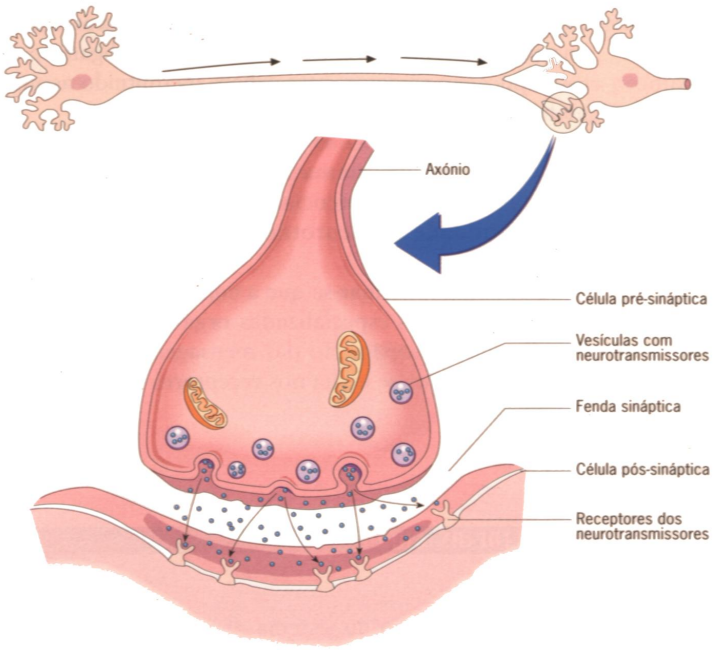
Para que possa ocorrer a transmissão das informações de um neurônio para outro é necessária a presença de estruturas especializadas denominadas como sinapse. As sinapses representadas na Figura 3 são locais de contatos entre os axônios e os dendritos que servem como comunicação entre as células (HAYKIN, 2007).

Figura 2 – Esquema de um neurônio biológico



Fonte: (FURTADO, 2019)

Figura 3 – Sinapse



Fonte: (TRISTÃO, 2019)

2.2.1 Neurônio Artificial

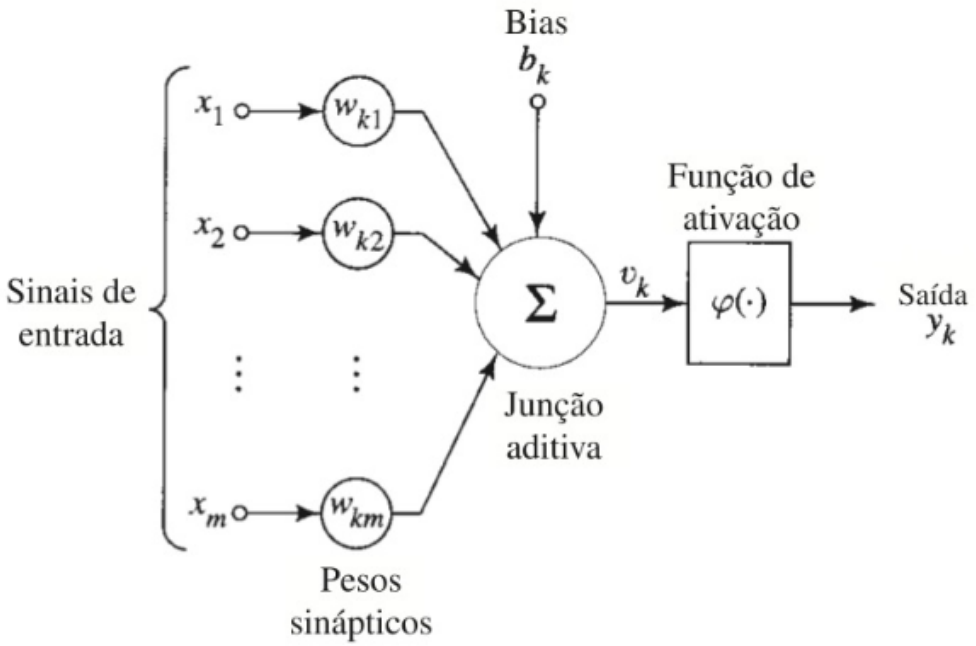
A inspiração para o desenvolvimento do neurônio artificial foi o modelo biológico. O neurônio artificial apresenta uma estrutura lógico-matemática que processa uma ou mais entradas e gera uma saída (FURTADO, 2019).

Um neurônio é o elemento básico, uma única unidade de processamento da informação

que é fundamental para a construção de uma rede neural (HAYKIN, 2007). Uma rede neural é composta por diversos neurônios interligados uns aos outros e que transmitem informações entre si. Um neurônio possui uma única saída, a qual pode estar conectada a diversas entradas de outros neurônios. O neurônio possui também uma unidade de processamento e pode apresentar diversas entradas distintas.

Um neurônio artificial pode receber múltiplas entradas x_j através das sinapses. O valor de entrada é dado pelo sinal recebido multiplicado pelo peso sináptico w_{kj}. Todas as entradas são somadas a partir da junção aditiva Σ e o valor obtido é então processado por uma função de ativação, também referida como função restritiva, já que restringe o intervalo de saída y_k a um valor finito. Além dos sinais vindos das entradas, cada neurônio é também excitado por uma polarização constante chamada bias b_k, que tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação. Na Figura 4 pode ser observado o esquema de um neurônio artificial.

Figura 4 – Esquema de um neurônio artificial



Fonte: (HAYKIN, 2007)

As redes neurais artificiais são normalmente utilizadas para a resolução de problemas complexos, quando o comportamento de suas variáveis não é rigorosamente conhecido. Uma das principais características de uma rede neural artificial é a sua capacidade de aprender por meio de exemplos, e produzir saídas adequadas para as entradas que não estavam presentes durante a etapa de treinamento (AMBRÓSIO, 2002).

Nas redes neurais artificiais pode-se associar os dendritos com as entradas das informações, responsáveis por perceberem os diferentes tipos de sinais e fazerem a ligação com o corpo celular artificial, através de canais de comunicação que estão associados a diferentes pesos, simulando as sinapses. Também é possível associar o corpo celular com o processamento, capaz de modificar sua própria programação de acordo com os sinais recebidos na entrada e gerados na saída. Já os axônios podem ser associados com a saída, que envia a informação de acordo com o processamento realizado, podendo estar conectados a diversos outros neurônios.

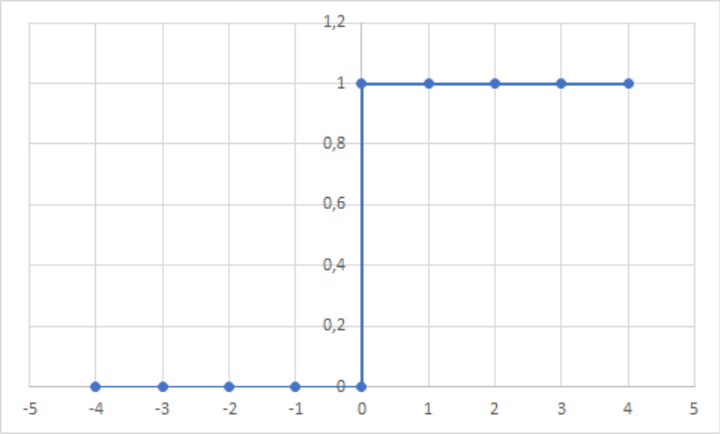
2.2.2 Funções de Ativação

Uma função de ativação tem a capacidade de limitar o sinal de saída de um neurônio, normalmente estando em um intervalo fechado [0,1] ou então [-1,1], podendo também o intervalo de saída estar entre [-∞,+∞] (FURTADO, 2019). Uma função de ativação decide se um neurônio deve ou não ser ativado, de acordo com as informações recebidas, realizando uma transformação não linear ao longo do sinal de entrada. A função de ativação representa o efeito que uma entrada e o estado atual exercem na definição do próximo estado, assim são definidas algumas das funções de ativação existentes:

a) **Função Limiar:** a função de ativação limiar normalmente restringe os valores de saída das redes neurais em [0,1], sendo 0 quando os resultados forem negativos e 1 quando os resultados forem positivos, podendo ser representada pela Figura 5:

$$f(u) = \begin{pmatrix} 1 & se & u \geq 0 \\ 0 & se & u < 0 \end{pmatrix} \tag{2.1}$$

Figura 5 – Função limiar

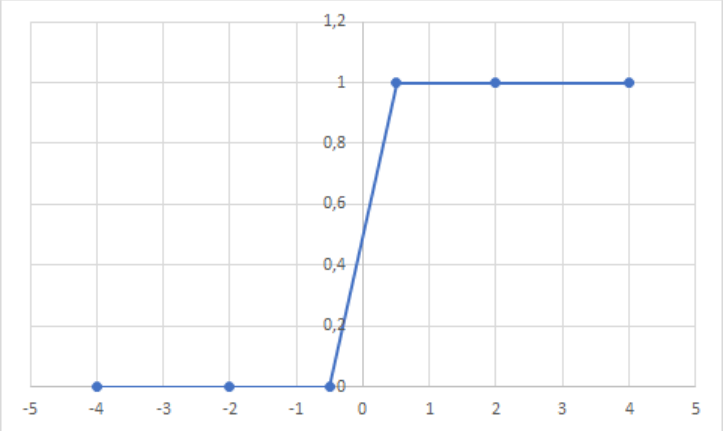


Fonte: (Autor, 2019)

b) **Função Linear por Partes:** esta forma de função de ativação pode ser vista como uma aproximação de um amplificador não linear. Na função linear por partes podem ser vistas duas formas especiais: caso a região linear de operação for mantida sem entrar em saturação, surge um combinador linear e caso o fator de amplificação da região linear for infinitamente grande a função linear por partes se reduz à função limiar (HAYKIN, 2007). A função linear por partes pode ser representada pela Figura 6:

$$f(u) = \begin{pmatrix} 1 & se & u \geq +1/2 \\ u & se + 1/2 > u > -1/2 \\ 0 & se & u \leq -1/2 \end{pmatrix} \tag{2.2}$$

Figura 6 – Função linear por partes

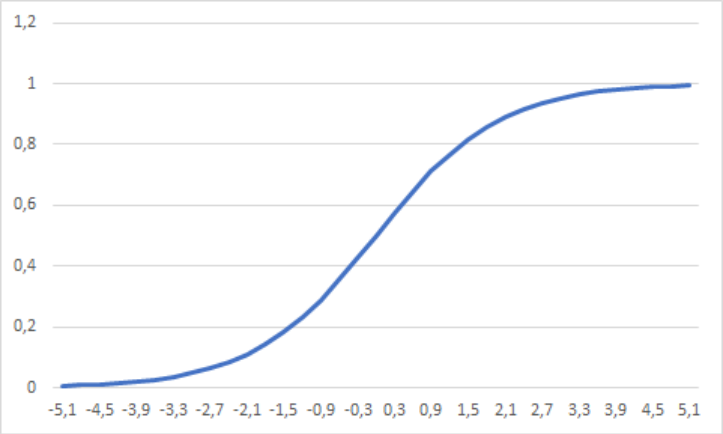


Fonte: (Autor, 2019)

c) **Função Sigmóide:** a função de ativação sigmóide é uma função crescente com balanceamento adequado entre o comportamento linear e o não linear, muito utilizada na construção de redes neurais. Na função sigmóide a inclinação se dá a partir do parâmetro α da função, e conforme o seu valor aumenta a curva da função tende a se tornar mais inclinada (HAYKIN, 2007). A função sigmóide pode ser representada pela Figura 7:

$$f(u) = \frac{1}{1 + exp(-au))} \tag{2.3}$$

Figura 7 – Função sigmóide

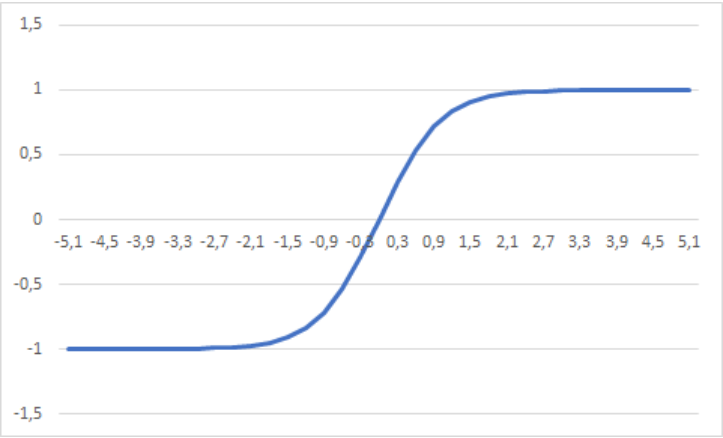


Fonte: (Autor, 2019)

d) **Função Tangente Hiperbólica:** pelo fato de a função logística apresentar valores de ativação apenas no intervalo (0, 1), ela acaba em muitos casos sendo substituída pela função tangente hiperbólica. A função tangente hiperbólica preserva a forma sigmóide da função logística, mas assume valores positivos e negativos. A função tangente hiperbólica pode ser representada pela Figura 8:

$$f(u) = tanh(u) \tag{2.4}$$

Figura 8 – Função tangente hiperbólica



Fonte: (Autor, 2019)

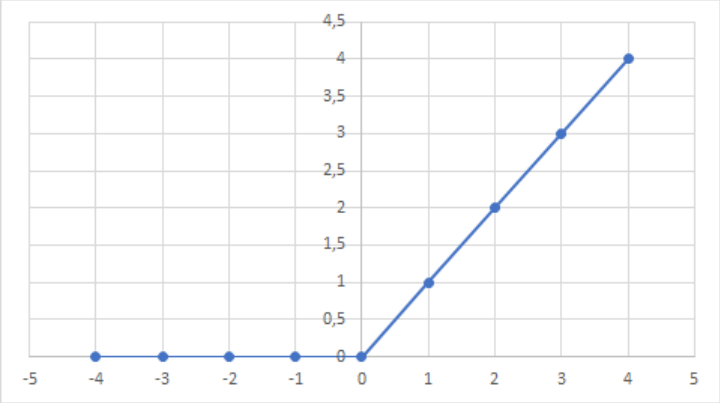
e) **Função *Softmax*:** a função de ativação *softmax* é utilizada para resolução de problemas de classificação e pode ser vista como uma generalização da função sigmóide. Esta função realiza o mapeamento de um vetor para uma probabilidade de pertencer a uma das *n* diferentes classes (WANG *et al.*, 2018), dada pela seguinte função:

$$f(x)_{\mathrm{i}} = \frac{e^{x_{\mathrm{i}}}}{\sum_{\mathrm{j}} e(x_{\mathrm{j}})} \tag{2.5}$$

f) **Função ReLU:** a função ReLU (do inglês *rectified linear unit*) é uma função de ativação não linear amplamente utilizada para extração de características. A função ReLU pode ser utilizada tanto para pré-treinamento como para classificação e possui uma alta velocidade de convergência (HARA; SAITO; SHOUNO, 2015), podendo ser representada pela Figura 9:

$$f(x) = max(0, x) \tag{2.6}$$

Figura 9 – Função *ReLU*

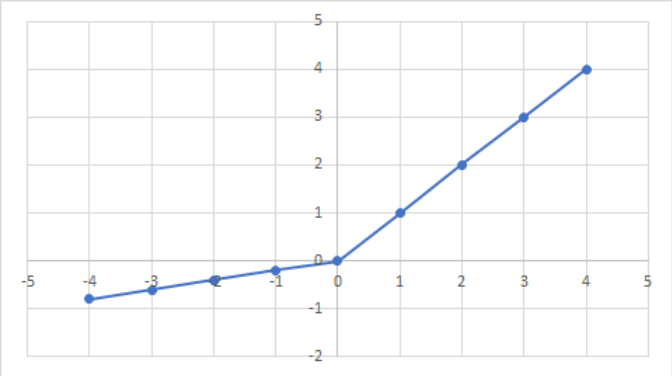


Fonte: (Autor, 2019)

g) **Função *Leaky* ReLU:** a função de ativação *Leaky* ReLU é uma versão melhorada da função ReLU. Nesta função é inserido um pequeno valor α para inclinação na parte negativa do seu domínio. A inserção do valor α faz com que a parte negativa da função fique ligeiramente inclinada, ao invés de uma linha horizontal zerada (MAAS; HANNUN; NG, 2013), podendo ser representada pela Figura 10:

$$f(x) = \begin{pmatrix} ax & se & x < 0 \\ x & se & x \geq 0 \end{pmatrix} \tag{2.7}$$

Figura 10 – Função *Leaky ReLU*



Fonte: (Autor, 2019)

2.2.3 Arquitetura das Redes Neurais

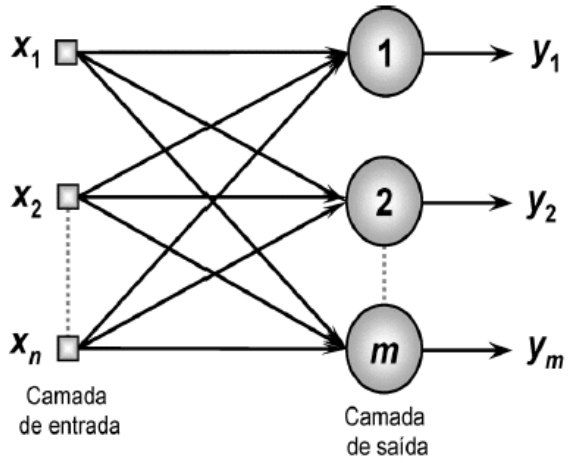
Uma das primeiras e mais simples formas de redes neurais artificiais foram os perceptrons, introduzido por Rosenblatt, em 1958. Os perceptrons de uma camada são úteis para a solução de problemas linearmente separáveis, podendo classificar padrões somente entre duas classes (AMBRÓSIO, 2002).

Quando um problema não admite uma separação linear exata, torna-se necessário utilizar os perceptrons multicamadas (MLP, do inglês *Multi Layer Perceptrons*), onde os neurônios estão agrupados em várias camadas que podem ser basicamente divididos em três partes: camada de entrada, onde são originados os dados; camadas intermediárias, também chamadas de camadas ocultas, onde são realizadas as extrações de características e inferência; camada de saída, responsável pela conclusão e apresentação do processamento.

Ainda segundo Furtado (2019) normalmente as arquiteturas fundamentais podem ser subdivididas em três diferentes classes:

a) **Rede *feedforward* de uma camada:** neste tipo de rede neural os neurônios são organizados em camadas, sendo a camada de entrada diretamente ligada a de saída, e sem possuir uma camada intermediaria. Nesta arquitetura de rede neural não existem ligações entre os neurônios de uma mesma camada, ou com camadas anteriores, os dados seguem sempre um único fluxo. Este tipo de arquitetura é normalmente empregada para a solução de problemas de filtragem e classificação de padrões. Na Figura 11 é mostrada a representação de uma rede *feedforward* de uma camada.

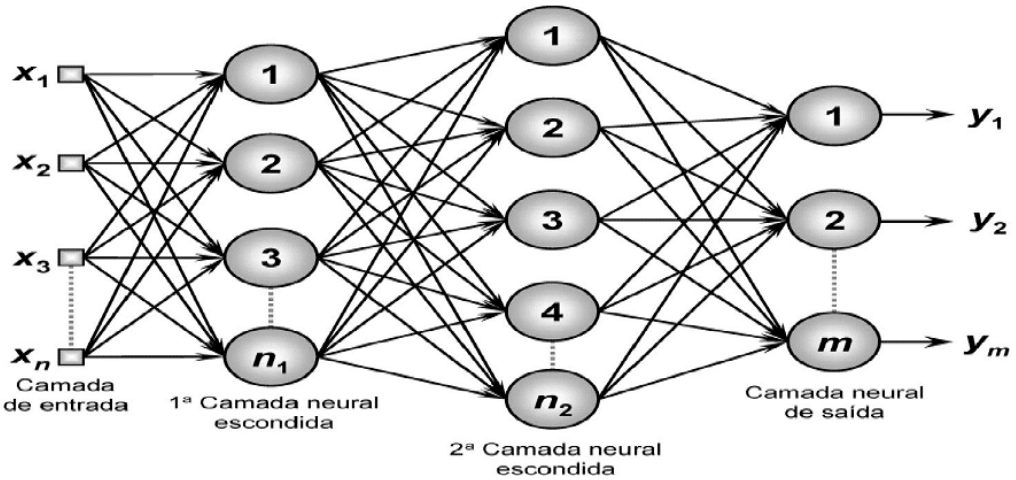
Figura 11 – Rede *feedforward* de uma camada



Fonte: (PALMIERE, 2016)

b) **Rede *feedforward* de múltiplas camadas:** este tipo de rede neural pode ser constituída de diversas camadas intermediárias, onde as conexões se dão sempre no sentido da camada de entrada para a camada de saída. As redes *feedforward* de múltiplas camadas podem ser classificadas como totalmente conectada, quando todos os nós de uma camada se comunicarem com todos os nós de outra camada, ou parcialmente conectada, quando apenas alguns dos nós de uma camada não estejam interligados a todos os outros nós da próxima camada. As redes *feedforward* de múltiplas camadas são normalmente empregadas na classificação de padrões, identificação de sistemas e controle de processos, sendo representada na Figura 12.

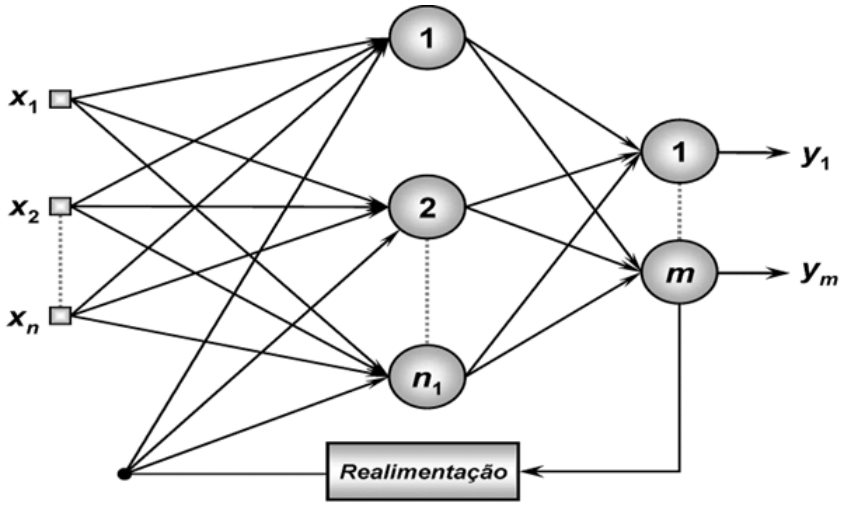
Figura 12 – Rede *feedforward* de múltiplas camadas



Fonte: (PALMIERE, 2016)

c) **Rede recorrente ou realimentada:** as redes neurais recorrentes podem ser constituídas de diversas camadas, porém se diferenciando das redes neurais *feedforward* por permitirem a realimentação de uma camada com as informações geradas pela camada posterior, ou ainda, podendo fazer a realimentação de um neurônio com a sua própria saída, sendo este processo chamado de (*self-feedback*). Este tipo de rede é normalmente utilizado em previsões de séries temporais e é representada na Figura 13.

Figura 13 – Rede recorrente



Fonte: (PALMIERE, 2016)

2.2.4 Treinamento e Aprendizagem

Uma das principais vantagens de se utilizar redes neurais é a sua capacidade de aprender através de seu próprio uso e melhorar o seu desempenho através dessa aprendizagem. Aprender significa adaptar-se ao ambiente, modificando o seu comportamento ao longo do tempo e de acordo com as regras, melhorando sua capacidade na resolução de problemas (AMBRÓSIO, 2002).

Para que uma rede neural artificial possa aprender e atingir seus objetivos, ela deve passar por uma etapa de treinamento, onde seus pesos passam a ser refinados no processo, de maneira que uma entrada futura seja corretamente classificada. Somente são permitidas alterações dos parâmetros de uma rede durante a etapa de treinamento, devendo permanecer estáticos durantes as fases de teste e de execução (AMBRÓSIO, 2002).

Em uma rede neural o conceito de aprendizado distingue-se de treinamento, pois o aprendizado está relacionado a uma tarefa que a rede está executando em função do treinamento, da topologia e da arquitetura da rede, já o treinamento é o processo de ensinar a rede neural (FURTADO, 2019).

Para realizar o treinamento são apresentadas amostras de dados na entrada da rede e a

saída é redirecionada para uma resposta esperada, forçando assim que seus pesos e parâmetros sejam modificados até atingirem uma validação desejada. Ao finalizar a etapa de treinamento, a rede terá então o conhecimento necessário sobre o ambiente em que está atuando (FLECK *et al.*, 2016).

A arquitetura da rede é quem define que tipo de treinamento deve ser aplicado para que ela possa ser capaz de resolver um problema. Segundo Haykin (2007), os algoritmos de aprendizagem podem ser divididos em três diferentes classes, sendo eles:

a) **Treinamento Supervisionado:** o treinamento supervisionado, utiliza-se de um agente externo para indicar a saída desejada a partir de um conjunto de amostras, aplicadas na entrada da rede. Através da diferença entre o resultado obtido e o resultado esperado (erro), a rede neural ajusta seus parâmetros até que esse erro seja mínimo, ou considerado aceitável, e somente a partir deste momento pode-se dizer que a rede está treinada. Entre os algoritmos de treinamento supervisionado podem ser citados o de Erro Médio Quadrático e o de *backpropagation*;

b) **Treinamento Não-Supervisionado:** o treinamento não-supervisionado não apresenta um agente externo indicando a saída esperada. A própria rede deve ser capaz de se organizar e identificar subconjuntos similares, criando assim novas classes de saída caso ainda não tenha identificado aquele tipo específico de dado. Durante o treinamento não-supervisionado os pesos e parâmetros da rede neural são ajustados pelo algoritmo de acordo com um conjunto de regras pré-estabelecidas, até que chegue à configuração desejada. Entre os algoritmos de treinamento não-supervisionado pode ser citado o de Aprendizado Competitivo;

c) **Treinamento por Reforço:** o treinamento por reforço é semelhante ao treinamento supervisionado, porém não possuindo as respostas esperadas na saída da rede. O treinamento por reforço tem a capacidade de saber se as saídas produzidas estão corretas ou não, sendo um método baseado na tentativa e erro, fazendo com que os ajustes dos pesos dependam única e exclusivamente das respostas obtidas durante o treinamento, de maneira a reforçar as respostas satisfatórias.

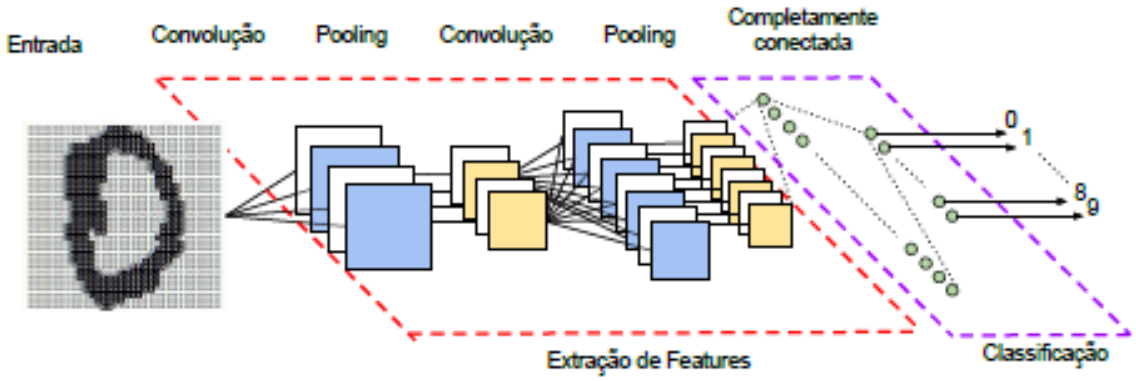
2.3 REDES NEURAIS CONVOLUCIONAIS

Uma rede neural convolucional (CNN, do inglês *Convolutional Neural Network* ou *ConvNets*) é um subtipo de rede neural *feedforward* de múltiplas camadas que se utiliza de treinamento supervisionado. As CNNs tiveram sua inspiração baseada no córtex visual e são amplamente utilizadas na área de visão computacional. As CNNs são normalmente aplicadas para processamento de dados formados por múltiplos vetores, como na classificação, detecção e reconhecimento de imagens e vídeos (LECUN; BENGIO; HINTON, 2015).

Semelhante aos processos tradicionais de visão computacional, uma CNN é capaz de aplicar filtros aos dados, mantendo a relação de vizinhança entre os *pixels* ao longo do processamento (VARGAS; PAES; VASCONCELOS, 2016). Uma CNN possui este nome, porque utiliza de uma operação matemática denominada convolução, um tipo de operação linear que a partir de duas funções, resulta em uma terceira, também chamada de mapa de características. Uma CNN utiliza-se de convolução no lugar de multiplicação em pelo menos uma de suas camadas.

A arquitetura típica de uma CNN é composta por duas grandes etapas, a extração das características pelas camadas convolucionais e a classificação. Dentro dessas duas etapas existem uma série de estágios a fim de obter uma saída que permita a classificação da imagem. Uma CNN é normalmente composta por diferentes tipos de camadas e entre elas estão as camadas de convolução, camadas de agrupamento (*pooling*) e camadas totalmente conectadas. Os primeiros estágios compostos por camadas de convolação e *pooling* são os considerados mais importantes para o processamento (LECUN; BENGIO; HINTON, 2015). O diagrama básico de uma CNN é mostrado na Figura 14.

Figura 14 – Rede neural convolucional



Fonte: (VARGAS; PAES; VASCONCELOS, 2016)

2.3.1 Camada de Convolução

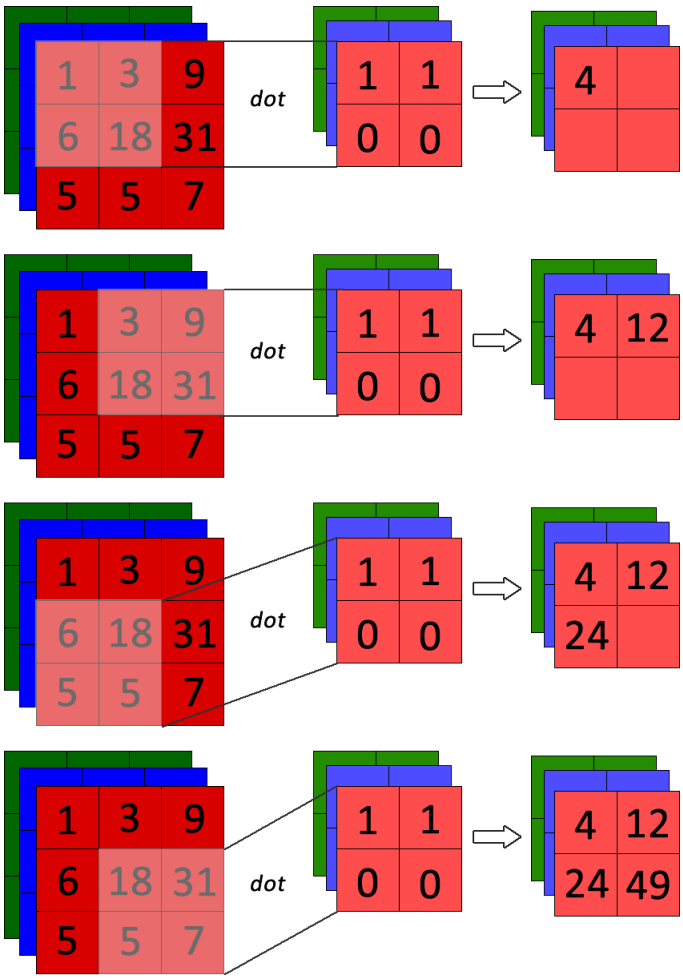
A camada de convolução é composta por diversos neurônios, onde cada um é responsável por aplicar um filtro em um pedaço específico da imagem. Os pesos atribuídos as conexões de um neurônio podem ser representados como uma matriz que identifica os filtros de convolução (também conhecidos como *kernel* ou máscara), possuindo valores reais e sendo capazes de aprender com o processo de treinamento (VARGAS; PAES; VASCONCELOS, 2016).

Cada filtro possui diferentes pesos, desta forma filtros distintos extraem diferentes características da imagem. Ao deslizar esses filtros sobre a imagem de entrada é originado uma imagem de saída, que então é repassada para a próxima camada. O deslocamento dos filtros em

relação à imagem original é controlado pelo parâmetro *stride*, podendo o deslocamento variar, desde que se encontre dentro dos limites de tamanho da imagem.

As saídas dos neurônios que tenham processado uma parte da imagem com um filtro em comum formam um mapa de características, e para que esses filtros possam ser aplicados em diferentes locais da imagem seus pesos são compartilhados durante o processo de treinamento, tal compartilhamento diminui significativamente o número de parâmetros que devem ser aprendidos e o tempo de treinamento da CNN (VARGAS; PAES; VASCONCELOS, 2016). A Figura 15 representa a aplicação de uma camada de convolução, utilizando um filtro de 2x2 em uma matriz de 3x3.

Figura 15 – Camada de convolução



Fonte: (PACHECO, 2019)

É comum ao final de uma camada de convolução ser aplicado uma função de ativação, modificando os dados recebidos. Normalmente são utilizadas funções que aplicam um certo grau de não-linearidade, tornando as categorias de saída linearmente separáveis, como a função de ativação ReLU.

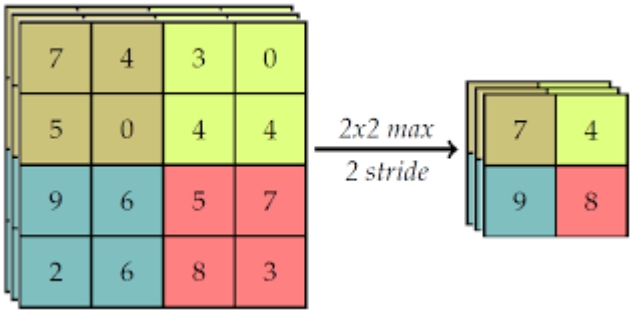
2.3.2 Camada de *Pooling*

Normalmente utilizada após a camada de convolução, a camada de *pooling* serve para simplificar os dados da camada anterior, reduzindo a dimensionalidade das informações. A redução das informações é necessária para que o treinamento possa ser realizado com maior agilidade, reduzindo os sobre ajustes e a invariância de transições (LEMLEY; BAZRAFKAN; CORCORAN, 2017)).

A camada de *pooling* funciona agrupando um conjunto de dados, utilizando-se dos conceitos de filtros e *strides* (quantidade de passos que podem ser dados de uma única vez), como, por exemplo em uma matriz de 4x4 sendo aplicado um filtro com configuração de 2x2 que irá selecionar um valor para representar cada área dessa matriz através de uma função para produzir o resultado final.

A escolha do valor de representação pode ser realizada por diversas funções, a mais utilizada é a de *max-pooling*, que seleciona o maior valor da região em análise. Outras funções como *min-pooling* que seleciona o valor do menor pixel e *avarege pooling* que calcula a média do valor dos *pixels* também podem ser utilizadas. A camada de *pooling* apenas reduz a altura e a largura de um mapa, não alterando sua profundidade (VARGAS; PAES; VASCONCELOS, 2016), sendo representada na Figura 16.

Figura 16 – Camada de *pooling*



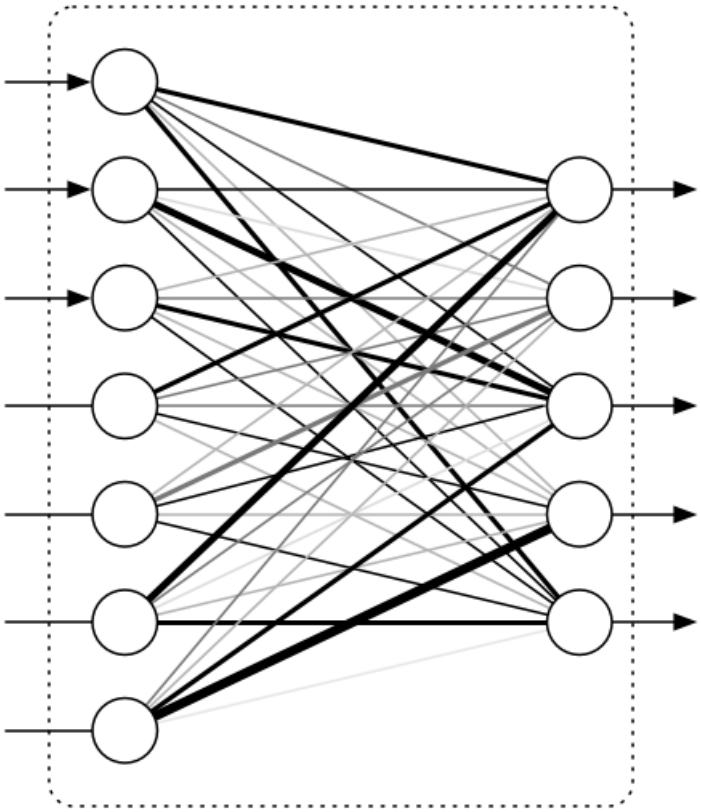
Fonte: (PACHECO, 2019)

2.3.3 Camada Totalmente Conectada

Após passar pelo estágio de extração das características a CNN deve realizar a tarefa de classificar as informações recebidas das camadas anteriores em uma das suas classes de domínio, este estágio recebe o nome de classificação.

As camadas totalmente conectadas são normalmente utilizadas como camadas finais, e seu nome está ligado a forma com que os neurônios desta camada se relacionam com a camada que a antecede, fazendo a ligação entre todos os neurônios conforme mostrado na Figura 17.

Figura 17 – Camada totalmente conectada



Fonte: (RAMSUNDAR; ZADEH, 2018)

2.3.4 Redes Neurais Convolucionais Existentes

A pesquisa e o desenvolvimento de CNNs é uma área em crescente evolução, e conta com uma grande diversidade de modelos existentes. Entre os modelos já desenvolvidos está a LeNet-5, sendo esta uma arquitetura considerada como uma das pioneiras no desenvolvimento de CNNs (LECUN *et al.*, 1998). A LeNet-5 possui 7 camadas e foi desenvolvida inicialmente para reconhecimento de caligrafia e códigos postais, tendo sido muito utilizada por bancos para reconhecer números escritos a mão em cheques.

Outra CNN existente e a AlexNet, que possui uma arquitetura semelhante a LeNet-5, porém mais profunda. Este modelo de CNN possui oito camadas, sendo cinco camadas de convolução e três totalmente conectadas (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). A AlexNet foi a primeira a utilizar a função ReLU como função de ativação.

A ZF Net foi a CNN vencedora do concurso ILSVRC2012 (do inglês *ImageNet Large Scale Visual Recognition Challenge 2012*). A ZF Net possui uma arquitetura semelhante a da AlexNet, mas com algumas modificações que as tornam mais eficientes, assim como a utilização de uma nova técnica de visualização denominada Rede Deconvolucional (ZEILER; FERGUS, 2013).

A CNN VGG Net possui uma arquitetura profunda, sendo composta por 19 camadas que utilizam apenas filtros de tamanho 3x3 com *stride* 1 junto com camadas *maxpooling* de 2x2 com *stride* 2 (SIMONYAN; ZISSERMAN, 2014a). Este modelo de CNN é capaz de conseguir uma taxa de erro de 7,3%.

A GoogLeNet também conhecida com Inception V1 é uma CNN que possui vinte e duas camadas e introduziu um novo modo chamado módulo inicial (SZEGEDY *et al.*, 2015). O modulo inicial é baseado em várias convoluções pequenas, utilizadas para reduzir o número de parâmetros e permitir que as operações de uma camada sejam executadas em paralelo.

Já a Rede Neural Residual (ResNet) foi a vencedora do concurso ILSVRC2015 com uma taxa de erro de 3,6%. A ResNet possui 152 camadas e introduziu a ideia de bloco residual, cuja utilização permite a rede pular conexões, tornando o seu treinamento mais rápido e eficiente (HE *et al.*, 2016).

2.4 TRABALHOS RELACIONADOS

Com o avanço das tecnologias e a criação de dispositivos inteligentes, a sala de aula inteligente (referida como *smart class*) surge como uma inovação. Segundo Fischer et al. (2019), uma sala de aula inteligente pode ser definida como um ambiente que apresenta as principais características buscadas pela Inteligência Artificial (IA), que são: a consciência (*self-awareness*) e a autonomia (*self-control*), necessárias para gerenciar um espaço de ensino e aprendizagem.

As salas de aula inteligentes visam proporcionar aos estudantes e professores um melhor, e mais seguro, ambiente de estudo. Desta maneira, almeja-se que o conteúdo educacional passe a ser apresentado de forma preferentemente interativa e dinâmica, tornando o processo de aprendizagem mais eficiente (FISCHER *et al.*, 2019).

A melhoria da qualidade na educação, o maior controle do professor sobre o andamento do aprendizado e a detecção das dificuldades enfrentadas pelos estudantes estão dentre os principais objetivos de se empregar o uso de sistemas inteligentes em uma sala de aula. Além disso, busca-se promover um maior conforto aos envolvidos e acompanhar o correto uso dos materiais disponíveis.

Em uma sala de aula inteligente podem ser empregados quadros interativos, onde os professores podem manipular o que está sendo projetado e fazer gravações das aulas. Podem ainda fazer uso de aplicativos que permitem a colaboração entre os estudantes e de sistemas que monitoram o ambiente, controlando a iluminação e temperatura e, desta forma, evitando desperdícios e contribuindo com a sustentabilidade.

Uma das formas de acompanhar, compreender e estudar o processo de ensino e aprendizagem pode ser por meio de imagens e vídeos capturados durante atividades na sala de aula. As imagens e vídeos obtidos, devem ser analisadas por meio de técnicas de visão computacional,

dentre elas as redes neurais.

Alguns estudos tentam identificar alguns dos estados emocionais dos estudantes, como interessado, cansado, confuso, animado e outros, através da análise de expressões faciais e corporais (ZALETELJ; KOŠIR, 2017) e (DINESH; S; BIJLANI, 2016). Tais identificações se dão por meio da observação de diferentes características, como ponto do olhar, movimentos da cabeça e das mãos, assim como movimentos do corpo e postura. Ao fazer tal análise é possível fornecer um *feedback* ao professor sobre cada um dos estudantes, permitindo assim, que ele possa adequar suas aulas de acordo com as dificuldades encontradas pelos estudantes em classe (WHITEHIL *et al.*, 2014).

Durante as pesquisas foram encontrados alguns trabalhos relacionados a área de computação visual e CNN, aplicados na implementação de salas de aulas inteligentes. Entre eles estão o trabalho desenvolvido por SR e LJ (2016) na Índia, que propõem a implementação de um sistema de reconhecimento facial em sala de aula para ser utilizado como uma ferramenta de auxílio aos professores e aos pais.

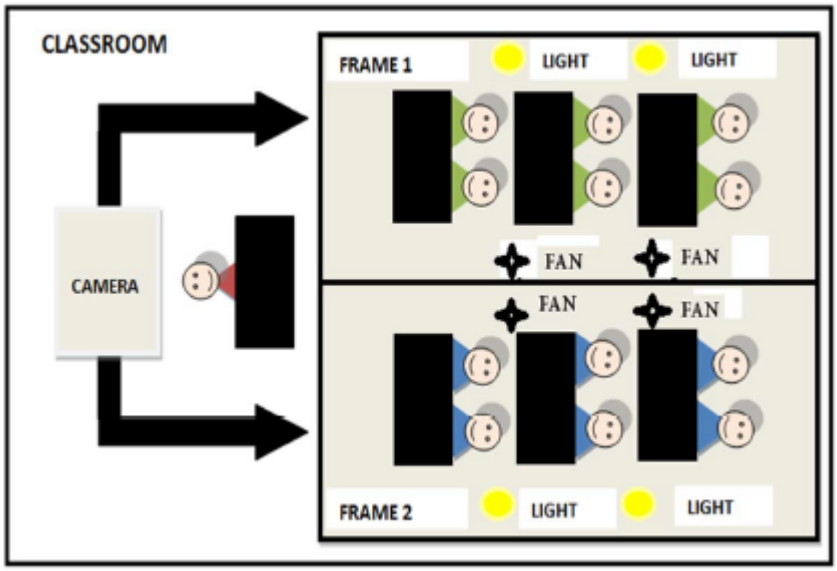
Para o desenvolvimento do trabalho são utilizados conceitos de visão computacional junto com dispositivos *IoT*, sendo implementado por intermédio da ferramenta Emgu CV, que é uma biblioteca .NET que permite a utilização de funções do *OpenCV*. Para realizar a detecção de faces são utilizados os classificadores *Haar Cascade*, já para a tarefa de reconhecimento das faces são utilizados os algoritmos PCA. Desta forma o projeto é capaz de monitorar o comportamento dos estudantes e informar aos pais sobre a sua presença ou ausência durante as aulas, sem a necessidade de que eles visitem fisicamente a escola. Na Figura 18 é apresentado o resultado obtido na realização do projeto.

Figura 18 – Implementação de um sistema de reconhecimento facial

Fonte: (SR; LJ, 2016)

O trabalho desenvolvido por Ani et al. (2018) na Índia, propõem a implementação de um sistema para detectar a presença de estudantes em sala de aula. Para realizar o desenvolvimento do trabalho foi utilizada uma câmera conectada a um dispositivo Arduíno que faz o controle do ambiente, podendo ligar ou desligar os equipamentos elétricos. Para reconhecer a presença dos estudantes foi utilizada a biblioteca *OpenCV*. Desta forma, através da análise das imagens dos estudantes e de sua posição na sala de aula é possível controlar a iluminação e a temperatura, aumentando o conforto e contribuindo com a sustentabilidade. A Figura 19 apresenta o esquema de controle de iluminação e temperatura projetados para o desenvolvimento do trabalho.

Figura 19 – Implantação de um sistema para detectar a presença de estudantes



Fonte: (ANI *et al.*, 2018)

O trabalho desenvolvido por Zaletelj e Košir (2017) na Eslovênia, propõem um sistema para predizer o nível de atenção dos estudantes em sala de aula, de acordo com parâmetros pré-estabelecidos. Para inferir um nível de atenção foram analisados os comportamentos dos estudantes, tais como: escrever, bocejar, apoiar a cabeça, se apoiar para trás na cadeira e o ponto do olhar. Os estudantes tiveram seus níveis de atenção divididos em três categorias:

a) **Nível Alto de Atenção:** o nível alto de atenção foi associado aos estudantes que olhavam para os slides, faziam anotações em 52% do tempo de análise e mantinham o corpo inclinado para frente durante 88% do tempo.

b) **Nível Médio de Atenção:** o nível médio de atenção foi associado aos estudantes que olhavam para os slides, mantinham o corpo inclinado para frente durante 84% do tempo e apoiavam a cabeça com uma mão durante 66% do tempo.

c) **Nível Baixo de Atenção:** o nível baixo de atenção foi associado a estudantes que demonstravam gestos de cansaço ou tédio, como inclinar-se para trás durante 41% do tempo, esfregar o pescoço, coçar a cabeça, bocejar 26% e olhar para longe.

A Tabela 1 mostra a relação entre os comportamentos observados nos estudantes e o nível de atenção estimado.

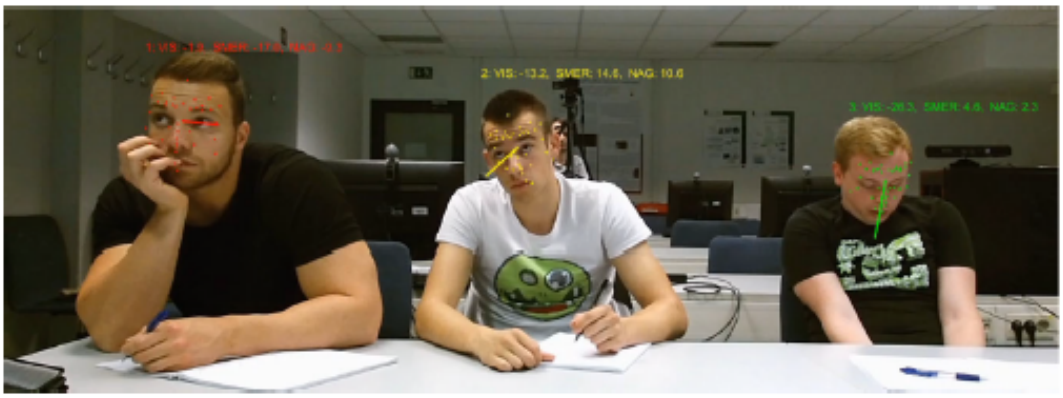
Tabela 1 – Relação entre comportamento observado e o nível de atenção

	Nível de Atenção (I)		
Comportamento	Baixo (1)	Médio (3)	Alto (5)
Escrever, $A_w(I)$	0.0	0.05	0.52
Bocejar, $A_y(I)$	0.26	0.06	0.001
Apoiar a cabeça, $A_s(I)$	0.21	0.61	0.11
Inclinar-se para trás, $A_b(I)$	0.41	0.16	0.12

Fonte: (ZALETELJ; KOŠIR, 2017)

Para o desenvolvimento do trabalho foi utilizado um sensor *Kinect One* com uma câmera para captar imagens 2D e 3D da postura corporal e das expressões faciais dos estudantes. Os dados foram capturadas em ambiente controlado, sendo utilizados pequenos grupos de estudantes para os testes. Durante a aquisição dos dados não haviam objetos na frente dos estudantes que pudessem atrapalhar posteriormente na análise conforme mostra Figura 20.

Figura 20 – Imagem capturada através do sensor *kinect one*

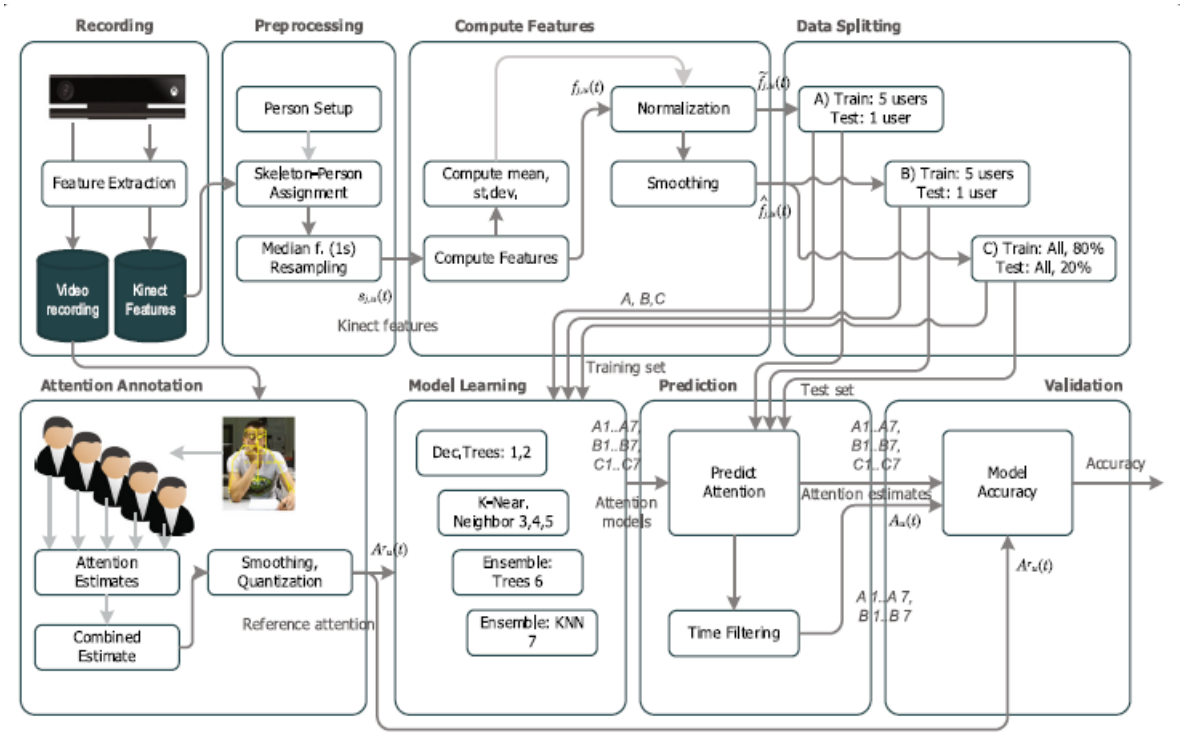


Fonte: (ZALETELJ; KOŠIR, 2017)

Após a aquisição dos dados, eles foram analisados através do *software* Matlab. O processo de análise foi dividido em partes conforme mostra a Figura 21. Também foram utilizados algoritmos de *machine learning* como *Decision tree, K-nearest neighbors e Bag of decision trees*

para realizar o treinamento dos classificadores e a avaliação dos níveis de atenção. Para servir de referência aos algoritmos de classificação foi utilizada a análise de observadores que estimaram os níveis de atenção de cada estudante, observando manualmente as imagens capturadas e atribuindo valores. Desta forma é possível estimar o nível de atenção dos estudantes durante as aulas através da análise de expressões faciais e corporais.

Figura 21 – Diagrama de processamento de dados



Fonte: (ZALETELJ; KOŠIR, 2017)

Já o trabalho realizado por Gupta, Ashwin e Guddeti (2019) na Índia, fez uma análise do humor dos estudantes no ambiente de sala de aula, por meio da coleta e análise de imagens e vídeos. O humor dos estudantes é dividido em 4 estados conforme mostrado na Tabela 2.

A coleta das imagens foi realizada através de câmeras, sendo gravada uma hora de video em sala de aula. As gravações foram feitas com pequenos grupos de estudantes, onde estes eram posicionados de maneira com que tivessem as expressões faciais captadas de forma clara pela câmera e sem nenhum material que pudesse atrapalhar as análises posteriores, conforme é mostrado na Figura 22.

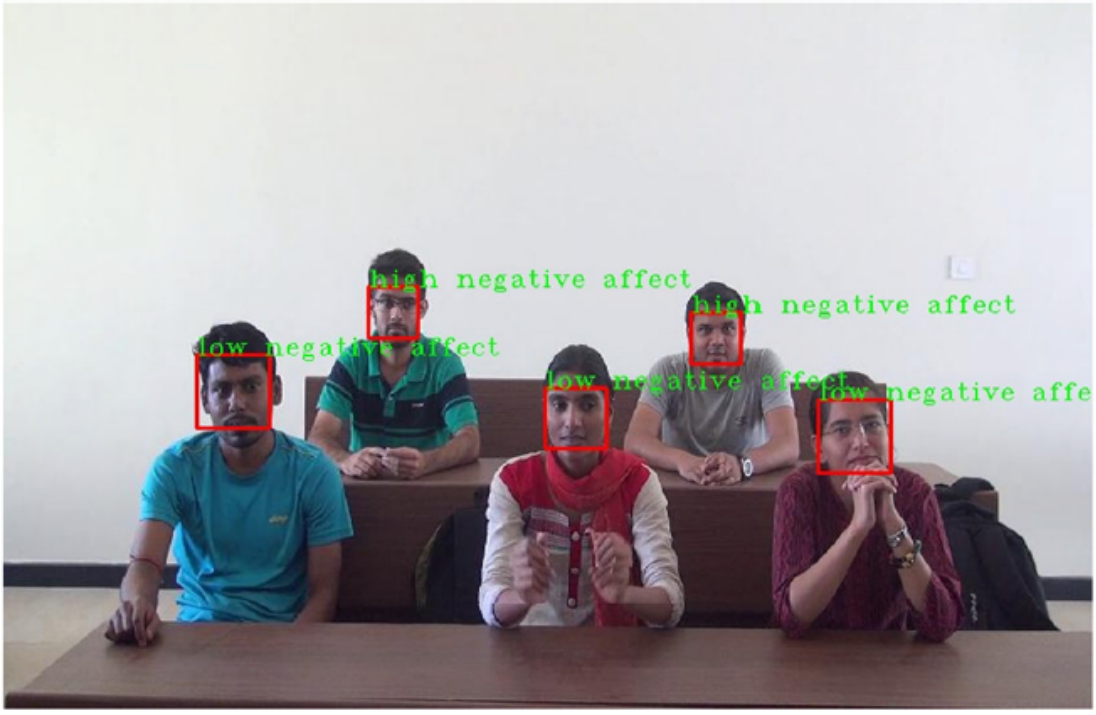
Após realizar a gravação do vídeo, este é subdividido em vídeos menores de 5 a 6 minutos cada. As faces são então extraídas de cada *frame* através da técnica MMFD (do inglês *Max-Margin Face Detection*). O reconhecimento do humor dos estudantes é realizado utilizado a CNN M-Inception-V3. Após analisar todo o vídeo é feito o cálculo da média do humor dos estudantes e um *feedback* é fornecido para o professor sobre o resultado obtido.

Tabela 2 – Classificação do humor baseada nas emoções

Emoção	Humor Correspondente
Alerta Excitado Exaltado Feliz	Afeto Alto Positivo
Contente Relaxado Calmo Neutro	Afeto Baixo Negativo
Fatigado Entediado Deprimido Triste	Afeto Baixo Positivo
Chateado Estressado Nervoso Tenso	Afeto Alto Negativo

Fonte: (GUPTA; ASHWIN; GUDDETI, 2019)

Figura 22 – Captura de imagens dos estudantes

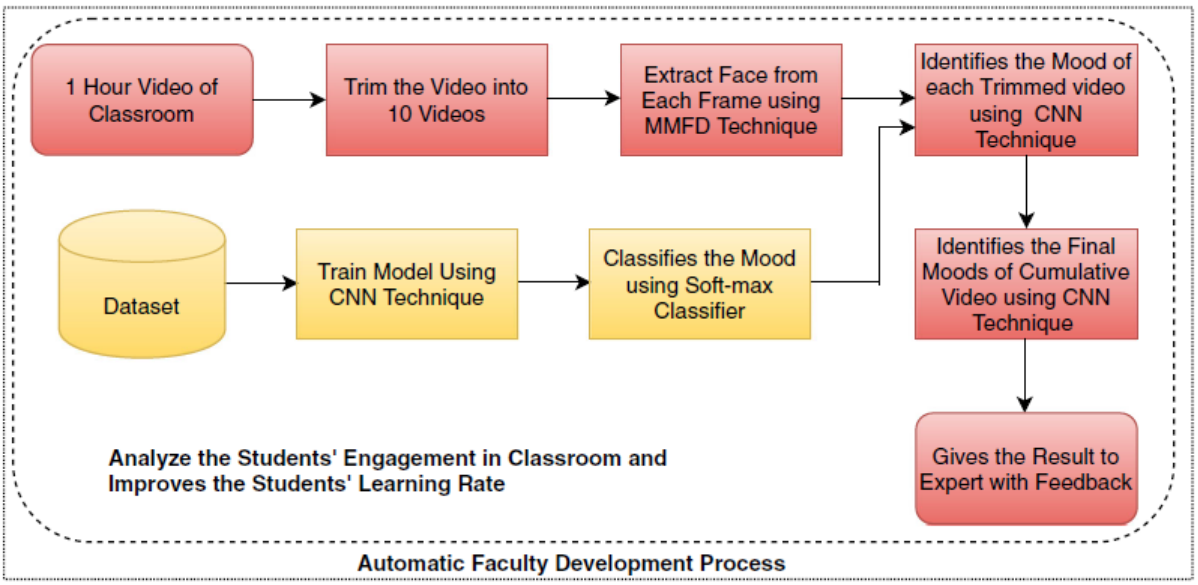


Fonte: (GUPTA; ASHWIN; GUDDETI, 2019)

A Figura 23 mostra o diagrama de processamento dos dados, com a captura do vídeo, o treinamento da CNN, a identificação do humor e o resultado obtido.

Através de uma forma não intrusiva como a captura de imagens faciais, e utilizando técnicas de visão computacional e *deep learning* é possível analisar o estado emocional dos estudantes, sendo que este exerce um papel fundamental no processo de aprendizagem de toda a turma.

Figura 23 – Diagrama de processamento de dados



Fonte: (GUPTA; ASHWIN; GUDDETI, 2019)

2.5 FERRAMENTAS PARA IMPLEMENTAÇÃO DE REDES NEURAIS

Para realizar implementações baseadas em técnicas de visão computacional e redes neurais artificiais é necessário utilizar ferramentas que facilitam e auxiliam no desenvolvimento de tais tarefas. Dentre as ferramentas disponíveis para desenvolvimento estão o *TensorFlow* e o *PyTorch*.

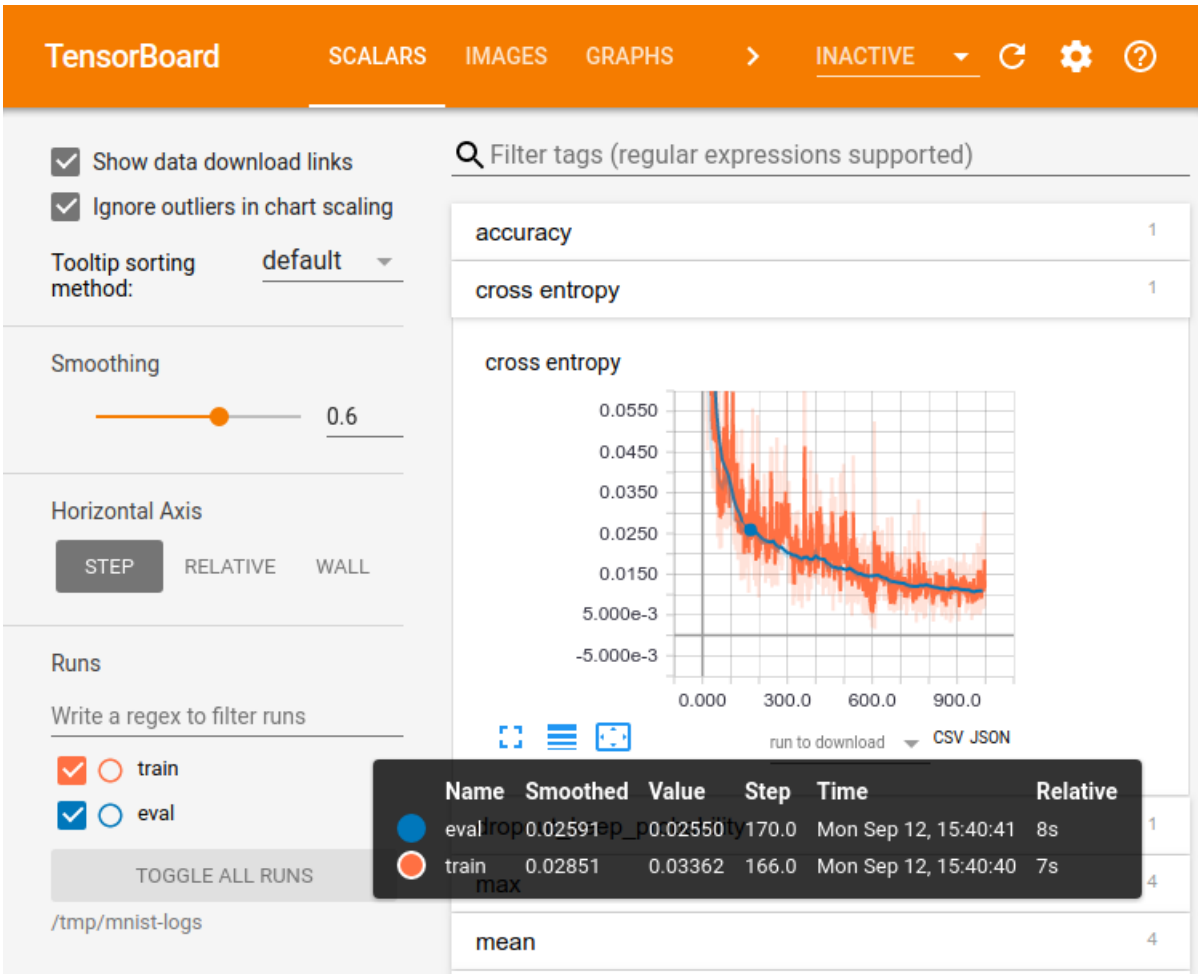
O *TensorFlow* é uma biblioteca de código aberto de *machine learning* para criação e treinamento de sistemas de redes neurais e detecção de padrões, tendo sido desenvolvido originalmente pela Google Brain Team e lançado em 2015, oferece APIs para pesquisa e desenvolvimento *desktop*, *mobile*, *web* e *cloud* (TENSORFLOW, 2019).

O *TensorFlow* permite treinar e executar uma CNN para classificação e detecção de objetos em imagens, classificação de vídeos, reconhecimento de fala, aprendizado por reforço e outros. A biblioteca *TensorFlow* pode ser utilizada tanto em plataformas, Windows, Linux, Max

OS e dispositivos moveis, sendo disponibilizado através das linguagens de programação Python, C++, Java e GO (ABADI *et al.*, 2016).

A biblioteca possui algumas ferramentas extras, que podem ser úteis durante o desenvolvimento de projetos. Uma das ferramentas de auxílio é o *TensorBoard*, um conjunto de aplicativos web que permitem fazer a visualização dos grafos e estatísticas do *TensorFlow*. A Figura 24 mostra o ambiente do TensorBoard para visualização de grafos e gráficos.

Figura 24 – *TensorBoard*

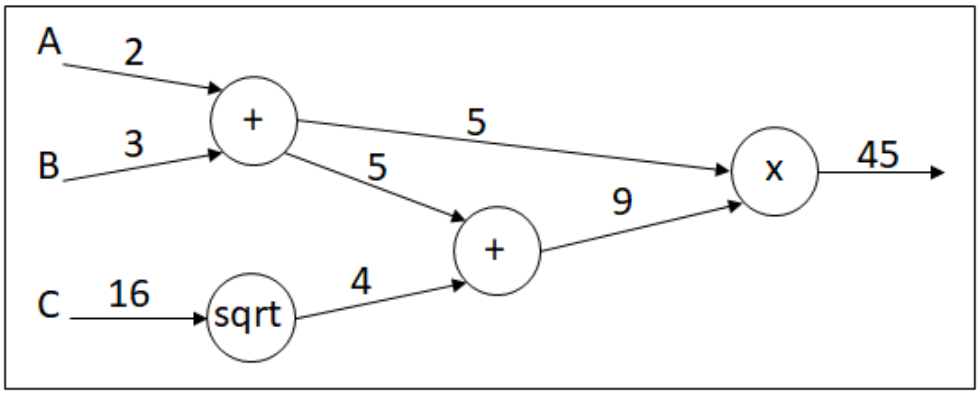


Fonte: (TENSORFLOW, 2019)

O *TensorFlow* permite expressar cálculos como gráficos de fluxos de dados estáticos, onde cada nó do gráfico representa uma operação matemática, enquanto as arestas do gráfico representam as matrizes de dados multidimensionais (tensores) comunicadas entre eles (ABADI *et al.*, 2016). Na Figura 25 é mostrada a representação de um fluxo de grafos no TensorFlow da Equação 2.8.

$$Y = ((A + B) + sqrt(16)) * A + B \tag{2.8}$$

Figura 25 – Representação de um grafo de fluxo no *TensorFlow*



Fonte: (Autor, 2019)

O *PyTorch* é uma biblioteca de código aberto para *machine learning*, baseada em *Torch*. Desenvolvida inicialmente pelo Facebook em 2016 e escrita em Python, C e CUDA. É uma biblioteca utilizada por empresas como Twitter e NVidea para aplicações que se utilizam de visão computacional e processamento de linguagem natural, tornando-se popular ao permitir construir de maneira mais simplificada arquiteturas complexas (NGUYEN *et al.*, 2019).

O *PyTorch* suporta o processamento em Linux, Mac OS e Windows, e diferentemente do *TensorFlow* que possui gráficos de dados estáticos o *PyTorch* possui gráficos de fluxos de dados dinâmicos o que significa que o gráfico é definido dentro loop onde ocorre a iteração dos dados (HEGHEDUS; CHAKRAVORTY; RONG, 2019).

2.6 CONSIDERAÇÕES SOBRE O CAPÍTULO

Dentro do contexto do que foi visto neste capítulo, e analisando os trabalhos relacionados com a área, é possível construir uma base para o desenvolvimento de um *software* que permita realizar a análise e a identificação das emoções através das expressões faciais de estudantes durante atividade em sala de aula. Para realizar a construção do *software* é necessário interligar os conceitos e as diferentes áreas de conhecimento como *IoT*, sala de aula inteligente, visão computacional e redes neurais artificiais.

Conforme o que foi observado nos trabalhos relacionados, é possível notar um aumento nas pesquisas desenvolvidas nestas áreas nos últimos anos. Alguns destes trabalhos buscam fazer o reconhecimento de emoções e sentimentos, assim como a identificação do nível de interesse e a concentração dos estudantes, através da análise de imagens. Ao identificar as dificuldades encontradas pelos estudantes é possível propor um recurso que auxilie nos processos de aprendizagem.

Algumas das plataformas tem se destacado na realização de tarefas como reconhecimento

e análise, entre elas estão a linguagem de programação *Python*, sendo muito utilizada juntamente com a biblioteca *OpenCV*. Outra plataforma que tem ganhado espaço na área de pesquisa e desenvolvimento é a biblioteca *TensorFlow*, muito utilizada para a construção de uma CNN.

O próximo capítulo trata da proposta de implementação do *software*, sendo abordadas quais características são utilizadas para análise das expressões faciais e como se deu a construção da CNN. Também são detalhadas algumas das particularidades dos *softwares* que foram utilizadas durante a realização deste projeto.

3 PROJETO: ANÁLISE DE IMAGENS EM SALA DE AULA

Por meio do estudo realizado e descrito no capítulo anterior, identificou-se que existem poucas iniciativas de sistemas que observem e monitorem as atividades dos estudantes em sala de aula, com a finalidade de avaliar o interesse e a motivação com o aprendizado. Reconhece-se, contudo, que a motivação do estudante exerce um papel fundamental durante o processo de aprendizagem, não impactando apenas nele, mas também em todos que estão ao seu redor. Um estudante distraído, que não consegue manter a concentração, pode acabar atrapalhando e desconcentrando todos os outros, assim prejudicando todo o andamento da aula. Dentro deste contexto, este trabalho busca avaliar o envolvimento dos estudantes com a aula.

Neste capítulo, inicialmente é apresentada a visão geral do projeto e quais componentes motivacionais foram analisados e reconhecidos. Posteriormente, são apresentados os *softwares* utilizados para realizar a captura e a análise das imagens, além do desenvolvimento, treinamento e avaliação da CNN que foi desenvolvida. Ao final do capítulo são apresentados testes que foram realizados com alguns participantes.

3.1 VISÃO GERAL DO PROJETO

Por se tratar de um contexto onde é possível obter a expressão facial dos estudantes, assume-se como componente motivacional o estado emocional inferido a partir da face do estudante. Para a coleta das imagens podem ser utilizados dispositivos *IoT* integrados a uma sala de aula. A coleta de imagens realizada dentro de um ambiente de ensino permite o reconhecimento dos diferentes tipos de emoções a partir da análise das expressões faciais.

O reconhecimento das emoções dos estudantes pode ser útil à medida que o professor, tomando conhecimento deles, pode agir apropriadamente. Ao fornecer um *feedback* ao professor, isso permite a ele reavaliar suas metodologias de ensino, possibilitando desta forma que ele possa adequar sua maneira de dar aula para os diferentes tipos de turmas. Desta forma, um *feedback* ao professor pode contribuir para o seu aprimoramento e desenvolvimento de uma sala de aula inteligente, objetivos deste trabalho.

3.2 EMOÇÕES

As emoções são manifestações automáticas, decorrentes de alterações fisiológicas e comportamentais provocadas por estímulos internos ou externos (GORDIN; LOIOLA, 2015). Ao ser exposto a um estímulo, o cérebro libera hormônios que alteram o estado emocional da pessoa e cada um reage de uma maneira, de acordo com as experiências passadas.

Existem diferentes tipos de emoções, as assim chamadas emoções primárias, secundárias

ou de fundo (DAMÁSIO, 2000). As emoções primárias são as mais perceptíveis para aqueles que estão ao nosso redor, sendo emoções tais como: alegria, tristeza ou surpresa. As emoções primárias são demonstradas facilmente através de expressões faciais. As emoções secundárias são menos visíveis, como culpa ou orgulho. Já as emoções de fundo não são perceptíveis, tais como: bem-estar ou mal-estar.

A maioria das emoções pode se manifestar através de expressões faciais por um breve período de tempo, devido a um estímulo. Segundo pesquisa realizada, os seres humanos são capazes de expressar cerca de 27 tipos diferentes de emoções para as mais variadas situações do cotidiano (COWEN; KELTNER, 2017). Para o desenvolvimento deste trabalho foram selecionadas algumas das emoções primárias (visíveis nas expressões faciais) sendo elas: alegria, surpresa, tristeza, nojo, raiva, medo e a expressão neutra. Cada uma das emoções possui uma característica capaz de as diferenciar das outras, sendo elas representadas na Figura 26.

Figura 26 – Características das expressões faciais

Fonte: (SILVA, 2016)

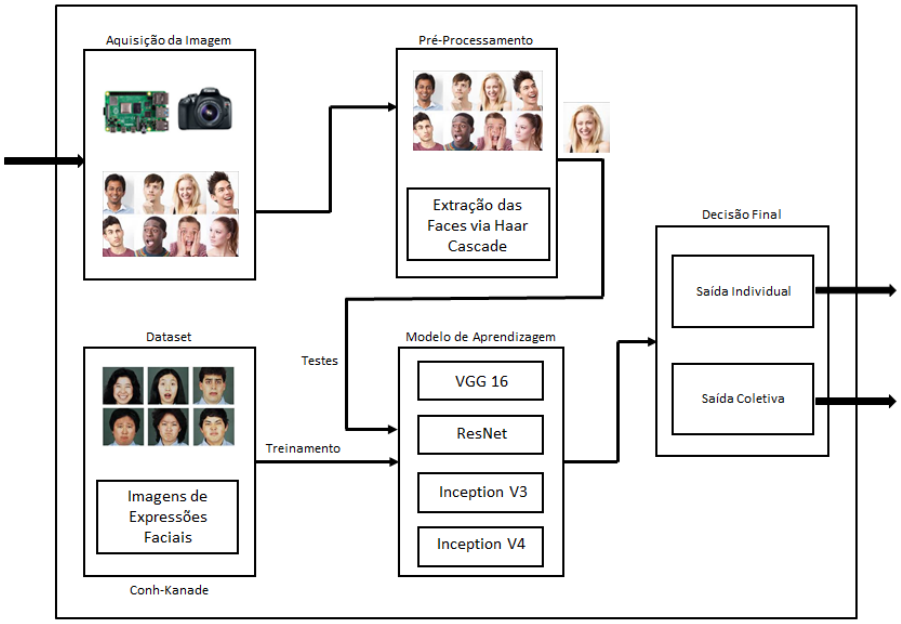
3.3 DEFINIÇÃO DO PROJETO

Nesta seção são definidos e especificados os elementos de *software* que compõem o projeto e foram utilizados no decorrer do trabalho, sendo que o desenvolvimento está dividido em três etapas: aquisição das imagens, implementação da CNN e testes.

A Figura 27 exibe o fluxo de desenvolvimento do projeto, iniciando na etapa de aquisição das imagens, etapa esta onde são coletadas as imagens dos estudantes durante atividade em sala de aula. Devido as medidas excepcionais de afastamento social do período, que forçaram o cancelamento das atividades presenciais em sala de aula, esta etapa foi prejudicada. Optou-se então por realizar a coleta de imagens em ambiente domiciliar, simulando uma disposição de estudantes em sala de aula. Após a aquisição das imagens, elas devem ser repassadas para a etapa do pré-processamento, que realizará a extração das faces utilizando o algoritmo de visão computacional denominado *Haar Cascade*.

Após a etapa de pré-processamento, a face então extraída segue para a etapa de modelo de aprendizagem, cujo treinamento foi realizado a partir do *dataset* Conh-Kanade. Já na etapa do modelo de aprendizagem, as imagens contendo as faces passam por uma CNN já desenvolvida e treinada que realiza o reconhecimento e a classificação da emoção entre alegria, surpresa, tristeza, nojo, raiva, medo ou neutra. Após a emoção ser classificada o resultado segue então para a última etapa, que é a saída, onde são computados os resultados de maneira a fornecer uma análise individual (emoção de cada estudante) ou coletiva (emoção geral da turma) das emoções reconhecidas através das expressões faciais.

Figura 27 – Diagrama de estrutura do projeto



Fonte: (Autor, 2019)

3.4 *SOFTWARES*

Para realizar o desenvolvimento do projeto foi utilizada a linguagem *Python* [1], através do ambiente de programação *Jupyter Notebook* [2]. O *Python* permite o uso de bibliotecas como o *OpenCV* [3] e o *TensorFlow* [4], além de outras como *Keras* [5] e *Sklearn* [6] que foram utilizadas durante o andamento do trabalho.

A primeira etapa realizada foi a instalação da plataforma Anaconda [7], programa que permite a utilização de *softwares* como o *Jupyter Notebook*, além de suas bibliotecas. Após ser feita a instalação do *Jupyter Notebook*, foram instaladas as principais bibliotecas através das linhas de comando que são demonstrados no algoritmo 1.

Algoritmo 1 – Instalação das bibliotecas

```
conda install -c conda-forge opencv
conda install -c conda-forge tensorflow
conda install -c conda-forge keras
conda install -c anaconda scikit-learn
conda install -c anaconda pandas
conda install -c anaconda seaborn
```

O *OpenCV* (do inglês *Open Source Computer Vision Library*) é uma biblioteca utilizada para visão computacional, de código aberto e de uso livre para fins comerciais e acadêmicos. Inicialmente desenvolvida pela Intel Corporation e escrita nas linguagens de programação C/C++, ela implementa uma variedade de ferramentas para o processamento de imagens, realizando desde operações simples até operações complexas como análise de imagens e vídeos, detecção e reconhecimento facial em tempo real (FOUNDATION, 2019).

A biblioteca *OpenCV* pode ser utilizada em diversas linguagens de programação como C++, *Python*, *Java* e outras, sendo compatível com sistemas, Windows, Linux e Mac OS. O *OpenCV* pode ser utilizado no desenvolvimento de aplicativos para diversas áreas do conhecimento e possui uma estrutura modular, sendo seus métodos divididos em diversos grupos (FOUNDATION, 2019), tais como:

a) Funcionalidades Principais

b) Processamento de Imagem

c) Análise de Vídeo

[1] Versão: 3.7.6, Disponível em: https://www.python.org - acesso em: 02 nov. 2019
[2] Versão: 6.0.3, Disponível em: https://jupyter.org - acesso em: 02 nov. 2019
[3] Versão: 3.4.2, Disponível em: https://opencv.org - acesso em: 01 out. 2019
[4] Versão: 2.0.0, Disponível em: https://www.tensorflow.org - acesso em: 01 out. 2019
[5] Versão: 2.3.1, Disponível em: https://keras.io - acesso em: 02 mar. 2020
[6] Versão: 0.22.1, Disponível em: https://scikit-learn.org/stable/index.html - acesso em: 02 mar. 2020
[7] Versão: 1.9.7, Disponível em: https://www.anaconda.com - acesso em: 10 nov. 2019

d) Calibração de Câmera e Reconstrução 3D

e) Estruturas de Recursos 2D

f) Detecção de Objetos

g) *High-level-GUI*

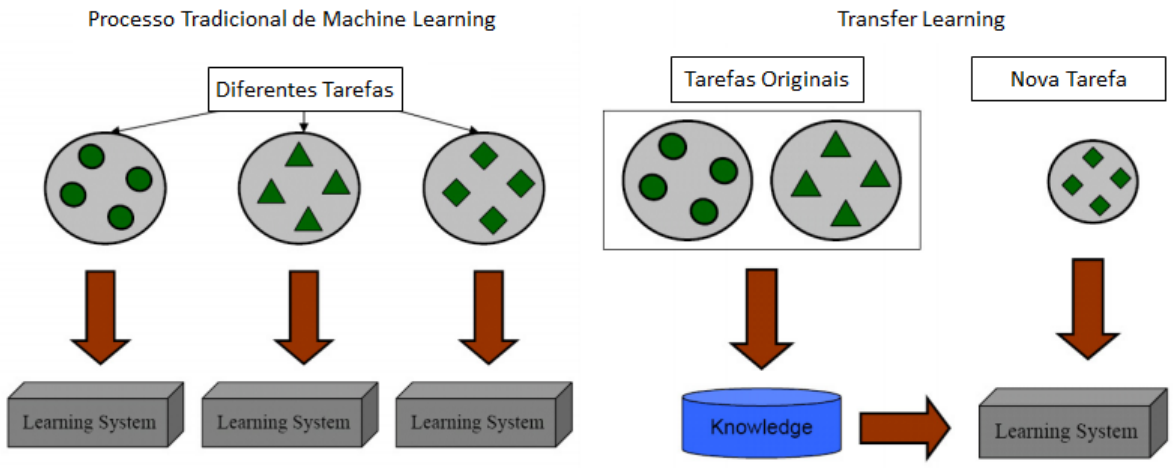
h) Entrada e Saída de Vídeo

3.5 DESENVOLVIMENTO DA REDE NEURAL CONVOLUCIONAL

Existem atualmente modelos de redes neurais convolucionais (CNN) pré-treinadas, para uso variado, que aceleram o desenvolvimento de produtos de software nesta área. Diversos desenvolvedores (Google, Microsoft, entre outros) disponibilizam suas implementações de redes neurais para que sejam adaptadas e reutilizadas em outros cenários, na forma de pacotes de software. Portanto, nesta etapa pode ser utilizado o processo de *Transfer Learning* (do inglês, transferência de aprendizado), que possibilita que uma CNN já treinada seja reutilizada com um novo conjunto de dados. Desta forma ela aprende incrementalmente novas classificações (PAN; YANG, 2010).

Diferentemente do modelo tradicional de implementação de *machine learning*, onde para cada problema a CNN deve ser treinada separadamente, no modelo de *transfer learning* utiliza-se uma CNN já treinada para diferentes tarefas, sendo feito um novo treinamento nela para que possa realizar a tarefa desejada, desta forma adicionando conhecimento a ela. A Figura 28 apresenta a diferença entre o modelo tradicional de treinamento e o modelo de *transfer learning*.

Figura 28 – Modelo tradicional de *machine learning* e *transfer learning*

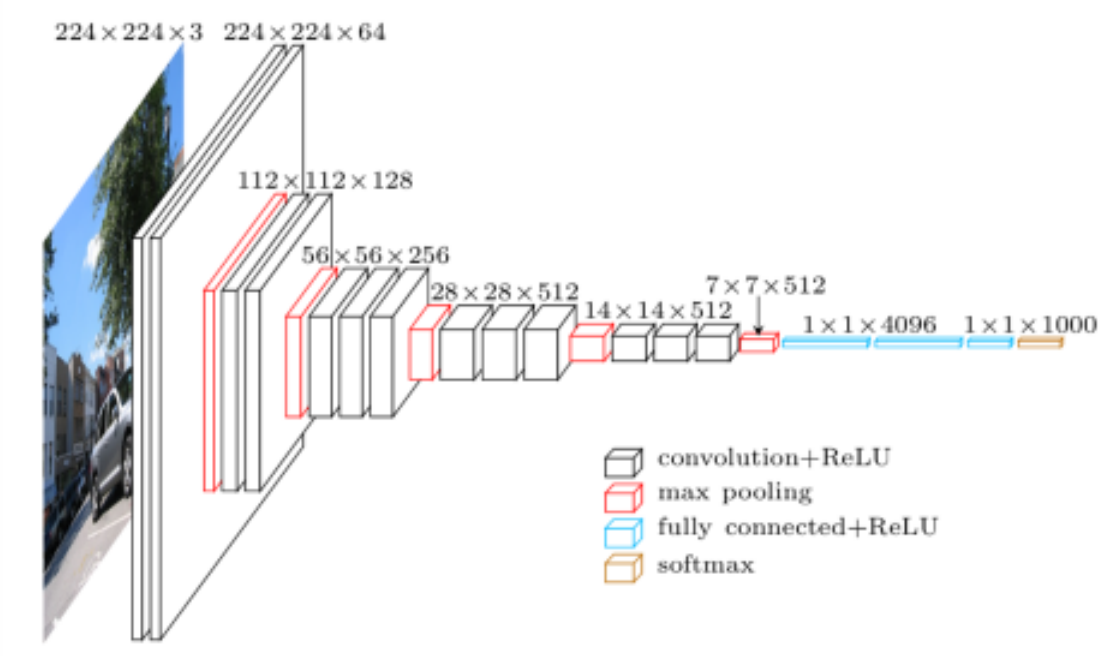


Fonte: (PAN; YANG, 2010)

Para realizar o desenvolvimento e treinamento de maneira a reconhecer expressões faciais, foi utilizada como base a CNN denominada VGG16 (SIMONYAN; ZISSERMAN, 2014b).

A VGG16 é um modelo melhorado da AlexNet (nome dado a sua versão anterior, bem conhecida) para classificação, sendo que ela possui entradas de 224x224x3 e dezesseis camadas variando entre convolução e *pooling*. A arquitetura padrão foi treinada com milhares de imagens e é capaz de realizar a classificação entre diversos tipos de classes, sendo esta arquitetura mostrada na Figura 29.

Figura 29 – Arquitetura padrão VGG16



Fonte: (ROSEBROCK, 2017)

Durante o desenvolvimento, a primeiro etapa foi a importação da CNN utilizando a biblioteca *keras*, e a partir dessa rede base foram feitas mudanças e testes para se chegar ao resultado desejado, que é a identificação dos sete diferentes tipos de expressões faciais.

A importação da CNN foi feita de maneira a desconsiderar as últimas camadas. As últimas camadas, chamadas totalmente conectadas realizam a função de classificação das imagens em uma das saídas de domínio. Na CNN VGG16 original a etapa de classificação é compostas por quatro camadas variando entre *flaten* e densa, sendo capaz de identificar diversos tipos diferentes de imagens.

Na versão modificada, a etapa de classificação foi substituída por seis novas camadas variando entre *flaten*, densa e *dropout*, com ativação *ReLu* e *softmax* para o reconhecimento das expressões desejadas. O código desenvolvido nessa etapa é mostrado no algoritmo 2 e a diferença entre a composição original e a modificada são mostradas na Figura 30.

Algoritmo 2 – Código da saída da CNN VGG16 - Modificada

```
vgg = keras.applications.VGG16(input_shape = (224,224,3),
    include_top = False, weights = 'imagenet')

output = vgg.output
flat = Flatten()(output)
dense1 = Dense(3078, activation = 'relu')(flat)
drop1 = Dropout(0.5)(dense1)
dense2 = Dense(256, activation = 'relu')(drop1)
drop2 = Dropout(0.2)(dense2)
out = Dense(7, activation = 'softmax')(drop2)

tf_model = Model(inputs = vgg.input, outputs = out)
```

Figura 30 – Comparação entre camadas de saída da CNN original e modificada

Classificação VGG16 - Original

flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

Classificação VGG16 - Modificada

flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 3078)	77223942
dropout_1 (Dropout)	(None, 3078)	0
dense_2 (Dense)	(None, 256)	788224
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 7)	1799

Fonte: (Autor, 2020)

Ao final da etapa de criação da CNN a nova versão apresentou um total de vinte e quatro camadas, totalizando 92.728.653 parâmetros a serem treinados, diferentemente dos 138.357.544 parâmetros que a CNN original possui. Na Figura 31 é mostrado o sumário da CNN VGG16 - Modificada e as camadas que a compõem. Já na Figura 32 é mostrada uma ilustração de como ficou a arquitetura da CNN.

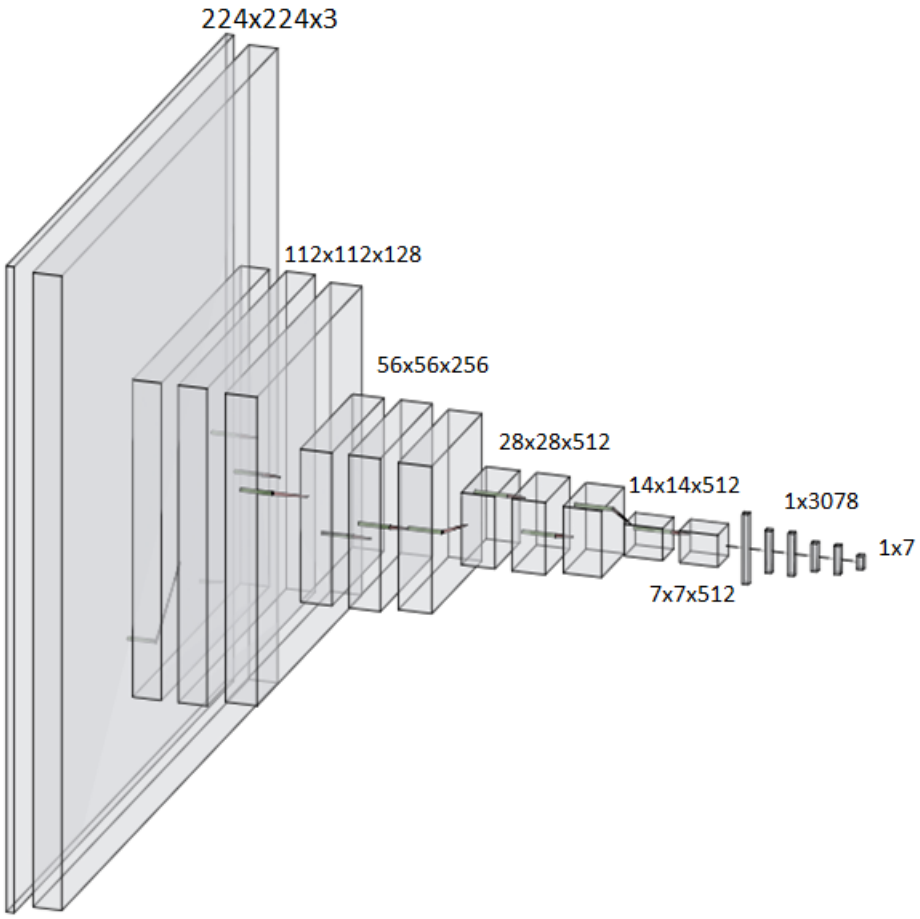
Figura 31 – Sumário final da CNN VGG16 - Modificada

```
In [4]: tf_model.summary()
```

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 3078)              77223942
_________________________________________________________________
dropout_1 (Dropout)          (None, 3078)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               788224
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 1799
=================================================================
Total params: 92,728,653
Trainable params: 92,728,653
Non-trainable params: 0
_________________________________________________________________
```

Fonte: (Autor, 2020)

Figura 32 – Arquitetura da CNN VGG16 - Modificada



Fonte: (Autor, 2020)

Além da adaptação do modelo VGG16 foi proposta a criação e treinamento de uma nova CNN denominada VGG16+ com base nos testes realizados com outras modelos. Alguns testes sugeriam que CNNs que possuíam menores números de camadas convolucionais e de *pooling* demonstravam uma melhor taxa de classificação.

Desta maneira foi feita a implementação de uma nova CNN com base nas que demonstraram melhores taxas. A entrada da CNN continuou sendo 224x224x3, e a esta entrada foram adicionadas camadas de convolução com ativação *relu* e camadas de *pooling* para reduzir a dimensionalidade das informações.

No novo modelo também foram adicionadas as últimas camadas para a classificação das imagens entre as classes de domínio. O código desenvolvido para a implementação da CNN VGG16+ está descrito no algoritmo 3. Ao final a CNN VGG16+ apresentou 19 camadas totalizando 33.368.455 parâmetros sendo estes demonstrados na Figura 33.

Algoritmo 3 – Código de implementação da CNN VGG16+

```
inp = Input((224,224,3))

conv1 = Conv2D(64, (5,5), padding = 'valid', activation = 'relu')(inp)
conv1 = MaxPooling2D(pool_size = (2,2))(conv1)
conv1 = BatchNormalization()(conv1)
conv2 = Conv2D(128, (4,4), padding = 'valid', activation = 'relu')(conv1)
conv2 = MaxPooling2D(pool_size = (2,2))(conv2)
conv2 = BatchNormalization()(conv2)
conv3 = Conv2D(256, (3,3), padding = 'valid', activation= 'relu')(conv2)
conv3 = MaxPooling2D(pool_size = (2,2))(conv3)
conv3 = BatchNormalization()(conv3)
conv4 = Conv2D(512, (3,3), padding = 'valid', activation= 'relu')(conv3)
conv4 = MaxPooling2D(pool_size = (2,2))(conv4)
conv4 = BatchNormalization()(conv4)

flat = Flatten()(conv4)
dense1 = Dense(512, activation = 'relu')(flat)
drop1 = Dropout(0.5)(dense1)
dense2 = Dense(64, activation = 'relu')(drop1)
drop2 = Dropout(0.1)(dense2)
out = Dense(7, activation = 'softmax')(drop2)

tf_model = Model(inp, out)
```

Figura 33 – Sumário final da CNN VGG16+

```
In [13]: tf_model.summary()
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 220, 220, 64)      4864
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 110, 110, 64)      0
_________________________________________________________________
batch_normalization_1 (Batch (None, 110, 110, 64)      256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 107, 107, 128)     131200
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 53, 53, 128)       0
_________________________________________________________________
batch_normalization_2 (Batch (None, 53, 53, 128)       512
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 51, 51, 256)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 25, 25, 256)       0
_________________________________________________________________
batch_normalization_3 (Batch (None, 25, 25, 256)       1024
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 23, 23, 512)       1180160
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 11, 11, 512)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 11, 11, 512)       2048
_________________________________________________________________
flatten_1 (Flatten)          (None, 61952)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               31719936
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                32832
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 455
=================================================================
Total params: 33,368,455
Trainable params: 33,366,535
Non-trainable params: 1,920
_________________________________________________________________
```

Fonte: (Autor, 2020)

3.6 TREINAMENTO DA REDE NEURAL CONVOLUCIONAL

Para realizar o treinamento das CNNs foi necessário primeiro definir qual seria o *dataset* utilizado durante esta etapa. Após definido o *dataset* foram separadas as imagens em grupos contendo as mesmas expressões faciais, também foi necessário realizar um pré-processamento nestas imagens para que elas estivessem de acordo com os requisitos exigidos. Somente após o pré-processamento das imagens foi então possível realizar o treinamento e a avaliação.

3.6.1 *Dataset*

Para realizar a análise das expressões faciais e a identificação das emoções representadas por elas, foi utilizado o *dataset* Conh-Kanade (KANADE; COHN; TIAN, 2000). O *dataset* Conh-Kanade é um *dataset* desenvolvido para análise de expressões faciais automáticas.

O *dataset* possui 1917 imagens de expressões faciais de 182 pessoas, entre elas homens e mulheres de 18 a 50 anos de vários grupos étnicos. A resolução das imagens é de 640x490 para as imagens em escala de cinza 8-bit e 640x480 para imagens coloridas de 24-bit.

A partir dessas imagens foi possível realizar a representação de sete emoções distintas tais como: alegria, surpresa, tristeza, nojo, raiva e medo, além da expressão neutra, pois o *dataset* contém sequências de imagens partindo sempre da expressão neutra até chegar a expressão que esta sendo demonstrada. A Figura 34 mostra um exemplo de sequência de imagens da expressão facial de surpresa.

Figura 34 – Sequência de imagens do *dataset* Conh-Kanade

Fonte: (KANADE; COHN; TIAN, 2000))

3.6.2 Pré-Processamento

As imagens originais do *dataset* são divididas em pastas de acordo com cada uma das expressões faciais, sendo necessário primeiro classificar todas as imagens e juntá-las em sete diferentes pastas, sendo cada uma das pastas correspondente a uma expressão facial. Para o treinamento foram utilizadas 100 imagens de cada uma das expressões, sendo selecionadas pelo autor as imagens que melhor representavam cada uma das expressões faciais.

Os primeiros testes de treinamento não apresentaram uma boa convergência de resultados devido a todas as imagens conterem um alto número de *pixels* cinza que acabavam por saturar a CNN, fazendo com que as taxas de identificação correta fossem muito baixas. Estes *pixels* cinzas estavam presentes devido ao fundo das imagens serem todos muito parecidos. Na Figura 35 está uma das imagens no formato original que foram utilizada durante as primeiras fases do treinamento.

Figura 35 – Imagem original

Fonte: (KANADE; COHN; TIAN, 2000)

Para resolver o problema de saturação de *pixels* cinza, foi utilizado um processo para recortar somente a face contida na imagem original, gerando uma nova imagem. Para identificar a face presente na imagem e recortá-la foi utilizado o algoritmo de visão computacional *Haar Cascade*.

O *Haar Cascade* é um algoritmo de *machine learning*, utilizado para detecção e reconhecimento de faces e objetos em imagens e vídeos (VIOLA; JONES, 2001). O algoritmo utiliza funções em cascata para identificar as diversas características de uma imagem e as utiliza para detectar e classificar os objetos que a compõem. O algoritmo necessita ser treinado com muitas imagens positivas (imagens que possuem o objeto a ser detectado) e negativas (imagens que não possuem o objeto a ser detectados), sendo dividido em 4 estágios:

a) Seleção de recursos *Haar*

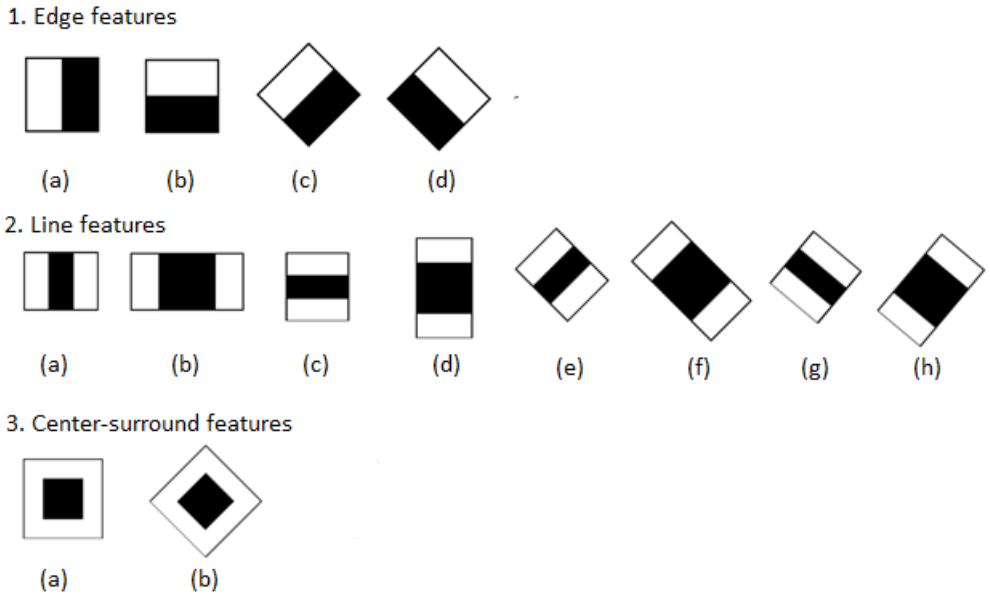
b) Imagens integrais

c) Treinamento *AdaBoost*

d) Classificadores em cascata

O algoritmo *Haar Cascade* extrai as características das imagens utilizando filtros, semelhantes aos das CNNs. Os filtros são chamados de recurso *Haar* e deslizam pela imagem calculando as intensidades dos *pixels* brancos e pretos, desta forma extraindo as suas características (LIENHART; MAYDT, 2002). Os diferentes tipos de filtros são demonstrados na Figura 36. Para auxiliar no processo de extração das características são utilizados as imagens integrais que simplificam os cálculos e os tornam mais rápidos.

Figura 36 – Filtros *Haar Cascade*

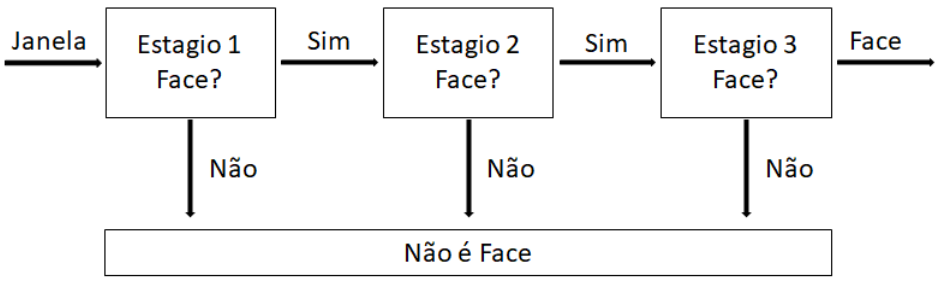


Fonte: (LIENHART; MAYDT, 2002)

Nem todos os recursos extraídos são bons para a classificação de uma imagem, por isso é utilizado o *AdaBoost*, que seleciona os melhores recursos e treina os classificadores que o reconhecem. O *AdaBoost* combina vários classificadores fracos, que sozinhos não são capazes de classificar uma imagem, para formar um classificador forte que é capaz de realizar a classificação.

Os classificadores em cascata aplicam os recursos nas janelas em estágios, somente passando ao próximo estágio se o recurso testado for aprovado. Em caso de falha em um estágio, todos os outros recursos são descartados e nada mais é testado naquela janela, passando assim para a próxima janela. Desta maneira o algoritmo não perde tempo testando e aplicando todos os recursos em uma janela de forma desnecessária. Um exemplo de classificador em cascata é demonstrado na Figura 37.

Figura 37 – Classificador em cascata



Fonte: (Autor, 2019)

Além do algoritmo *Haar Cascade* foram utilizadas as funções *imread()* (para leitura das imagens) e *imwrite()* (para salvar as novas imagens), sendo estas funções implementadas através da biblioteca *OpenCV*. O código apresentado no algoritmo 4 mostra o processo de leitura de uma imagem, identificação do rosto e a geração de uma nova imagem, sendo um exemplo das imagens geradas ao final desta etapa de pré-processamento mostrada na Figura 38.

Ao realizar novo treinamento com as novas imagens geradas foi obtido um real ganho de desempenho na classificação das expressões faciais.

Algoritmo 4 – Código para extração do rosto

```
classificador = cv2.CascadeClassifier('haarcascade_frontalface.xml')

imagens_dir = glob.glob(os.path.join('C:\\emocao\\', '*'))

for i in range(len(imagens_dir)):
    imagem_dir = imagens_dir[i]
    nome_imagem = imagem_dir.split('\\')

    imagem = cv2.imread(imagem_dir)
    imagemCinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
    facesDetectadas = classificador.detectMultiScale(imagemCinza)

    for (x,y,l,a) in facesDetectadas:
        rosto = imagem[y:(y+a), x:(x+l)]
        cv2.imwrite("C:\\emocao\\rosto\\" + nome_imagem[2], rosto );
```

Figura 38 – Imagem recortada

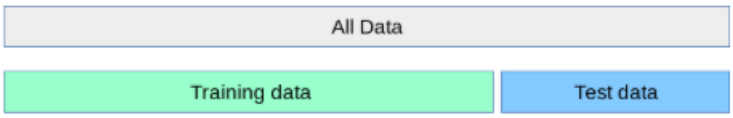
Fonte: (Autor, 2020)

3.6.3 Treinamento

Para poder realizar o treinamento de uma CNN é necessário separar os dados entre dados de treino e dados de teste. Um dos métodos para fazer essa divisão é o denominado *Hold Out*, implementado pelo método *train_test_split()* da biblioteca *sklearn*.

No referido método o conjunto de dados é dividido de acordo com os parâmetros utilizados. Normalmente são destinados 70% dos dados para treinamento e 30% para teste. Essa divisão é mostrada na Figura 39. Para obter um melhor resultado, pode ser realizado o treino diversas vezes recombinando os dados para formar diferentes conjuntos de treino e de teste. O parâmetro *epoch* conta as iterações usadas no treinamento, sendo uma possível condição de parada.

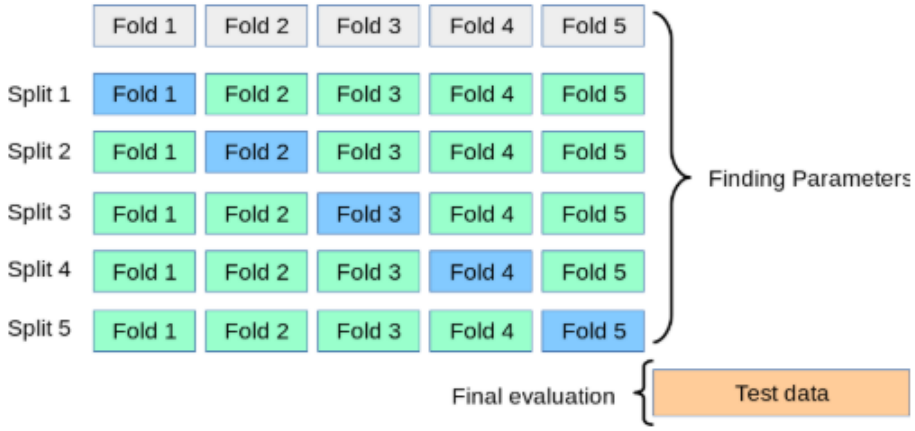
Figura 39 – Método *Hold Out*



Fonte: (SCIKITLEARN, 2020)

Outro método existente para separar os conjuntos de treino e teste é o *Cross-Validation*. Nesse método, os dados são divididos de forma estratificada ou aleatória em k partições, sendo mutualmente exclusivos e de tamanho idêntico. A partir do valor definido o treinamento é realizado k vezes, onde para cada um dos treinos é utilizado um conjunto diferente. Assim, ao final, todos os dados são utilizados tanto para treinar como para avaliar os resultados obtidos. A Figura 40 mostra a divisão dos dados, considerando o método *cross-validation* para k=5.

Figura 40 – Método *Cross-Validation* para 5 partições



Fonte: (SCIKITLEARN, 2020)

Ambos os métodos de amostragem foram utilizados durante o treinamento da CNN. No método *Hold Out* foi utilizada um particionamento dos dados de 70% para treino e 30% para teste. O código desenvolvimento para a separação para os conjuntos de dados para treinamento é demonstrada no algoritmo 5.

Algoritmo 5 – Código de treinamento utilizando o método *Hold Out*

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size = 0.3,
    random_state = 42, stratify = y)

tf_model.compile(optimizer = 'RMSprop', loss = 'categorical_crossentropy',
    metrics = ['acc'])

tf_model.fit(X_train, y_train, batch_size = 1, epochs = 500, initial_epoch
    = 0, validation_data = (X_val, y_val))
```

Em ambos os métodos de treinamento o parâmetro *epoch* (épocas de treinamento) foi utilizado com o valor 500, ou seja, para cada um dos testes a CNN foi treinada 500 vezes (para cada instância). Alcançou-se este valor a partir dos sucessivos testes realizados que indicaram que este seria um número ideal de épocas de treinamento.

Já no método *Cross-Validation* foi utilizado o número de partições k = 10 (valor default). Desta forma, são utilizadas 9 partições dos dados para treino e uma para teste, sempre alternando os conjuntos de treino. O código desenvolvido para a separação dos dados usados no treinamento esta demonstrado no algoritmo 6.

Algoritmo 6 – Código de treinamento utilizando o método *Cross-Validation*

```
kf = KFold(n_splits = 10)

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    tf_model.fit(X_train, y_train, batch_size = 1, epochs = 50,
        initial_epoch = 0, validation_data = (X_test, y_test))
```

Ao realizar testes com os 2 métodos foi observado que o *Cross-Validation* apresentou uma melhor eficácia e menos necessidade de treinamento para chegar aos resultados obtidos com o método *Hold-Out*.

Durante esta etapa de treinamento foi utilizada também a biblioteca *pickle* [8], para salvar e carregar os estados de cada treinamento da CNN, através dos métodos *dump()* e *load()*, cujo o código pode ser visto no algoritmo 7. Com o uso dessa biblioteca foi possível fazer uma

[8] Disponível em: https://docs.python.org/3.7/library/pickle.html - acesso em: 02 mar. 2020

série de treinamentos e em seguida salvar o seu estado para que então pudesse ser analisada a sua evolução, podendo então prosseguir como o treinamento ate observar que não havia mais evolução significativa na CNN.

Algoritmo 7 – Código para salvar e carregar os estados da CNN

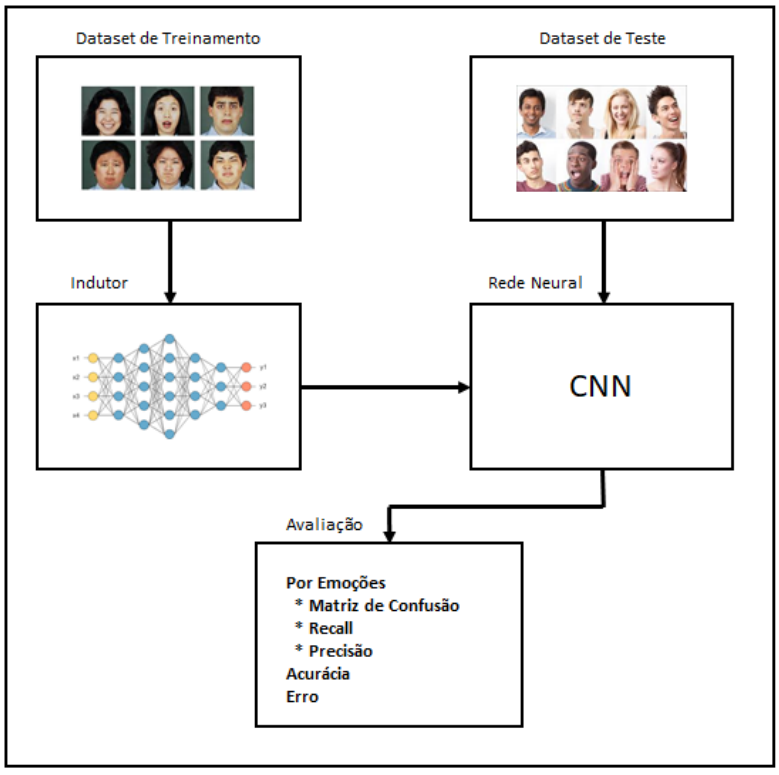
```
with open('Emocoes.pkl', 'wb') as file:
    pickle.dump(tf_model, file)

with open('Emocoes.pkl', 'rb') as file:
    tf_model = pickle.load(file)
```

3.7 ANÁLISE E AVALIAÇÃO DOS RESULTADOS PARA O *DATASET* DE TREINO

Durante a etapa de análise de desempenho das CNNs, foram utilizadas métricas padronizadas comuns na área (método de amostragem), para a avaliação dos resultados, sendo que para essa análise foi utilizado o conjunto de imagens de treino. A Figura 41 mostra o diagrama de avaliação do modelo das CNNs.

Figura 41 – Avaliação do modelo de CNN



Fonte: (Autor, 2020)

Uma das representações utilizadas durante esta etapa foram as matrizes de confusão. Em uma matriz de confusão os resultados de classificação para cada uma das classes são colocados de maneira com que possam ser analisados, de forma a mostrar a frequência com que ocorrem os erros e acertos do classificador. Uma matriz de confusão é exibida na Figura 42, sendo ela composta por quatro tipos diferentes de classificação:

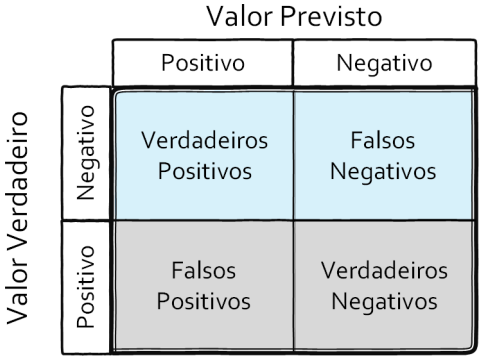
a) **Verdadeiro Positivo (VP):** essa classificação ocorre quando o resultado positivo obtido corresponde ao esperado, ou seja, a instância foi classificada corretamente como sendo da classe positivo. Ex.: Ao analisar uma imagem que contém uma expressão facial de alegria o modelo classificou a imagem como alegria, ou seja, de maneira correta.

b) **Verdadeiro Negativo (VN):** essa classificação ocorre quando o resultado negativo obtido corresponde ao esperado, ou seja, a instância foi classificada corretamente como sendo da classe negativo. Ex.: Ao analisar uma imagem que não contém uma expressão facial de alegria o modelo classificou a imagem como não sendo alegria, ou seja, pertencendo a outra emoção de forma correta.

c) **Falso Positivo (FP):** essa classificação ocorre quando o resultado foi previsto como sendo da classe positivo, mas de maneira incorreta. Ex.: Ao analisar uma imagem que não contém uma expressão facial de alegria o modelo classificou a imagem como sendo alegria, ou seja, de maneira incorreta.

d) **Falso Negativo (FN):** essa classificação ocorre quando o resultado foi previsto como sendo da classe negativo, de maneira incorreta. Ex.: Ao analisar uma imagem que contém uma expressão facial de alegria o modelo classificou a imagem como não sendo alegria, ou seja, de maneira incorreta.

Figura 42 – Matriz de confusão



Fonte: (RODRIGUES, 2019)

A partir de uma matriz de confusão é possível calcular-se as seguintes métricas para a análise de resultados:

a) **Acurácia:**

$$Acc = \frac{VerdadeirosPositivos(VP) + VerdadeirosNegativos(VN)}{Total} \tag{3.1}$$

b) **Precisão:**

$$Prec = \frac{VerdadeirosPositivos(VP)}{VerdadeirosPositivos(VP) + FalsosPositivos(FP)} \tag{3.2}$$

c) **Recall:**

$$Recall = \frac{VerdadeirosPositivos(VP)}{VerdadeirosPositivos(VP) + FalsosNegativos(FN)} \tag{3.3}$$

d) **F1:**

$$F1 = \frac{2 * Prec * Recall}{Preciso + Recall} \tag{3.4}$$

Para realizar a avaliação da evolução do aprendizado das CNNs, foi utilizado o método *Cross-Validation* com 10 partições. Os valores das métricas de avaliação da VGG16 - Modificada e da VGG16+ para as imagens de treinamento são apresentados na Tabela 3.

A CNN que apresentou os melhores resultados na identificação das expressões faciais foi a CNN VGG16 - Modificada, apresentando uma acurácia de 0,975 para as imagens de treino. Já a VGG16+ apresentou uma acurácia de 0,884. As matrizes de confusão para as imagens de treino tanto da CNN VGG16 - Modificada como da CNN VGG16+ podem ser vistas nas Figuras 44 e 43 respectivamente.

Tabela 3 – Avaliação das CNNs para as imagens de treino

	VGG16 - Modificada			**VGG16+**		
	Precisão	**Recall**	**F1-Score**	**Precisão**	**Recall**	**F1-Score**
Alegria	0.97	1.00	0.99	0.89	0.93	0.91
Medo	0.92	0.97	0.95	0.78	0.88	0.83
Nojo	1.00	1.00	1.00	1.00	0.95	0.97
Raiva	0.99	1.00	1.00	0.90	0.78	0.93
Surpresa	1.00	1.00	1.00	1.00	0.88	0.94
Tristeza	0.97	0.97	0.97	0.91	0.96	0.94
Neutra	1.00	0.91	0.95	0.83	0.89	0.86

Fonte: (Autor, 2020)

Figura 43 – Matriz de confusão das imagens de treino para CNN VGG16 - Modificada

Fonte: (Autor, 2020)

Figura 44 – Matriz de confusão das as imagens de treino para a CNN VGG16+

Fonte: (Autor, 2020)

Ao analisar ambas as matrizes de confusão pode-se verificar que na CNN VGG16 - modificada todas as emoções tiveram uma boa taxa de acerto, sendo apenas a emoção neutra a que obteve os menores resultados, com uma taxa de 91% para as imagens que pertenciam a esse

grupo. Já ao observar a matriz de confusão da CNN VGG16+ pode-se verificar que os resultados obtidos foram inferiores à CNN anterior, sendo a emoção de raiva, que obteve uma taxa de acerto de 78% a mais baixa entre as emoções. As emoções de nojo e tristeza foram as que tiveram as melhores taxas de acerto sendo elas acima de 95%.

3.8 AQUISIÇÃO DAS IMAGENS E TESTES COM OS VOLUNTÁRIOS

Para a etapa de aquisição das imagens pode ser utilizado um microcomputador Raspberry acoplado a uma câmera. O dispositivo então pode ser instalado em uma sala de aula e irá realizar o monitoramento e a captura das expressões faciais dos estudantes durante atividade, armazenado as imagens para mais tarde então serem analisadas pelo *software*.

Por ser um dispositivo relativamente pequeno, ele acaba por não chamar muita atenção, assim não interferindo na concentração dos estudantes durante as aulas, o que permite que elas aconteçam de forma natural.

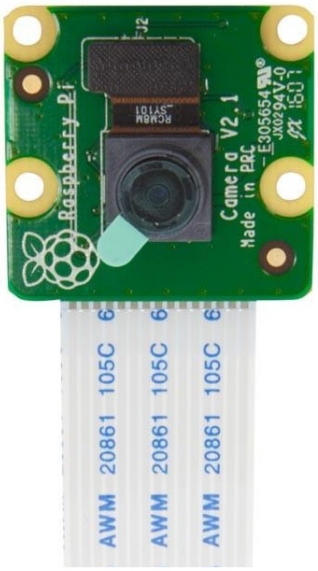
O dispositivo de processamento para aquisição de imagens que pode ser utilizado é uma Raspberry Pi 3 Modelo B+ (RASPBERRY-STORE, 2019a), um computador de baixo custo e tamanho reduzido, demonstrada na Figura 45. A Raspberry Pi 3 Modelo B+ possui 4 portas USB 2.0, conexão com WiFi e *Bluetooth*, alimentação de 5V, além de uma entrada para cartão Micro SD que permite o armazenamento de dados e uma conexão especifica para uma Câmera Module V2 (RASPBERRY-STORE, 2019b).

Figura 45 – Raspberry Pi 3 Modelo B+

Fonte: (RASPBERRY-STORE, 2019a)

A Câmera Module V2 demonstrada na Figura 46, possui 8 MP de resolução e pode ser utilizada para gravar vídeos em alta resolução e tirar fotos, podendo também ser acoplada em diferentes modelos de dispositivos.

Figura 46 – Câmera Raspberry Pi Module V2



Fonte: (RASPBERRY-STORE, 2019b)

A ideia inicial deste projeto era realizar experimentos em sala de aula e coletar imagens reais de estudantes durante os períodos de atividades na UCS. Porém, devido aos impedimentos causados pelo isolamento social e cancelamento das atividades presenciais na UCS, não foi possível prosseguir com a ideia original.

Para poder realizar os experimentos de forma adequada, mesmo fora de um ambiente de ensino real, foi realizada a coleta das imagens de maneira a simular atividades de ensino em sala de aula. As imagens foram coletadas com uma câmera fotográfica digital, sendo as fotos armazenadas em um cartão micro SD e posteriormente analisadas pelo *software* proposto.

Foram selecionados alguns voluntários e para cada um deles foram feitas sequências de imagens de cada uma das expressão desejadas, tais como: alegria, surpresa, tristeza, nojo, raiva, medo e neutra. Após serem coletadas, as imagens foram então submetidas a um pré-processamento, onde foram selecionadas pelo autor dez imagens de cada uma das expressões. Estas imagens correspondem as que melhor identificam emoções nas expressões faciais do sujeito fotografado.

Após a seleção, foram extraídos os rostos de cada uma das imagens para passar pelo *software* de modo a avaliar as expressões faciais e ver se ele era capaz de reconhecê-las e identificar de maneira correta, avaliando o nível de acerto. A Figura 47 apresenta um exemplo

das imagens que foram capturadas dos voluntários para testes de controle.

Figura 47 – Sequencia de imagens de um voluntário

Fonte: (Autor, 2020)

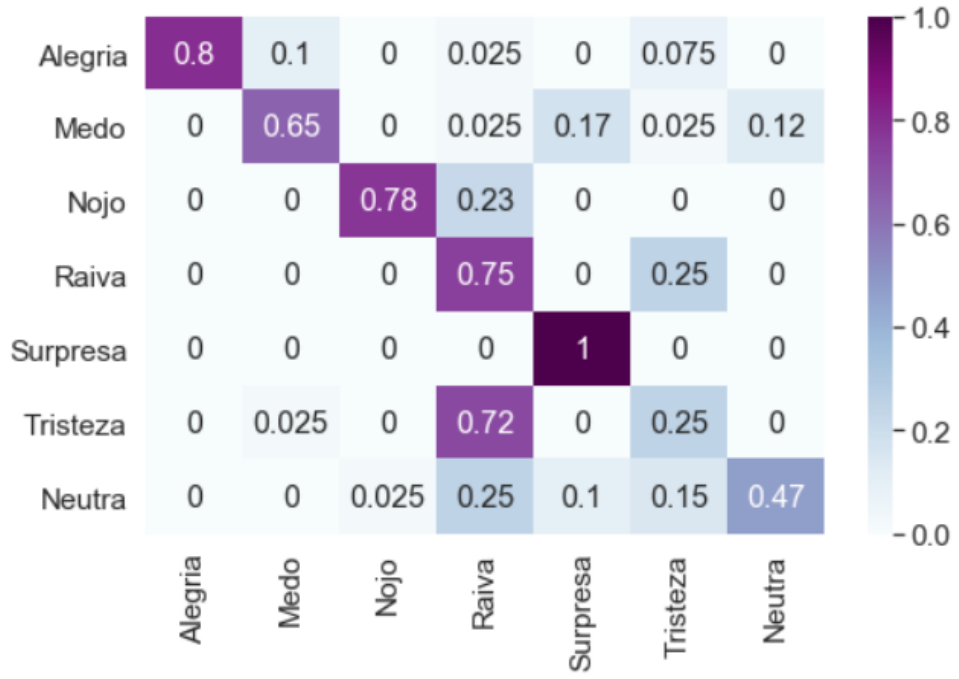
Para a avaliação foram utilizadas dez imagens de cada uma das sete emoções de quatro participantes. As 280 imagens foram usadas para teste na CNN VGG16 - Modificada (*dataset* de teste). Destaca-se que a CNN não foi treinada para reconhecer tais imagens. A acurácia obtida para este *dataset* de teste é 0.6714, sendo que a Tabela 4 mostra os valores de precisão, recall e F1-score obtidos na avaliação dessas imagens. Já a Figura 48 apresenta a matriz de confusão gerada nessa avaliação.

Tabela 4 – Avaliação da CNN VGG16 - Modificada com *dataset* de teste

	Precisão	**Recall**	**F1-Score**
Alegria	1.00	0.80	0.89
Medo	0.84	0.65	0.73
Nojo	0.97	0.78	0.86
Raiva	0.38	0.75	0.50
Surpresa	0.78	1.00	0.88
Tristeza	0.33	0.25	0.29
Neutra	0.79	0.47	0.59

Fonte: (Autor, 2020)

Figura 48 – Matriz de confusão da CNN VGG16 - Modificada com *dataset* de teste



Fonte: (Autor, 2020)

A partir da análise da tabela de avaliação e da matriz de confusão, pode-se perceber que para esse *dataset* de teste a CNN encontrou uma maior dificuldade em reconhecer corretamente algumas das emoções com tristeza e neutra, tendo uma taxa de acerto de 25% para as imagens que continham expressões de tristeza e 47% para as imagens que continham expressões neutras. As imagens que continham expressões de tristeza e neutra tiveram uma tendência maior de serem classificadas incorretamente como imagens de raiva. Somente a emoção de surpresa obteve uma taxa de acerto de 100% para as imagens que realmente possuíam expressões faciais de surpresa.

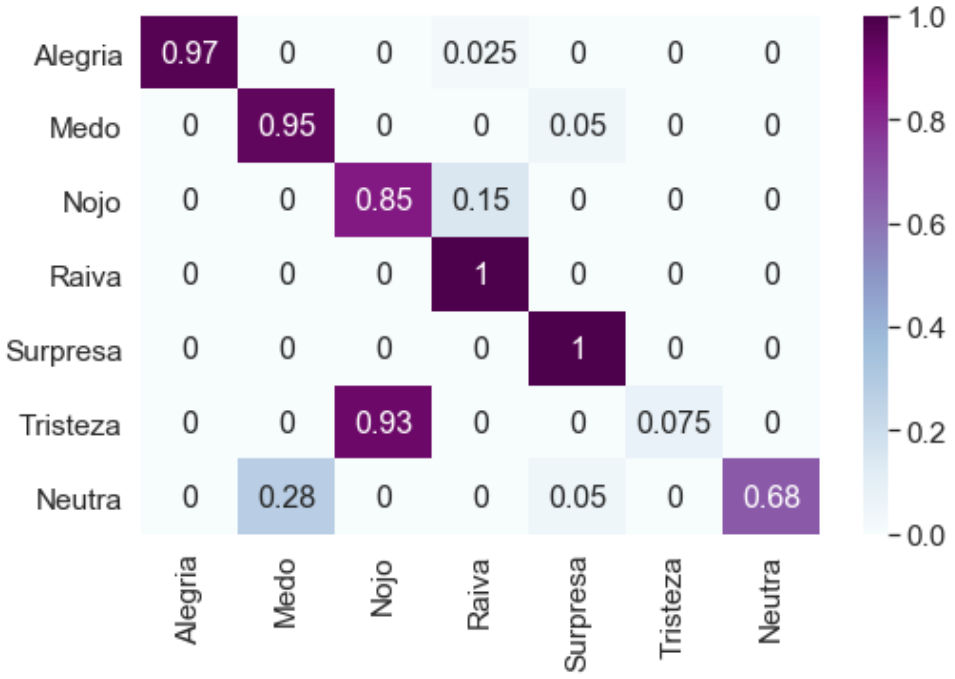
Como algumas das emoções apresentaram uma taxa de acerto mais baixa foi realizado uma nova etapa de treinamento da CNN incluindo o *dataset* de teste ao *dataset* de treino. Esse treinamento complementar da CNN auxilia na identificação das expressões dos participantes, melhorando assim o desempenho do classificador. A acurácia da CNN para o *dataset* de teste com mais essa etapa de treinamento complementar foi de 0.7892, apresentando assim uma maior taxa de acerto, comparada com o resultado sem o treinamento complementar. Os resultados gerados na nova análise para precisão, recall e F1-score podem ser vistos na Tabela 5, já a matriz de confusão pode ser vista na Figura 49.

Tabela 5 – Avaliação da VGG16 - Modificada com mais treinamento

	Precisão	**Recall**	**F1-Score**
Alegria	1.00	0.97	0.99
Medo	0.78	0.95	0.85
Nojo	0.48	0.85	0.61
Raiva	0.85	1.00	0.92
Surpresa	0.91	1.00	0.95
Tristeza	1.00	0.07	0.14
Neutra	1.00	0.68	0.81

Fonte: (Autor, 2020)

Figura 49 – Matriz de confusão da VGG16 - Modificada com mais treinamento



Fonte: (Autor, 2020)

A partir da análise da tabela de avaliação e da matriz de confusão, após a nova etapa de treinamento com as imagens dos participantes, pode-se verificar que houve um aumento na correta classificação das expressões faciais. Com este novo treinamento, a maioria das expressões obteve um aumento na correta classificação. Como exceção, a expressão de tristeza sofre aumento na classificação incorreta. Observou-se que a CNN passou a classificar expressões de tristeza como sendo expressões de nojo, e não mais como expressões de raiva, como havia sido observado previamente.

Dando sequência na etapa de avaliação, os voluntários foram colocados dispostos lado a lado de maneira com que todos estivessem com o rosto visível, sem nenhuma interferência externa. Assim, procurou-se simular uma sala de aula real com estudantes assistindo a uma aula. Foi pedido para que eles assistissem a um vídeo e que demonstrassem diferentes expressões durante a gravação para que suas expressões pudessem serem analisadas posteriormente. A figura 50 apresenta uma das imagens que foram capturadas e analisadas.

Figura 50 – Simulação de uma sala de aula

Fonte: (Autor, 2020)

Após a captura do vídeo foi realizado um processo de seleção de um *frame* a cada 5 segundos, sendo analisados 50 *frames* ao total. O grupo de imagens capturadas compôs o conjunto de dados de teste. Cada imagem foi classificada pelo *software* VGG16 - Modificado, para extrair a emoção captada. Cada imagem foi também avaliada de maneira manual, para fins de comparação. As emoções analisadas pelo *software* foram classificadas conforme apresentado na tabela 6, e para cada um dos participantes são apresentadas a quantidade total de vezes que cada uma das emoções apareceram nas imagens que foram extraídas do video e analisadas.

Ao analisar os dados nota-se que em um ambiente de ensino, a partir de algumas modificações e melhorias no *software* pode ser possível realizar a identificação das expressões faciais de estudantes em sala de aula. Não obstante, a classificação de emoção de tristeza, que obteve menores valores de acurácia, precisa ser aprimorada neste modelo.

Tabela 6 – Avaliação da simulação

	Pessoa 1	**Pessoa 2**	**Pessoa 3**
Alegria	5	8	5
Medo	5	6	0
Nojo	11	13	21
Raiva	11	20	16
Surpresa	3	0	0
Tristeza	6	2	8
Neutra	9	1	0

Fonte: (Autor, 2020)

3.9 CONSIDERAÇÕES SOBRE O CAPÍTULO

Neste capítulo foi apresentada a construção, treinamento e avaliação de uma CNN com o objetivo de reconhecer expressões faciais. Também foi demonstrada viabilidade de uso de uma CNN para implementar um sistema em sala de aula que reconheça emoções por meio das expressões faciais dos estudantes. Assim, espera-se contribuir para auxiliar atividades de ensino e ajudar o professor a entender como está o andamento de suas aulas e como anda a motivação e o interesse de seus alunos. A partir dos resultados fornecidos pelo *software* o professor pode então tomar as medidas que achar necessárias.

Também foi apresentado durante este capítulo o conceito de como pode ser feita a implementação na sala de aula. Ao coletar as imagens dos estudantes e realizar um treinamento adicional da CNN com elas, isso faz com que a CNN aprenda a reconhecer melhor os rostos dos estudantes da sala fornecendo resultados mais precisos.

Como o objetivo deste trabalho é reconhecer as expressões bem definidas, esta CNN possui limitações ao reconhecer expressões que estão entre os limites das apresentadas. Para o sistema poder ser utilizado em um ambiente real seria necessário aumentar o treinamento da CNN com imagens que possuem expressões de transição entre as expressões bases.

Os resultados obtidos com as avaliações de ambas as CNNs, tanto para o *dataset* de treino como para o *dataset* de teste demonstram um bom potencial de reconhecimento das emoções através das expressões faciais. Quanto maior for o treinamento e mais imagens forem utilizadas, melhores serão os resultados obtido e menor será a taxa de erro.

4 CONCLUSÃO

A presente pesquisa buscou abordar a construção de uma sala de aula inteligente. Ao analisar os requisitos necessários para o desenvolvimento de um *software* capaz de reconhecer as emoções de estudantes através da análise das expressões faciais dos mesmos, foram abordados temas como visão computacional, construção de redes neurais e *IoT*.

Durante a pesquisa também foi abordado a maneira com que as emoções dos estudantes influenciam no seu aprendizado, e a forma como elas tendem a ter interferência sobre toda a turma. Dentro deste contexto, ao unir as diferentes áreas do conhecimento, mostrou-se possível a construção e o desenvolvimento de *software* que possa analisar as emoções e o nível de interesse dos estudantes, de maneira a proporcionar ao professor uma forma automática de reconhecimento, para poder avaliar a motivação e o interesse de seus estudantes durante as aulas, assim auxiliando na construção de uma sala de aula inteligente.

No presente estudo foi proposto o desenvolvimento e treinamento de dois modelos de CNN para reconhecimento de emoções através das expressões faciais, sendo o treinamento realizado com 100 imagens para cada uma das expressões. O primeiro modelo implementado a partir de uma CNN já treinada para o reconhecimento de diversos outros tipos de imagens, chamada VGG16, foi o que obteve os melhores resultados durante os testes. O segundo modelo proposto, foi a criação e o treinamento de uma CNN a partir do zero. Também foi proposto uma forma de análise das emoções de maneira automática dos estudantes em sala de aula durante o ensino.

4.1 CONTRIBUIÇÕES DO TRABALHO

Com o desenvolvimento deste trabalho foi possível analisar a influência que a emoção de um estudante tem em seu aprendizado e no aprendizado da turma como um todo. Ao entender isso, é possível avaliar diferentes maneiras de conseguir identificar as emoções dos estudantes, sua motivação e o seu interesse durante as aulas.

Ao realizar esta analise é possível sugerir diferentes maneiras no sentido de realizar a identificação destes componentes de forma automática, fazendo com que o professor possua uma maior compreensão de como esta o andamento de sua turma e tomar as medidas que julgar necessárias para aumentar o desenvolvimento de toda a turma.

Assim este projeto apresentou uma implementação inicial de um sistema para identificação de emoções através das expressões faciais dos estudantes, desta forma adicionando inteligencia a uma sala de aula.

4.2 TRABALHOS FUTUROS

Existem algumas alternativas para dar continuidade ao desenvolvimento de soluções de reconhecimento de emoções através de expressões faciais, colaborando assim com a construção de uma sala de aula inteligente, desta forma como trabalhos futuros sugere-se:

a) Treinamento com um maior numero de imagens para melhorar o nível de acerto da CNN, e o treinamento com imagens de transição entre as expressões faciais.

b) Implementação de um sistema para coleta de imagens automático em um ambiente de sala de aula real.

c) Testes com outras CNNs para avaliar a performance e o nível de acerto delas.

REFERÊNCIAS

ABADI, M. *et al.* Tensorflow: Large-scale machine learning on heterogeneous distributed systems. **CoRR**, abs/1603.04467, 2016.

AMBRÓSIO, P. E. **Redes neurais artificiais no apoio ao diagnóstico diferencial de lesões intersticiais pulmonares**. Dissertação (Mestrado) — Faculdade de Filosofia Ciências e Letras de Ribeirão Preto, Universidade de São Paulo, Ribeirão Preto, 2002.

ANI, R. *et al.* An approach towards building an iot based smart classroom. In: **2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [S.l.: s.n.], 2018. p. 2098–2102.

ARBIB, M. A. (Ed.). **The Handbook of Brain Theory and Neural Networks**. 2nd. ed. Cambridge, MA, USA: MIT Press, 2002. ISBN 0262011972.

AUGUSTO, J. C. Ambient intelligence: The confluence of ubiquitous/pervasive computing and artificial intelligence. In: ____. **Intelligent Computing Everywhere**. London: Springer London, 2007. p. 213–234. ISBN 978-1-84628-943-9.

COWEN, A. S.; KELTNER, D. Self-report captures 27 distinct categories of emotion bridged by continuous gradients. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, 2017. ISSN 0027-8424.

DAMÁSIO, A. **O mistério da consciência: Do corpo e das emoções ao conhecimento de si**. São Paulo: Companhia das Letras, 2000.

DINESH, D.; S, A. N.; BIJLANI, K. Student analytics for productive teaching/learning. In: **2016 International Conference on Information Science (ICIS)**. [S.l.: s.n.], 2016. p. 97–102.

ESQUEF, I. A. **Técnicas de Entropia em Processamento de Imagens**. Dissertação (Mestrado) — Centro Brasileiro de Pesquisas Físicas, Rio de Janeiro, dec 2002.

FISCHER, I. A. *et al.* Improving efficiency and availability in smart classroom environments. In: **2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)**. [S.l.: s.n.], 2019. p. 52–56.

FLECK, L. *et al.* Redes neurais artificiais: princípios básicos. **Revista Eletrônica Científica Inovação e Tecnologia**, v. 7, n. 15, p. 47–57, 2016.

FOUNDATION, O. **Open Source Computer Vision Library**. 2019. Disponível em: https://opencv.org. Acesso em: 01 out. 2019.

FURTADO, M. I. V. **Redes neurais artificiais: uma abordagem para sala de aula**. Ponta Grossa, PR: Atena Editora, 2019.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais**. [S.l.]: Edgard Blucher, 2000.

GORDIN, S.; LOIOLA, E. **Emoções, aprendizagem e comportamento social: conhecendo para melhor educar nos contextos escolares e de trabalho**. São Paulo: Casapsi Livraria e Editora Ltda, 2015.

GUPTA, S. K.; ASHWIN, T. S.; GUDDETI, R. M. R. Students' affective content analysis in smart classroom environment using deep learning techniques. **Multimedia Tools and Applications**, Springer Science and Business Media LLC, v. 78, n. 18, p. 25321–25348, maio 2019.

HARA, K.; SAITO, D.; SHOUNO, H. Analysis of function of rectified linear unit used in deep learning. In: **2015 International Joint Conference on Neural Networks (IJCNN)**. [S.l.: s.n.], 2015. p. 1–8.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. [S.l.]: Artmed, 2007. ISBN 9788577800865.

HE, K. *et al.* Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016. p. 770–778.

HEGHEDUS, C.; CHAKRAVORTY, A.; RONG, C. Neural network frameworks. comparison on public transportation prediction. In: **2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)**. [S.l.: s.n.], 2019. p. 842–849.

HERRERA, T.; NÚNEZ, F. An iot-ready streaming manager device for classroom environments in a smart campus. In: **2018 IEEE International Conference on Consumer Electronics (ICCE)**. [S.l.: s.n.], 2018. p. 1–5.

KANADE, T.; COHN, J. F.; TIAN, Y. Comprehensive database for facial expression analysis. In: **Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)**. [S.l.: s.n.], 2000. p. 46–53.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1**. USA: Curran Associates Inc., 2012. (NIPS'12), p. 1097–1105.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, maio 2015.

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

LEMLEY, J.; BAZRAFKAN, S.; CORCORAN, P. Deep learning for consumer devices and services: Pushing the limits for machine learning, artificial intelligence, and computer vision. **IEEE Consumer Electronics Magazine**, v. 6, n. 2, p. 48–56, April 2017.

LI, X.; SHI, Y. Computer vision imaging based on artificial intelligence. In: **2018 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)**. [S.l.: s.n.], 2018. p. 22–25.

LIENHART, R.; MAYDT, J. An extended set of haar-like features for rapid object detection. In: **Proceedings. International Conference on Image Processing**. [S.l.: s.n.], 2002. v. 1, p. I–I.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: **in ICML Workshop on Deep Learning for Audio, Speech and Language Processing**. [S.l.: s.n.], 2013.

NAKASHIMA, H.; AGHAJAN, H.; AUGUSTO, J. C. (Ed.). **Handbook of Ambient Intelligence and Smart Environments**. [S.l.]: Springer US, 2010.

NEVES, L. A. P.; NETO, H. V.; GONZAGA, A. (Ed.). **Avanços em Visão Computacional**. 1. ed. Curitiba, PR: Omnipax, 2012. 406 p. ISBN 978-85-64619-09-8.

NGUYEN, G. *et al.* Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. **Artificial Intelligence Review**, v. 52, n. 1, p. 77–124, Jun 2019. ISSN 1573-7462.

PACHECO, A. G. C. Classificação de espécies de peixe utilizando redes neurais convolucional. **CoRR**, abs/1905.03642, 2019.

PALMIERE, S. E. **Arquiteturas e Topologias de Redes Neurais Artificiais**. 2016. Disponível em: https://www.embarcados.com.br/redes-neurais-artificiais. Acesso em: 01 out. 2019.

PAN, S. J.; YANG, Q. A survey on transfer learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, Oct 2010.

RAMSUNDAR, B.; ZADEH, R. B. **TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning**. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2018. ISBN 1491980451, 9781491980453.

RASPBERRY-STORE. **Raspberry Pi 3 Model B+**. 2019. Disponível em: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus. Acesso em: 10 nov. 2019.

____. **Raspberry Pi Camera Module V2**. 2019. Disponível em: https://www.raspberrypi.org/products/camera-module-v2. Acesso em: 10 nov. 2019.

RAUTARAY, S.; AGRAWAL, A. Vision based hand gesture recognition for human computer interaction: A survey. **Artificial Intelligence Review**, v. 43, 01 2012.

RODRIGUES, V. **Machine Learning: o que são Acurracy, Precision, Recall e F1 Score**. 2019. Disponível em: https://medium.com/@vilsonrodrigues/machine-learning-o-que-sC3A3o-acurracy-precision-recall-e-f1-score-f16762f165b0. Acesso em: 02 mar. 2020.

ROSEBROCK, A. **ImageNet: VGGNet, ResNet, Inception, and Xception with Keras**. 2017. Disponível em: https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras. Acesso em: 28 mar. 2020.

SCIKITLEARN. **Cross-validation: evaluating estimator performance**. 2020. Disponível em: https://scikit-learn.org/stable/modules/cross-validation.html. Acesso em: 02 mar. 2020.

SILVA, C. C. D. P. **Como melhorar Expressão Facial e Corporal**. 2016. Disponível em: http://tertuliasdelibras.blogspot.com/2016/01/como-melhorar-expressao-facial-e.html - acesso em: 02 nov. 2019.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv 1409.1556**, 09 2014.

____. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. 2014.

SINGH, G.; CHITRANSH, A.; TANWAR, G. Monitoring ambient light conditions of a school using iot. In: **2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)**. [S.l.: s.n.], 2016. p. 3446–3449.

SR, S.; LJ, S. Monitor student's presence in classroom. **Journal of Information Technology & Software Engineering**, v. 6, 01 2016.

SZEGEDY, C. *et al.* Going deeper with convolutions. In: **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. p. 1–9.

TENSORFLOW. **Site Oficial**. 2019. Disponível em: https://www.tensorflow.org. Acesso em: 01 out. 2019.

TRISTÃO, I. **Sinapse - O que é, tipos, como e por que acontece?** 2019. Disponível em: https://conhecimentocientifico.r7.com/sinapse. Acesso em: 26 out. 2019.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: ALIAGA, D. G. *et al.* (Ed.). **Proceedings...** Porto Alegre: Sociedade Brasileira de Computação, 2016.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: **Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001**. [S.l.: s.n.], 2001. v. 1, p. I–I.

WANG, F. *et al.* Additive margin softmax for face verification. **IEEE Signal Processing Letters**, v. 25, n. 7, p. 926–930, July 2018.

WHITEHIL, J. *et al.* The faces of engagement: Automatic recognition of student engagementfrom facial expressions. **IEEE Transactions on Affective Computing**, v. 5, n. 1, p. 86–98, Jan 2014.

ZALETELJ, J.; KOŠIR, A. Predicting students' attention in the classroom from kinect facial and body features. **EURASIP Journal on Image and Video Processing**, Springer Nature, v. 2017, n. 1, dez. 2017.

ZEILER, M.; FERGUS, R. Visualizing and understanding convolutional neural networks. In: . [S.l.: s.n.], 2013. v. 8689.