UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

DEEP-EMOTIVE: PROTÓTIPO DE SISTEMA PARA RECONHECIMENTO DE EXPRESSÕES FACIAIS UTILIZANDO APRENDIZADO PROFUNDO

DIÓGENES ADEMIR DOMINGOS

BLUMENAU
2017

DIÓGENES ADEMIR DOMINGOS

DEEP-EMOTIVE: PROTÓTIPO DE SISTEMA PARA RECONHECIMENTO DE EXPRESSÕES FACIAIS UTILIZANDO APRENDIZADO PROFUNDO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof(a). Andreza Sartori, Doutora - Orientadora

BLUMENAU
2017

DEEP-EMOTIVE: PROTÓTIPO DE SISTEMA PARA RECONHECIMENTO DE EXPRESSÕES FACIAIS UTILIZANDO APRENDIZADO PROFUNDO

Por

DIÓGENES ADEMIR DOMINGOS

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof(a). Andreza Sartori, Doutora – Orientador, FURB
Membro:	Prof(a). Roberto Heinzle, Doutor – FURB
Membro:	Prof(a). Aurélio Faustino Hoppe, Mestre – FURB

Blumenau, 27 de outubro de 2017

Dedico este trabalho aos meus pais, que sempre lutaram para me fornecer uma educação de qualidade.

AGRADECIMENTOS

A Deus, que traçou minha caminhada para alcançar este momento.

À minha mãe Tania Barbosa, pelo apoio e incentivo nas madrugadas de estudo.

Ao meu pai Orival Marcelino Domingos, pela criação com princípios.

À minha orientadora Andreza Sartori, pela dedicação e motivação para a conclusão deste trabalho.

A todos os professores, por moldarem minha formação. Foram e serão fonte de inspiração.

It matters not how straight the gate. How charged with punishments the scroll. I am the master of my fate: I am the captain of my soul.

William Ernest Henley

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo para reconhecimento de emoções através das seis expressões faciais universais de Ekman et al. (1987), utilizando a técnica de Aprendizado Profundo (Deep Learning). A extração das características faciais foi realizada através da aplicação Aprendizado Profundo com redes neurais convolucionais, técnica considerada estado da arte na área de visão computacional para o reconhecimento de objetos em imagens. Para o desenvolvimento deste trabalho foram utilizadas técnicas de Processamento Digital de Imagens, além do uso do *framework* Keras para criação das redes neurais profundas. Para treinamento e testes do protótipo foi utilizado a base de dados Cohn-Kanade AU-Coded Expression Database Version 2 (CK+). O protótipo desenvolvido foi capaz de reconhecer oito emoções: alegria, tristeza, medo, surpresa, desgosto, raiva, desprezo e neutra, através das características das expressões faciais, com precisão de 96,33%, demonstrando a viabilidade das técnicas empregadas. Além disso, o algoritmo desenvolvido foi validado através da técnica de transferência de aprendizado, na qual foi testado em duas novas bases de dados: Japanese Female Facial Expression (JAFFE), obtendo a precisão de 93,02%, e Facial Expression Recognition 2013 (FER-2013), com precisão de 60,62%. Este trabalho pode servir como referência para a comunidade científica, auxiliando na geração de novos algoritmos computacionais, baseado nesta arquitetura, para o reconhecimento de emoções através de expressões faciais.

Palavras-chave: Visão computacional. Processamento digital de imagens. Redes neurais artificiais. Aprendizado profundo. Redes neurais convolucionais. Reconhecimento de emoções. Reconhecimento de expressões faciais. Transferência de aprendizado.

ABSTRACT

This work presents the development of a prototype for emotion recognition of the Ekman et al. (1987) universal facial expressions using deep learning. It was applied convolutional neural networks to extract the facial features, which is the state of the art technique in computer vision area, used on the object recognition of images. To the development of this work it was used Digital Image Processing techniques jointly with the Keras framework, to build the deep neural networks. The Cohn-Kanade AU-Coded Expression Database Version 2 (CK +) was applied to train and test the prototype. The proposed work achieved all the pre-established objectives, being able to recognize eight emotions: joy, sadness, fear, surprise, disgust, anger, contempt and neutral through the extracted features of facial expressions. The prototype reached a 96.33 % of accuracy, demonstrating the feasibility of the techniques used. In addition, the transfer learning technique validated the algorithm developed in which it was tested in two new databases: Japanese Female Facial Expression (JAFFE), obtaining a precision of 93.02%, and Facial Expression Recognition 2013 (FER-2013), with an accuracy of 60.62%. The development of this prototype proved to be relevant on the support of new computational algorithms based on this architecture to the emotion recognition of facial expressions.

Keywords: Computer vision. Digital image processing. Artificial neural networks. Deep learning. Convolutional neural networks. Emotion recognition. Facial expressions recognition. Transfer learning.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Atualmente o volume de dados que trafega na rede mundial de computadores cresce expressivamente. Segundo pesquisa realizada pela Cisco Systems (2016), o tráfico de dados na internet no ano de 2016 obteve um crescimento médio global de 32%, atingindo 73,1 Exabytes (EB) de dados trafegados por mês. Com o desenvolvimento do campo de inteligência artificial, a evolução constante dos dispositivos tecnológicos, e a transição para computação ubíqua, é possível extrair conhecimento dessa massa de dados e utilizá-lo como entrada de parâmetros para a análise do relacionamento de pessoas com os dispositivos tecnológicos.

De fato, a área de Interação Homem-Computador (IHC) busca evoluir a interface de relacionamento das pessoas com os dispositivos tecnológicos a fim de melhorar as interações, tornando-as mais claras e ágeis (BOOTH, 1989). A interação com dispositivos tecnológicos se dá através de periféricos, na qual atualmente é possível utilizar como parâmetro de entrada os dados obtidos pelo mouse, teclado, microfone, sensor de toque, câmera, entre outros. Com a computação ubíqua e as técnicas de Visão Computacional se parte para uma nova fase desta interação, a qual é possível interpretar dados impalpáveis como a emoção humana, extraída de uma ou mais imagens obtidas por uma câmera, e utilizá-las como parâmetro para uma tomada de decisão.

Os estudos que utilizam emoções expressas por usuários como parâmetros de entrada fazem parte da área da Computação Afetiva. Esta considera que os dispositivos tecnológicos, além de reconhecer emoções, devem ser dotados da capacidade de demonstrar afeto (PICARD, 1997). Este campo de estudo aborda, além da IHC, linhas de pesquisas em áreas como Visão Computacional, Inteligência Artificial, Psicologia e Neurociência.

O reconhecimento das emoções é realizado de várias formas, através da fala, de efeitos sonoros, cores e em todas elas, existe a interpretação de expressões faciais (EKMAN, 2003). A interpretação correta dos dados obtidos pelo processo de reconhecimento das seis expressões faciais definidas por Ekman et al. (1987) e reconhecidas universalmente, como alegria, tristeza, medo, surpresa, desgosto e desprezo, podem auxiliar na predição de uma melhor abordagem para realizar determinada ação, pois segundo Schultz et al. (2001) reações apropriadas e adaptativas em situações particulares dependem do conhecimento do estado emocional do outro.

Diante do exposto, este trabalho apresenta um protótipo para reconhecer e classificar as expressões faciais das emoções, alegria, desgosto, medo, raiva, surpresa, tristeza,

consideradas por Ekman et al. (1987), como básicas e universais, utilizando técnicas já consolidadas das áreas de Processamento Digital de Imagem e Visão Computacional. Estas técnicas são combinadas com uma nova abordagem, considerada a mais avançada no reconhecimento visual de objetos através do aprendizado de máquina, o Aprendizado Profundo ou Deep Learning.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um protótipo capaz de reconhecer automaticamente as emoções universais de Ekman através da identificação das expressões faciais.

Os objetivos específicos são:

a) reconhecer a face em imagens utilizando técnicas de visão computacional e processamento digital de imagem;
b) extrair as características faciais relevantes para identificar as expressões faciais utilizando técnicas de visão computacional e processamento digital de imagem;
c) implementar um classificador de emoções utilizando redes neurais artificiais com a tipologia de Aprendizado Profundo ou Deep Learning;
d) reconhecer as seis emoções consideradas básicas e universais.

1.2 ESTRUTURA

O trabalho proposto é composto por quatro capítulos. O primeiro destinado à apresentação da introdução e objetivos para o desenvolvimento do trabalho. O capítulo seguinte aborda a fundamentação teórica, necessária para o entendimento da técnicas e ferramentas utilizadas, explanando conceitos como as expressões faciais universais, computação afetiva, visão computacional e redes neurais. O terceiro capítulo descreve a arquitetura do protótipo através dos diagramas de caso de uso e de atividade. Este capítulo apresenta também as ferramentas utilizadas, a implementação e os resultados obtidos nos testes realizados. Por fim, o quarto e último capítulo se destina à exposição das conclusões e limitações do trabalho, bem como sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo introduz os assuntos e técnicas abordadas para desenvolvimento do trabalho proposto. A seção 2.1 aborda o conceito de emoção e as expressões faciais. A seção 2.2 introduz o conceito de computação afetiva e a seção 2.3 apresenta conceitos de visão computacional. Na seção 2.4 apresenta as principais características de redes neurais e a técnica do Aprendizado Profundo ou Deep Learning. Por fim, os trabalhos correlatos são apresentados na seção 2.5.

2.1 EMOÇÕES E AS EXPRESSÕES FACIAIS

As emoções fazem parte de um conjunto de mecanismos que o indivíduo possui visando a sobrevivência. Segundo Damásio (2000, p. 61), a emoção "[...] associa-se às reações fisiológicas coordenadas e em grande medida automáticas que são necessárias para manter estáveis os estados internos de um organismo vivo.". Ainda segundo o autor, elas podem ser definidas como padrões formados através de reações químicas e neurais na qual desempenham um papel específico para o indivíduo e tem como objetivo auxiliar na conservação da vida.

As emoções desempenham um papel fundamental em nossas vidas, seja nos relacionamentos íntimos, nas interações sociais e familiares, bem como auxiliando na adaptação a inúmeras ocasiões (REEVE, 2006). Segundo Schultz et al. (2001), para que uma reação seja apropriada e adaptativa à uma determinada situação, ela depende do conhecimento do estado emocional do outro.

Na psicologia, as emoções são divididas em quatro modelos (BOROD, 2000): o modelo dimensional que classifica as emoções em duas categorias arousal (calmo/excitado) e valência (positiva/negativa), distinguindo cada emoção pela sua característica (PICARD, 1997); o modelo discreto, também conhecido como modelo de emoções básicas, que são alegria, tristeza, medo, surpresa, desgosto e raiva (EKMAN et al., 1987); o modelo baseado em significados, que visa definir a semântica da emoção expressa (BOROD, 2000); e o modelo baseado em componentes, que busca definir como as emoções evoluem hierarquicamente (BOROD, 2000).

Em 1965, Darwin realizou um trabalho para estabelecer uma detalhada descrição anatômica da expressão facial humana, comparando as reações entre diversas espécies (DARWIN; LORENZ, 2000). Este estudo serviu como base para trabalhos como de Ekman et al. (1987), que concluiu que as emoções do modelo discreto são reconhecidas universalmente através de expressões faciais. Ekman et al. (1987) considerou este conjunto de emoções como

universal por apresentarem as mesmas características em indivíduos de cultura, raça, gênero e idade diferentes.

As expressões faciais universais de Ekman são apresentadas na Figura 1. A primeira emoção é a de surpresa, caracterizada pelas sobrancelhas levantadas, boca aberta ou semiaberta, podendo apresentar rugas horizontais na testa e olhos arregalados. A segunda emoção é de tristeza, apresentando um declínio nos cantos da boca, projeção dos lábios. A terceira emoção é o desgosto, apresentando um franzir entre os olhos e o nariz. A quarta emoção é raiva, na qual ocorre a aproximação das sobrancelhas, lábios comprimidos. A quinta é a alegria, representada pelo sorriso, os lábios são deslocados para traz e para cima, pálpebras elevadas, e sem ocorrência de rugas na testa. A última emoção é o medo, olhos bem abertos, os cantos dos lábios estão declinados e sobrancelhas levemente erguidas.

Figura 1 - Expressões das seis emoções universais

Fonte: CK+ (LUCEY et al. 2010).

Ekman e Friesen (1976) desenvolveram o Sistema de Codificação da Ação Facial (Facial Action Coding System - FACS), que faz o reconhecimento das expressões faciais de cada emoção baseado nas alterações e intensidade dos movimentos dos músculos faciais. Cada alteração muscular foi mensurada e representada através de uma unidade de ação, denominada Action Units (AU). Esta classificação do movimento facial descreve a ação do estímulo muscular realizado pela combinação de dois ou mais músculos. Algumas expressões faciais são mensuradas através da combinação de duas ou mais unidades de ações. A Figura 2 ilustra algumas unidades de ação e combinações. A imagem ilustra uma face com emoção neutra, apresentando olhos, sobrancelhas e bochechas relaxados (AU0 - neutral). Em seguida, a AU1 é caracterizada pela parte interior das sobrancelhas estarem levantadas. A AU2, apresenta a parte externa da sobrancelha levantada, e, a AU5 mostra as pálpebras superiores levantadas. Observa-se que, a união entre a AU1 + AU2 + AU5 pode ser utilizada para definir

uma face com a emoção de surpresa, ou espanto. Contudo uma emoção é definida com a combinação de várias AU's.

Figura 2 - Unidades de ação do sistema FACS

AU 1	AU 2	AU 4
		
Inner portion of the brows is raised.	Outer portion of the brows is raised.	Brows lowered and drawn together
AU 5	AU 6	AU 7
		
Upper eyelids are raised.	Cheeks are raised.	Lower eyelids are raised.
AU 1+4	AU 4+5	AU 1+2
		
Medial portion of the brows is raised and pulled together.	Brows lowered and drawn together and upper eyelids are raised.	Inner and outer portions of the brows are raised.
AU 1+2+4	AU1+2+5+6+7	AU0(neutral)
		
Brows are pulled together and upward.	Brow, eyelids, and cheek are raised.	Eyes, brow, and cheek are relaxed.

Fonte: adaptado de Tian, Kanade e Cohn (2001, p. 3).

Com as medidas do sistema FACS inicia-se estudos de reconhecimento automatizado das emoções, como o trabalho de Bartlett et al. (2003), que tem como objetivo localizar automaticamente faces em um fluxo de vídeo e codificar a expressão visual dinamicamente. Segundo Sandbach et al. (2012), a utilização das unidades de ação do sistema FACS, agregado a algoritmos de visão computacional, torna possível o desenvolvimento de sistemas computacionais que reconheçam emoções através da análise das expressões faciais.

2.2 COMPUTAÇÃO AFETIVA

A Computação Afetiva (CA), subárea da Inteligência Artificial, é definida por Picard (1997) como a área que aborda a aplicabilidade da afeição em sistemas não-biológicos. Aplicações da CA são utilizadas em diversas áreas, como por exemplo, na área da segurança, tornando possível predizer se algum indivíduo manifesta emoção considerada agressiva a um determinado pronunciamento; na educação, predizendo quais alunos apresentam mais dificuldade no aprendizado; e na saúde, identificando os sinais não verbais como as emoções das expressões faciais dos pacientes, que podem proporcionar um aconselhamento eficaz e oferecer um melhor tratamento psicoterapêutico (FOLEY; GENTILE, 2010).

Segundo Picard (1997), a aplicação de um sistema CA pode ser confrontada pelo Teste de Turing (Turing, 1950). A premissa do teste consiste em um sistema que consiga interagir e ludibriar um terço dos seus locutores humanos de forma que o locutor humano não consiga distinguir se está interagindo com uma máquina ou outro humano (Turing, 1950). Picard (1997) afirma que não é possível uma máquina obter sucesso no teste de Turing sem que esta seja dotada da capacidade de demonstrar emoções.

Picard (1997) declara também que o estado emocional de uma pessoa não pode ser totalmente observado por outra, é possível apenas perceber a ação ocorrida com troca de experiências entre os indivíduos, também chamado de comportamentos observáveis. Quando um indivíduo é induzido a experimentar o estado emocional de outro, a comportamento produzido por ele, pode, ou não, demonstrar alguma emoção de forma natural, em relação a situação exposta pelo outro indivíduo. Ao contrário, se o indivíduo é explicitamente obrigado a demonstrar alguma emoção acerca da situação, este passará a fornecer respostas não naturais, fornecendo ações observáveis mais tensas. Ainda segundo a autora, os sistemas de computação afetiva se denotam do reconhecimento de padrões multimodais, e apresenta quatro abordagens que podem ser utilizadas para o reconhecimento de emoções: inflexão de voz; comportamentos observáveis; expressões faciais; e linguagem corporal.

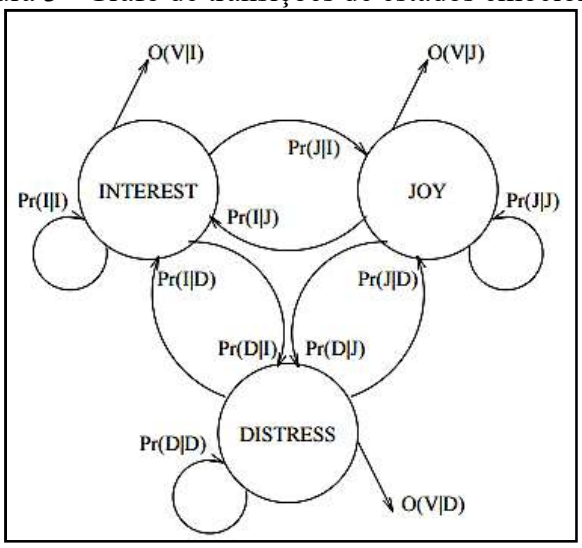
A primeira abordagem, inflexão de voz, é oriunda da prosódia, estudo das particularidades dos sons, como acentos, quantidade e entoação. A qualidade da modularização do som pode ser utilizada como parâmetros em um sistema dotado da capacidade de fala, demonstrando afeto (MURRAY; ARNOTT, 1993). Esta abordagem pode ser exemplificada na ação realizada por um animal de estimação em resposta a um comando vocal de seu dono, ou seja, o animal não tem a capacidade de compreender a palavras ditas, mas sim da forma como elas foram ditas.

A segunda abordagem dá-se através de comportamentos observáveis dos estados emocionais, que segundo Picard (1997), propõe medir através de cálculos probabilísticos, os comportamentos observáveis das expressões emocionais, deliberadas ou não, para inferir o estado emocional. Para tanto, é possível utilizar um processo estocástico, como Hidden Markov Models (HMM), uma evolução das Cadeias de Markov (Markov Chains). Jelinek (1997) diz que, as Cadeias de Markov, em sua representação regular, utiliza como parâmetro a probabilidade da transição de um estado para outro, a fim de determinar as variáveis ocultas do conjunto.

A Figura 3 ilustra o grafo os estados emocionais utilizando algumas emoções do modelo dimensional, representadas como: Interesse (I); Aflição (D) e Alegria (J), bem como a

probabilidades de características observáveis como a inflexão de voz (V). Além dos modelos de Markov, uma Rede Neural Artificial também pode ser treinada para representação dos estados emocionais. Por ser um problema de classificação supervisionado, existem outras técnicas de reconhecimento e aprendizagem disponíveis.

Figura 3 - Grafo de transições de estados emocionais



Fonte: Picard (1997, p. 7).

A terceira abordagem é o reconhecimento através de expressões faciais, que segundo Picard (1997), é a forma de reconhecimento emocional de maior amplitude, sendo que a maioria dos sistemas que utilizam esta abordagem, baseiam-se na aplicação das métricas do sistema FACS. Um modelo de representação gráfica pode ser utilizado para imitar a expressão reconhecida de forma a tornar a interação homem-máquina mais próxima da humana, como no trabalho realizado por Bartlett et al. (2003).

A quarta abordagem, linguagem corporal, é aquisição de características de sinais fisiológicos como Eletromiograma1, tensão muscular ou respiração, agregados a outra abordagem mais ampla, pois estas apresentam apenas pequenas evidências de excitação e valência (PICARD, 1997). O trabalho de Soleymani et al. (2016) demonstra a utilização de características da linguagem corporal, características estas obtidas através do Eletroencefalograma2 e associadas ao reconhecimento da expressão facial. Para identificar a emoção, os autores utilizaram técnicas de regressão e verificaram a correlação entre as características.

Picard (1997) conclui que, a computação afetiva pode gerar um ganho de informação valiosa. Observando o comportamento do usuário, extraindo e analisando as características da

[1] Eletromiograma é um exame de funcionamento do sistema nervoso e os músculos (BASMAJIAN, 1985).
[2] Eletroencefalograma exame que realiza o mapeamento do cérebro (BARLOW, 1983).

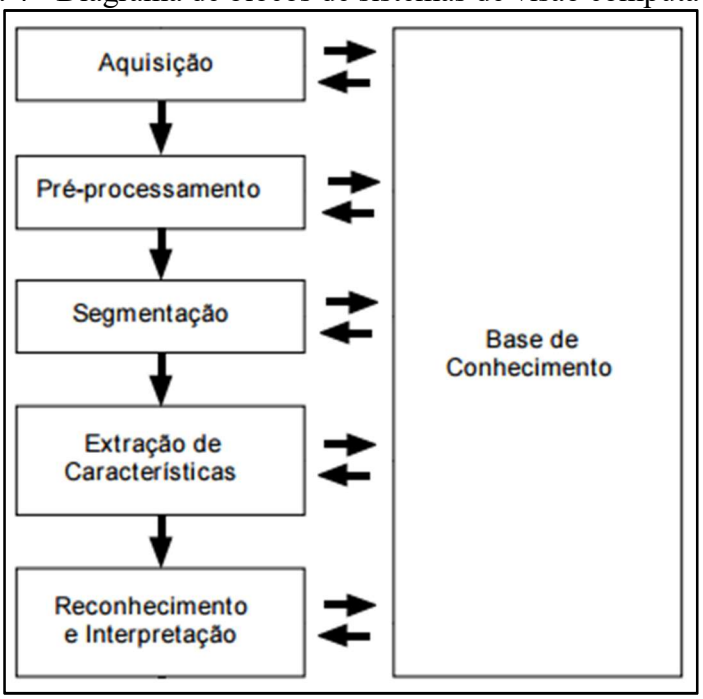
face, que atualmente pode ser adquirida através de computadores pessoais, câmeras de segurança, smartphones, entre outros dispositivos, combinado com outros dados, como local, alimentação, e, identificando a emoção a procura de novos padrões, pode fornecer dados valiosos para pesquisas que infiram esse conhecimento ao próprio usuário ou sirvam como entrada para outro sistema.

2.3 VISÃO COMPUTACIONAL

O sistema visual do ser humano é capaz de perceber a estrutura em 3D de qualquer objeto que o cerca, e reconhecer suas características como forma, iluminação e distribuição de cor, e ainda segmentar objetos na cena observada (SZELISKI, 2011). Segundo Szeliski (2011) a Visão Computacional tenta realizar o inverso do processo biológico e descrever o mundo que vemos através de uma ou mais imagens, transformando essas propriedades em dados que o computador consiga interpretar. Ainda segundo o autor, a visão computacional é a combinação das áreas de Processamento Digital de Imagem e Inteligência Artificial. Esta possui determinado grau de dificuldade, tentando realizar o reconhecimento de uma incógnita através de informações insuficientes.

Segundo Gonzalez e Woods (2010), o processo de análise de uma imagem por sistemas de visão computacional é dado através de etapas, as quais são processadas pelo sistema de forma sequencial, isto é, o resultado de uma etapa é utilizado como entrada para a etapa seguinte. As etapas do sistema de visão computacional mais relevantes para realização deste trabalho são: aquisição da imagem; pré-processamento; segmentação; extração de características; reconhecimento e interpretação. Estas etapas são ilustradas pela Figura 4. Existem outras etapas que compõem um sistema de visão computacional, como filtragem, restauração e compressão da imagem, porém não serão utilizadas no escopo deste trabalho. Este trabalho aborda apenas as etapas relevantes para o desenvolvimento do protótipo.

Figura 4 - Diagrama de blocos de sistemas de visão computacional



Fonte: adaptado de Marques Filho e Vieira Neto (1999, p. 9).

O diagrama mostra um processamento sequencial, não obstante, o processamento pode abster-se de uma ou mais etapas, e.g. o resultado da etapa de aquisição pode seguir direto para etapa de segmentação, tudo depende da complexidade do problema a ser resolvido. Observa-se que todas as etapas possuem dupla ligação com a base de conhecimento, representando o domínio do problema codificado, isto concretiza que normalmente existe um conhecimento prévio do resultado a ser obtido (GONZALEZ; WOODS, 2010).

2.3.1 Pré-processamento

Segundo González e Woods (2010), a etapa de pré-processamento aplica técnicas e transformações na imagem digitalizada, e.g. redimensionamento, redução de ruídos, que resultem em uma nova imagem aprimorada. As transformações realizadas na imagem podem ser em domínio espacial ou de domínio de frequência (SZELISKI, 2011). O método de domínio espacial aplica transformações diretamente nos pixels da imagem e podem ser definidas pela equação vista no Quadro 1, tendo $f(x, y)$ como imagem de entrada, e $g(x, y)$ como imagem resultante da função de pré-processamento, T um operador sobre f , este podendo ser, por exemplo, uma operação de média ou somatório, aplicado a uma vizinhança do pixel no espaço (x, y) (SZELISKI, 2011).

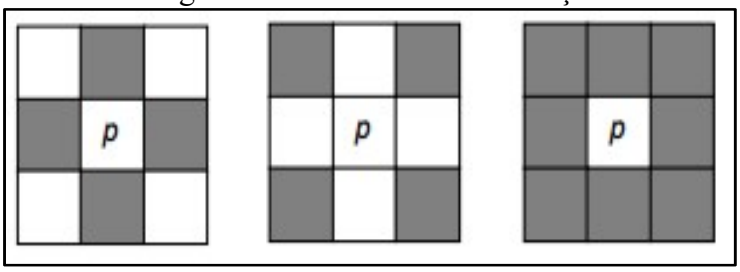
Quadro 1 - Função de processamento de imagens em domínio espacial

$$g(x,y) = T[f(x,y)]$$

Fonte: Gonzalez e Woods (2010, p. 68).

Segundo González e Woods (2010), o conceito de vizinhança de um pixel (x, y) tem embasamento na conectividade entre pixels, a representação mais comum é denominada de $N_4(p)$, cada pixel p nas coordenadas (x, y) possui quatro vizinhos horizontais e verticais, estes possuem as coordenadas $(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)$. Existe também a representação $N_8(p)$ que é a soma de $N_4(p)$ com suas diagonais $N_8(p) = (x + 1, y + 1), (x + 1, y - 1), (x - 1, y + 1), (x - 1, y - 1)$. Uma abordagem utilizada para definir a vizinhança de um pixel é usar um núcleo (*kernel*). Este é representado na forma de uma subimagem, uma matriz quadrada e com quantidade ímpar elementos, sendo o pixel em operação localizado ao centro. Figura 5 ilustra os tipos de vizinhança em um núcleo 3x3.

Figura 5 - Conceito de vizinhança

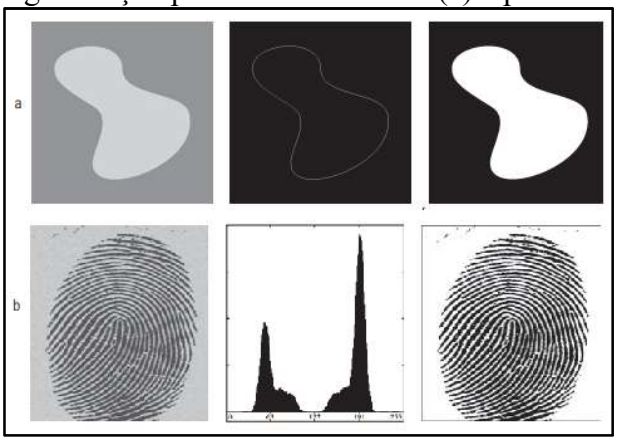


Fonte: Marques Filho e Vieira Neto (1999, p. 26).

2.3.2 Segmentação

A etapa de segmentação subdivide a imagem digital em partes ou em regiões de interesse, que, segundo Gonzalez e Woods (2010), é uma das tarefas com maior grau de dificuldade. Os algoritmos de segmentação de imagens em escala de cinza utilizam os processos de descontinuidade e similaridade. A Figura 6 ilustra a utilização dos dois métodos. O primeiro analisa a queda brusca no nível de cinza na região, utilizado para reconhecimento de bordas, pontos e linhas. O segundo processo, de similaridade, utiliza técnicas como a limiarização, estipulando um limiar T sobre todos os pixels da imagem $f(x, y)$. Caso o pixel seja maior que o limiar estipulado, então o pixel está dentro da região do objeto, caso contrário o pixel pertence ao fundo da imagem, separando assim os objetos em duas classes.

Figura 6 - Segmentação por descontinuidade (a) e por similaridade (b)

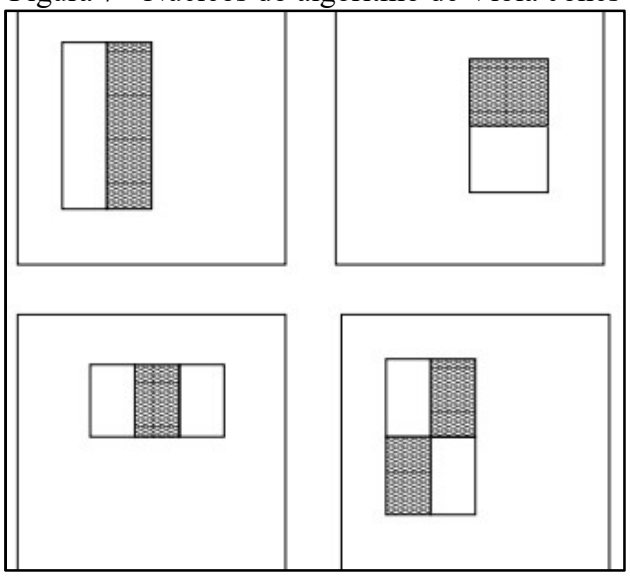


Fonte: Gonzalez e Woods (2010, p.455 e 489).

2.3.3 Extração de características

A etapa de extração de características tem sua fundamentação na morfologia matemática, que, segundo Serra (1983), é a teoria de análise de estruturas espaciais embasada na teoria dos conjuntos. Esta etapa recebe como entrada uma imagem e retorna um conjunto de informações referentes os dados da imagem para representação e descrição desse conjunto. Um algoritmo já consolidado para extração de características é o Viola-Jones, que realiza o reconhecimento da face em uma imagem, utilizando quatro tipos de núcleos, conforme ilustrados na Figura 7 (VIOLA; JONES, 2001).

Figura 7 - Núcleos do algoritmo de Viola-Jones



Fonte: Viola e Jones (2001, p. 512).

Segundo Viola e Jones (2001), o conjunto de características é dado pela diferença entre o somatório dos pixels das regiões retangulares. Este pode ser obtido transformando a imagem em sua representação integral, conforme apresentado no Quadro 2. Ainda segundo os autores,

a vantagem em utilizar a representação integral da imagem é a facilidade para realizar o cálculo de uma área retangular da imagem, obtendo apenas quatro pixels da matriz.

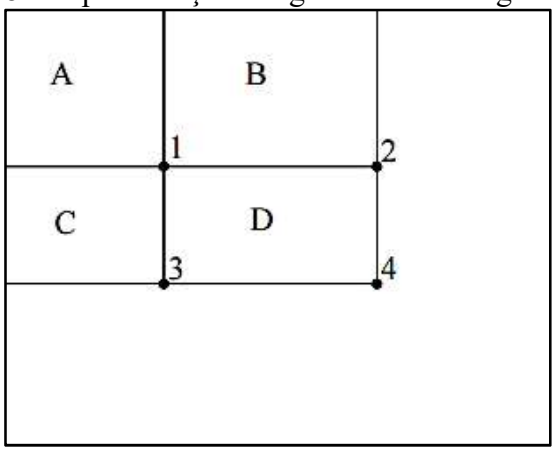
Quadro 2 - Cálculo da integral de uma imagem

$$ii(x,y) = \sum_{x \le x',\ y \le y'} i(x',y')$$

Fonte: Viola e Jones (2001, p. 512).

Conforme ilustrado na Figura 8, o valor da imagem integral localizada no pixel 1 é o somatório dos pixels da região A. O valor da integral no pixel 2, é o somatório dos pixels de $A + B$. O valor da integral no pixel 3 é o somatório dos pixels das regiões $A + C$. O valor da integral do pixel 4 é o somatório de $A + B + C + D$. Logo, o resultado da soma dos pixels da região D pode ser computado por $\sum_{(x,y)}^{D} = ii(4) + ii(1) - (ii(2) + ii(3))$.

Figura 8 - Representação integral de uma imagem digital



Fonte: Viola e Jones (2001, p. 513).

2.3.4 Representação e descrição

Segundo Gonzalez e Woods (2010), na etapa de representação e descrição, as características relevantes extraídas são convertidas para uma representação computacional, denominada de representação por fronteira ou representação por região, dependendo da função geradora de segmentação, limiarização ou similaridade. A representação por fronteira é utilizada quando foco está nas características externas do objeto, como a identificação de vértices. A representação por região é utilizada quando o interesse está nas características internas do objeto, como na identificação da forma.

Ainda segundo os autores, o processamento da descrição da imagem consiste em selecionar características que gerem informação qualitativa, para ser utilizada na classificação dos objetos. Esta etapa caracteriza um ponto chave no processo reconhecimento do objeto,

demandando um bom conhecimento do domínio do problema, pois a partir dos descritores selecionados é realizado separação entre as classes de objetos na fase de reconhecimento.

2.3.5 Reconhecimento de padrões

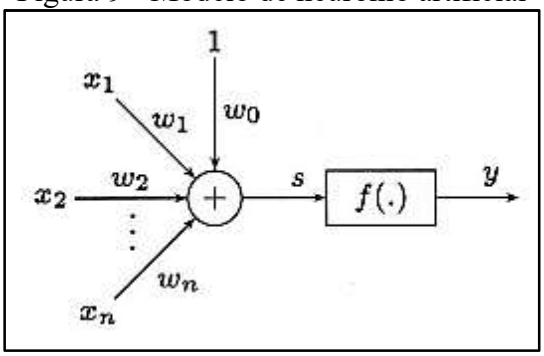
O mais alto nível do sistema de visão computacional é o reconhecimento de padrões e interpretação da imagem. Um padrão é um conjunto de características, em sua forma representativa, e denotada pela sua descrição. Gonzalez e Woods (2010, p. 17) afirmam que "O reconhecimento é o processo que atribui um rótulo (por exemplo, 'veículo') a um objeto com base em seus descritores.". Este processo de reconhecimento de um padrão dá-se através de duas categorias: decisão teórica e por reconhecimento estrutural. A categoria mais comum utilizada é a de decisão teórica, implementando o classificador através da técnica de matching. Esta técnica calcula a distância mínima, no espaço euclidiano, para cada novo padrão desconhecido, e o classifica para a classe de maior proximidade. Outra técnica utilizada para reconhecimento por decisão teórica é a rede neural artificial, descrita na seção seguinte. A categoria de reconhecimento estrutural, busca encontrar padrões com base na representação qualitativa dos seus descritores, porém não será abordada neste trabalho.

2.4 REDE NEURAL ARTIFICIAL

Segundo Gonzalez e Woods (2010), uma rede neural artificial é semelhante ao modelo biológico. Cada unidade de processamento em uma rede neural é denominada de neurônio e está ligado a n outras unidades de processamento. A computação de um neurônio é a uma combinação do processamento produzido por outros neurônios ao qual está conectado. A Figura 9 ilustra o modelo de neurônio artificial proposto por Mcculloch e Pitts (1943).

Segundo Mcculloch e Pitts (1943), o processo de classificação realizado pelo neurônio artificial é análogo ao processo químico de transmissão de um sinal elétrico em um neurônio biológico, o qual libera a passagem do sinal, representado matematicamente por uma característica x_i, de acordo com um limiar.

Figura 9 - Modelo de neurônio artificial



Fonte: Marques (2005, p. 164).

O limiar é computado de acordo com o aumento ou queda do potencial elétrico no neurônio, descrito no Quadro 3. A função $f(.)$, denominada de função de ativação, recebe como entrada $d(x)$, que é a soma ponderada entre um elemento do vetor de características $x = [1, x_1, x_2, \ldots, x_n]^t$, por um elemento do vetor pesos (*weight*) $\omega = [\omega_1, \omega_1, \ldots, \omega_n]^t$. A função realiza a limiarização por decisão binária com base em um limiar (*threshold*) t, e considera-se que o neurônio está ativo se $y = 1\ para\ s > t$, e considera-se que o neurônio está inativo se $y = 0\ para\ s \leq t$ (MARQUES, 2005).

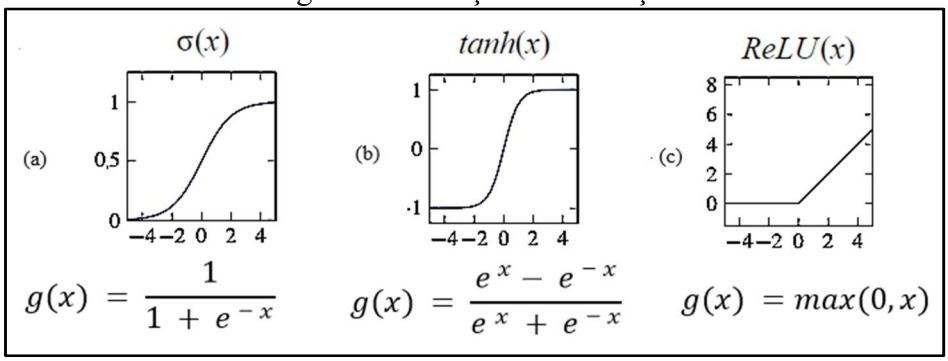
Quadro 3 - Computação do neurônio artificial

$$s = d(x) = \sum_{i=1}^{n} \omega_i x_i + \omega_{n+1}$$

Fonte: Gonzalez e Woods (2010, p. 581).

Existem diferentes abordagens que definem qual função de ativação $f(.)$ deve ser utilizada. O tipo de função está relacionando ao tipo de dados que a função será exposta. A Figura 10 ilustra algumas funções de ativação. Pode ser uma função para dados lineares, conforme já descrito anteriormente, pode ser função para dados não lineares como função sigmoide (a), sigmoide hiperbólica tangente (b), ou uma função retificadora linear (c).

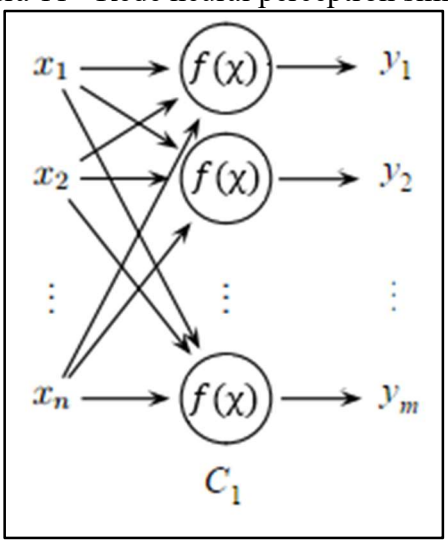
Figura 10 - Funções de ativação



Fonte: elaborado pelo autor.

A topologia mais básica de uma rede neural é denominada Rede Alimentada Adiante (FeedForward). Esta é disposta em camadas e a saída gerada por um neurônio serve de entrada para outros n neurônios. Segundo Marques (2005, p. 165), "as entradas das unidades de uma camada são saídas das unidades da camada anterior". A Figura 11 ilustra uma representação simples de rede neural, de camada única, criada na década de 50 por Rosenblatt (1962). Esta representação deu origem às classes de máquinas que aprendem denominadas de perceptrons.

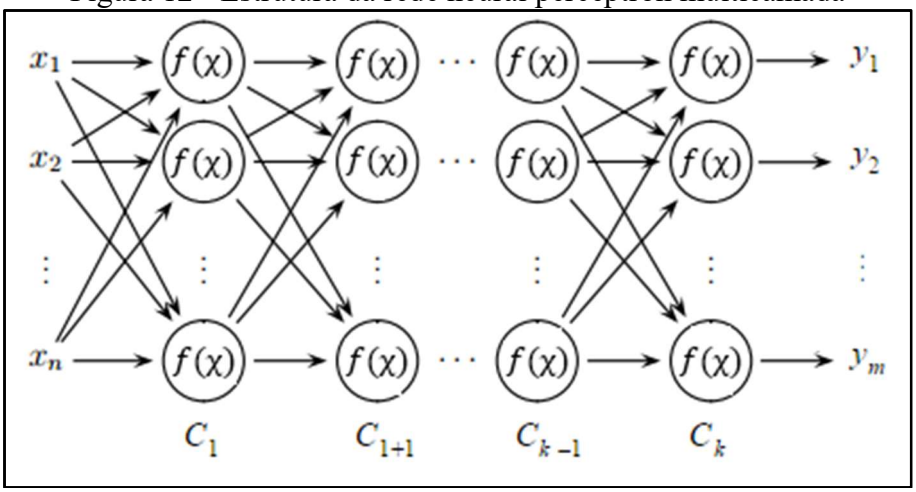
Figura 11 - Rede neural perceptron simples



Fonte: elaborado pelo autor, baseado em Gonzalez e Woods (2010, p. 585).

Esta arquitetura torna possível resolver problemas linearmente separáveis, porém apresenta resultados insatisfatórios para características que não podem ser separadas linearmente. Para tanto, existe diversas outras estruturas de redes neurais. Uma estrutura comumente utilizada, para reconhecimento de multiclasses, ou seja, dados não lineares, é a Perceptron Multicamada (Multi-Layer Perceptron - MLP), ilustrada na Figura 12. Esta pode conter um numero k de camadas (GONZALEZ; WOODS, 2010). Semelhante ao modelo biológico, i.e., o neurônio humano, cada unidade de processamento em uma rede neural artificial está ligada a n outras unidades de processamento e a sua computação é uma combinação linear de entradas produzidas por outros neurônios ao qual está conectado.

Figura 12 - Estrutura da rede neural perceptron multicamada



Fonte: elaborado pelo autor, baseado em Gonzalez e Woods (2010, p. 585).

A camada C_1 é denominada de camada de entrada (*input*), as camadas localizadas entre C_{1+1} até C_{k-1} são denominadas camadas ocultas, posteriormente está a camada C_k, chamada de camada de saída. Neste tipo de rede a função de ativação do neurônio é uma função sigmoide e a saída gerada está no intervalo entre [0,1]. O reconhecimento de uma classe y_i, é dado quando a *k*-ésima saída da rede possui valor probabilístico considerado alto, enquanto as demais saídas possuem um valor considerado baixo. Valores altos e baixos podem ser considerados como 0,95 e 0,05 respectivamente (MARQUES, 2005).

Segundo Marques (2005, p. 163), "[...] as redes neurais artificiais são constituídas por elementos simples interligados, com capacidade de aprendizagem a partir dos dados". A capacidade de aprendizagem de uma rede neural é dada pela correção automática dos seus pesos (*weights*). O aprendizado das redes neurais perceptrons é realizado pelo ajuste adequado dos pesos $\omega = [\omega_1, \omega_1, \ldots, \omega_n]^t$, realizada através do cálculo do erro que pode ser visto no Quadro 4. Este ajuste é atualização iterativa, até encontrar o valor correto para todos os pesos, ou um valor aceitável, denominado *learning rate*, a fim de minimizar a medida do erro. O erro é a derivada do valor esperado pelo valor calculado na rede para cada saída.

Quadro 4 - Atualização dos pesos pelo erro em uma rede perceptron

$$\Delta\omega_{qp} = -\alpha \frac{\partial E_Q}{\partial \omega_{qp}}$$

Fonte: Gonzalez e Woods (2010, p. 587).

Na temática de encontrar valores corretos para os pesos $\omega = [\omega_1, \omega_1, \ldots, \omega_n]^t$, utiliza-se um hiperparâmetro denominado de viés (*bias*), cujo usualmente possui o valor $b = 1$. Este hiperparâmetro tem a finalidade de aumentar o grau de liberdade da função, fornecendo uma

melhor adaptação ao aprendizado da rede (MARQUES, 2005). Para redes MLP o aprendizado se dá pela aplicação de técnicas de retropropagação do erro (Backpropagation), como a descida gradiente que busca de minimizar o erro quadrático. O Quadro 5 apresenta a equação de retropropagação, uma variação da regra delta de Widrow-Hoff, que busca reduzir a taxa da função de erro visto no Quadro 4.

Quadro 5 - Equação de retropropagação do erro

$$E_Q = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - O_q)^2$$

Fonte: Gonzalez e Woods (2010, p. 587).

Segundo Lecun et al. (1998), mesmo após a definir a melhor técnica a ser utilizada pelo sistema de visão computacional, ainda persiste o problema de escolher quais são os pontos relevantes (características) a serem extraídos e descritos, bem como quais são os pesos exatos para ter uma boa taxa de precisão. A taxa de precisão do reconhecimento está relacionada diretamente com a escolha das melhores características. Atualmente esta escolha é realizada por um especialista em reconhecimento de padrões no domínio do problema. A fim de resolver este problema, a comunidade vem, recentemente, utilizado a técnica de Deep Learning, abordada na próxima seção.

2.4.1 Rede Neural Profunda (Deep Learning - DL)

O Aprendizado Profundo de Máquina (conhecido na comunidade internacional como Deep Learning - DL) paira sobre a problemática de treinar uma rede neural artificial de forma que camadas de neurônios estejam dispostas hierarquicamente, sejam organizadas sequencialmente ou de forma análoga ao sistema biológico, representando os campos receptivos do córtex visual, estes responsáveis pelo reconhecimento de padrões. O sistema biológico do córtex compartilha uma estrutura de oito camadas, sendo seis delas ocultas (DICARLO; ZOCCOLAN; RUST, 2012).

Aprendizado Profundo consiste no aprendizado em várias camadas hierarquizadas, baixo e alto nível, compostas de múltiplas transformações não lineares (BENGIO; COURVILLE; VINCENT, 2013). As camadas de baixo nível são responsáveis por identificar características específicas, por exemplo a identificação de bordas, forma geométrica. Este processo é executado sequencialmente até as camadas de alto nível, que possuem um maior nível de abstração. A camada final utiliza o conhecimento adquirido nas camadas anteriores para classificação do reconhecimento do padrão. Este reconhecimento é denominado

aprendizagem de representação, que permite alimentar o sistema de visão computacional com dados brutos, no contexto deste trabalho, uma imagem, e descobrir automaticamente quais características são relevantes para a detecção e classificação dos padrões, sem especificar explicitamente quais os pontos relevantes. (LECUN; BENGIO; HINTON, 2015).

Outro fator importante das redes neurais profundas é referente a conectividade das unidades de processamento, a qual, diferentemente de uma rede neural artificial padrão, a saída do processamento de cada neurônio não está completamente conectada a entrada de todos os neurônios da camada adjacente, devido ao alto custo de processamento da pré-ativação das funções de decisão de cada neurônio. Usualmente três tipos de classes de redes profundas são utilizados: redes auto codificadoras, redes recorrentes e redes convolucionais. Esta última é abordada mais detalhadamente por ser utilizada no contexto deste trabalho.

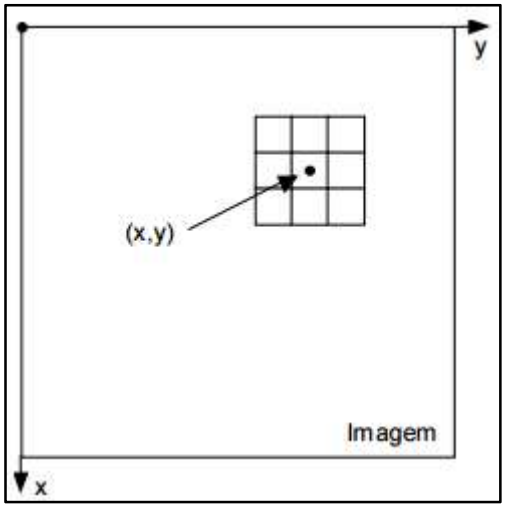
A Rede Auto Codificadora (Autoenconders) normalmente não é utilizada para classificar algum padrão, mas para representar o próprio dado da camada de entrada, de forma mais aproximada possível, na camada de saída. Esta rede realiza dois tipos de transformações sobre os dados: a primeira denominada de função de codificação que transforma os dados em uma forma de representação; e a segunda chamada de função decodificadora que realiza o processo inverso dado como entrada para a representação (GOODFELLOW; BENGIO; COURVILLE, 2016). O tipo de função de transformação a ser utilizada vai depender do domínio do problema, tipicamente é utilizado uma função sigmoidal, vista anteriormente na Figura 10. O treinamento deste modelo de rede é dado pela otimização do erro, representada por uma função de custo.

A Rede Neural Recorrente (Recurrent Neural Networks - RNN) possui uma peculiaridade na evolução do seu estado, que é a geração de um contexto baseado no estado atual da rede em correlação com o tempo. As conexões deste tipo de rede possuem ciclos, na qual são descritos como uma memória para arquivar os dados processados pela rede. O processamento nas camadas ocultas depende do conhecimento adquirido pelas unidades no tempo $t-1$ para o processamento no tempo t (NEVEROVA et al., 2016). Este tipo de rede é usualmente utilizado para processamento de linguagem natural. Para predizer a próxima palavra em uma frase no tempo t, é necessário o conhecimento do estado no tempo $t-1$. A rede recorrente utiliza a técnica de compartilhamento de pesos, a qual define que os pesos de ativação de um neurônio no tempo t são compartilhados para o próximo estado da rede em $t+1$. (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.4.1.1 Rede Neural Convolucional

A Rede Neural Convolucional (Convolutional Neural Network - CNN) é amplamente realizada para o reconhecimento de padrões em imagens. Este modelo reconhece padrões através da convolução de imagens. A convolução gera uma nova imagem resultante da do processamento da imagem original, dada como entrada, sobre um operador linear invariante, comumente é utilizado o kernel, e denomina-se máscara de convolução. De fato, o uso da convolução é a implementação do deslocamento do núcleo pela imagem resultando também conhecido como mapa de características (Feature Map). A Figura 13 ilustra o deslocamento do núcleo em uma imagem.

Figura 13 - Deslocamento do kernel em uma imagem



Fonte: Gonzalez e Woods (2010, p. 84).

O Quadro 6 descreve o deslocamento da convolução em uma imagem bidimensional $M\ x\ N$, sendo S o mapa de características resultante da convolução, I a imagem de entrada, representando a função $f(x, y)$, K é o operador linear, representando o núcleo (GOODFELLOW; BENGIO; COURVILLE, 2016).

Quadro 6 - Convolução discreta

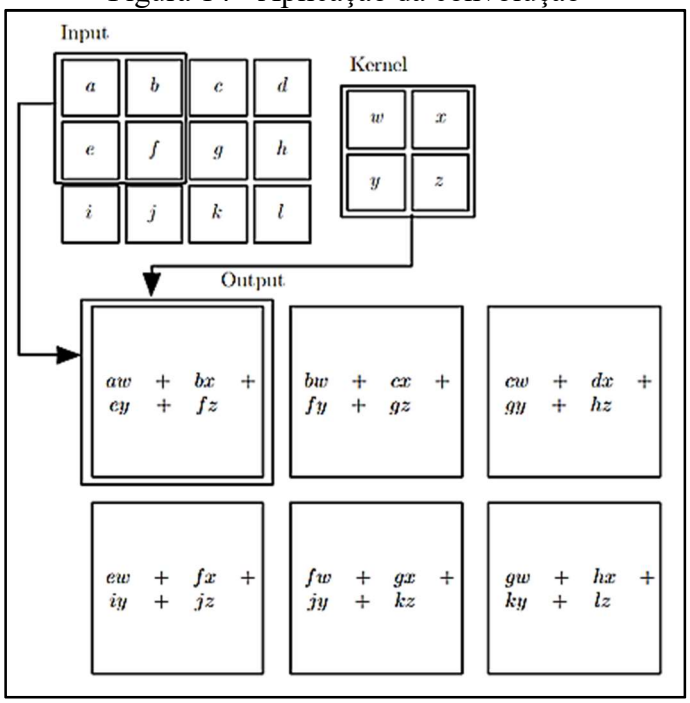
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

Fonte: Gonzalez e Woods (2010, p.).

Segundo Goodfellow, Bengio e Courville (2016), o processamento de reconhecimento de uma CNN padrão é feito em três etapas: convolução de cada camada de entrada pelos núcleos extratores de características; a aplicação de uma função de ativação não linear; e a aplicação função de subamostragem (*Pooling*), a fim de causar modificações na camada de saída. Observa-se na Figura 14 que a etapa de convolução é definida pela passagem do núcleo

(*kernel*) pela imagem (*input*), e, o conhecimento adquirido desta computação, é disposto em um conjunto denominado mapa de características convoluídas (*output*). Estes mapas de características são compartilhados com toda a rede CNN, para que sejam utilizados por outras regiões do conjunto de entrada, a fim de realizar o reconhecimento de padrões.

Figura 14 - Aplicação da convolução

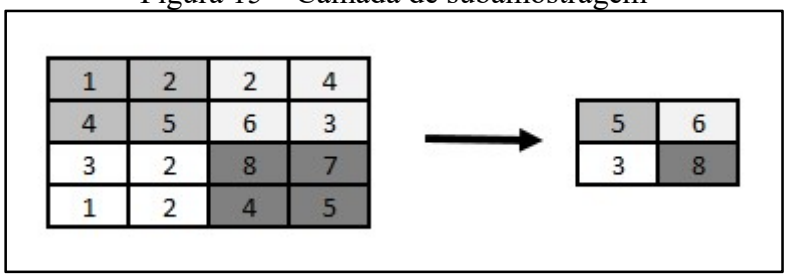


Fonte: Goodfellow, Bengio e Courville (2016, p. 330).

A função de ativação comumente utilizada, por tratar-se de imagens, é a ReLU, também conhecida como função de ativação retificadora. A função retificadora possui vantagens matemáticas em relação a outras funções como a Sigmoide e Tangente, devido a sua propriedade de não negatividade. A resposta oposta ao reconhecimento do padrão pela unidade de processamento será zero (0), facilitando a representação esparsa, pois, após a pré-ativação das unidades de processamento, 50% das unidades ocultas possuíram o valor zero em sua saída (GLOROT; BORDES; BENGIO, 2011). Uma das variações da função de ativação ReLU utilizado neste trabalho, é PReLU que adiciona a cada camada um parâmetro extra informando que o coeficiente é negativo. A adição deste parâmetro extra não gera um aumento relevante no custo computacional, tão pouco aumenta a chances de sobreposição (overfitting) (HE et al., 2015). A sobreposição é um termo utilizado para descrever quando um modelo estatístico se ajusta aos dados de treinamento, de forma a quase memorizar o aprendizado, e assim tornando-se ineficiente para prever novos resultados. (GOODFELLOW; BENGIO; COURVILLE, 2016).

A camada de subamostragem (pooling), é o processo de evolução da rede, tornando-se a camada de nível superior mais abstrata a partir dos padrões obtidos na camada inferior adjacente. A Figura 15 ilustra a criação da camada de subamostragem, gerada pela técnica de maior valor, conhecida como Max Pooling.

Figura 15 – Camada de subamostragem



Fonte: elaborado pelo autor.

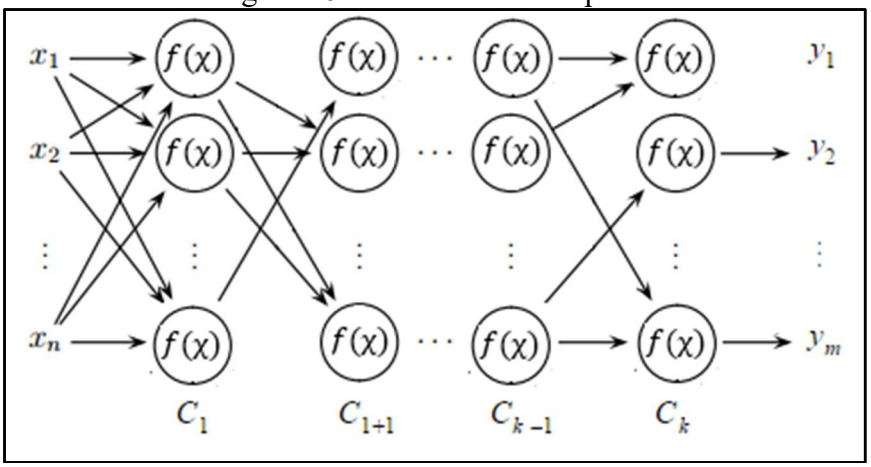
Esta técnica extrai o maior valor dos mapas de características, criados pela computação da convolução, e os agrupa em uma nova camada (ZHOU; CHELLAPPA, 1988). Esta camada busca reduzir a invariância dos mapas de características através da redução espacial dos dados, ou seja, caso o primeiro quadrante apresentado pela Figura 15, possuísse os números x = [4, 3, 4, 5], ainda assim, após a aplicação do Max Pooling, a saída da camada não seria alterada, gerando o mesmo mapa de característica apresentado na imagem (SCHERER; MÜLLER; BEHNKE, 2010). Existem várias técnicas que podem ser utilizadas para o agrupamento das características na camada de subamostragem, como agrupamento pela média, pela norma e pelo maior valor.

A convolução em imagens produz grande quantidade de parâmetros a serem computados. Em uma rede neural artificial padrão, uma entrada de tamanho 32x32x3, representando altura, largura e profundidade (canais de cor), gera uma quantidade de 32 * 32 * 3 = 3.072 parâmetros, ou seja, pesos a serem computados por cada neurônio. A topologia da rede neural convolucional, possui algumas particularidades que auxiliam na redução e tratamento desta computação, este são: conectividade esparsa das unidades de processamentos; campo receptivo local; arranjo espacial; compartilhamento de pesos; e Dropout (GOODFELLOW; BENGIO; COURVILLE, 2016).

Segundo Goodfellow, Bengio e Courville (2016), a conectividade esparsa, em conjunto com o campo receptivo, define, diferentemente das redes neurais tradicionais, que nem todas as unidades de processamento sejam ativadas para determinado padrão, ou seja, o resultado do processamento de uma unidade não está ligado a todas as unidades da camada superior adjacente, mas a apenas um pequeno grupo. Ainda segundo os autores, esta abordagem diminui a complexidade do cálculo da rede em tempo de execução, pois reduz a quantidade de

operações que seriam computadas. Em uma rede com M entradas e N saídas, a matriz de convolução em uma rede completamente conectada possuiria $O_{(M \times N)}$, com a conectividade esparsa, cada saída da rede possuirá apenas K entradas, reduzindo a complexidade para $O_{(k \times N)}$. Figura 16 ilustra a conectividade esparsa entre as unidades de processamento.

Figura 16 - Conectividade esparsa



Fonte: Elaborado pelo autor, baseado em Gonzalez e Woods (2010, p. 585).

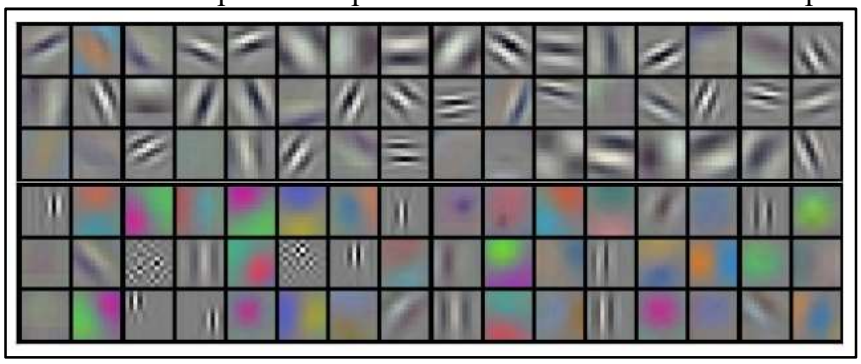
O campo receptivo é um hiperparâmetro que corresponde a análise de uma pequena porção do conjunto de dados de entrada, realizado pelas camadas no nível mais inferior. Subsequentemente a camada de nível superior abrange a combinação de alguns campos receptivos da camada inferior adjacente, permitindo assim que a camada superior aprenda características com um maior nível de abstração (LINDEBERG et al., 2013). Na prática o campo receptivo é equivalente ao tamanho do núcleo utilizado. Para um núcleo de 5x5x3, o campo receptivo será encarregado de computar 5 * 5 * 3 = 75 pesos + 1 viés.

O arranjo espacial é constituído por três hiperparâmetros que definem a quantidade de neurônios utilizadas em cada camada da rede convolucional, ou seja, o volume da saída da rede. Estes hiperparâmetros são: a profundidade (*depth*), que equivale a quantidade de núcleos que serão utilizados; o passo (*stride*), que representa o deslocamento núcleo pelos dados de entrada, ou seja, um passo de tamanho 2 equivale ao núcleo deslocar-se 2 pixels por vez, produzindo como saída dados menores que os da entrada; e o preenchimento (*padding*) que é uma técnica que adiciona zeros (0) as bordas dos dados para auxiliar tanto na convolução como também para tornar o volume de saída igual ao de entrada.

Segundo Goodfellow, Bengio e Courville (2016), o compartilhamento de pesos é outra técnica aplicada a fim de reduzir a quantidade de processamentos realizados pela rede. Os

autores citam a AlexNet3 (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) como exemplo da importância do compartilhamento de pesos. Sem o uso compartilhamento de pesos, o volume de entrada utilizada na AlexNet seria gerado um volume total de 105,705,600 parâmetros para serem processados na primeira camada de convolução. A rede neural convolucional pressupõe a lógica que, um peso considerado, bom para um pixel $p(x, y)$, também será bom para $p(x_2, y_2)$, i.e., se os pesos que compõem um núcleo obterem uma baixa taxa de perda, para uma determinada região da imagem, então estes mesmos pesos podem ser úteis em outra região da imagem. Com isto os pesos são compartilhados para todos os neurônios. Estes pesos compartilhados são mantidos em um vetor, comumente chamado de núcleo ou mapa de características. A computação dos parâmetros é realizada sobre a convolução da imagem de entrada por esse núcleo, assim reduzindo para 34,944 parâmetros de pesos a serem computados. Figura 17 ilustra os 96 núcleos aprendidos pela AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), observa-se na imagem que alguns núcleos são responsáveis pela extração de características de frequência, como bordas e outros núcleos responsáveis pela identificação do padrão de bolhas de cor.

Figura 17 - Núcleos aprendidos por uma rede neural convolucional profunda



Fonte: Krizhevsky, Sutskever e Hinton (2012, p. 6).

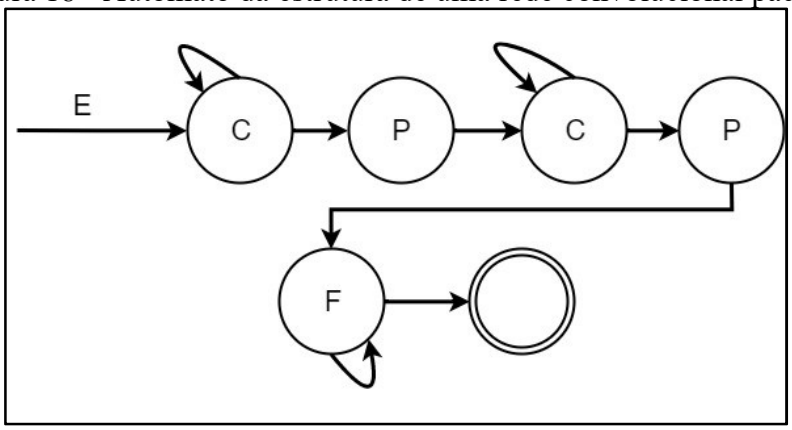
O Dropout é uma técnica utilizada para reduzir o sobreajuste da rede que comumente acontece quando a rede neural é treinada para um conjunto pequeno de dados (SRIVASTAVA et al., 2014). Segundo Srivastava et al. (2014), esta técnica realiza uma mutação na rede neural, semelhante a mutação dos algoritmos genéticos, desativando aleatoriamente 50% dos detectores de características da rede na fase de treinamento, evitando que a rede se adapte aos dados. Esta técnica torna as camadas ocultas mais robustas.

Dado as particularidades de uma rede neural convolucional, torna-se mais compreensível a visualização da topologia integral da rede. A forma de estruturar as camadas desta rede CNN é dada pelo autômato ilustrado na Figura 18, sendo E a camada de entrada, C

[3] AlexNet é uma rede neural convolucional de aprendizado profundo, desenvolvida por Krizhevsky et al. (2012).

a camada de convolução, P a camada de subamostragem e F a camada de classificação. A camada de classificação possui conectividade não esparsa, na qual todas as unidades de processamento possuem conectividade com todas as outras unidades.

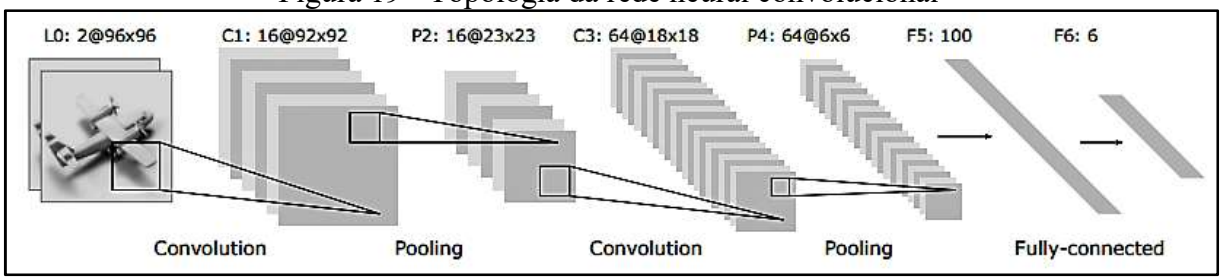
Figura 18 - Autômato da estrutura de uma rede convolucional padrão



Fonte: elaborado pelo autor.

Nota-se pela Figura 19, ilustração de uma CNN padrão, que o processo de subamostragem mantém a mesma quantidade de mapas de características, 64 núcleos com dimensão 18x18 (64@18x18), devido a matriz de Hadamard, que define que o produto de duas matrizes $X_{m \times n}$ e $Y_{m \times n}$ resulta em uma matriz $G_{m \times n}$ tal que $g_{i\,j} = x_{i\,j}\, y_{i\,j}$ (HORADAM, 2012). Porém, existindo outras camadas de convolução, o número de mapas aumenta novamente de acordo com o número de núcleos extratores aos quais a rede será convoluída.

Figura 19 - Topologia da rede neural convolucional



Fonte: Scherer, Müller e Behnke (2010, p 3).

Um dos grandes problemas da área de visão computacional é a invariância de iluminação. Uma variação na estrutura da rede neural convolucional trata essa problemática com a inclusão de nova camada após a camada de pooling, denominada camada de normalização. Esta realiza a normalização do contraste, aplicando a transformação no campo receptivo local. Tipicamente, após a última camada de subamostragem, é incluído uma ou mais camadas completamente conectadas, responsáveis pelo estágio de classificação. O aprendizado de rede neural profunda, voltada para o reconhecimento de padrões em imagens, se dá técnica de retropropagação, na qual corrige os pesos de cada neurônio através de uma

função de perda. Comumente utiliza-se uma variação da técnica de descida gradiente, denominado de descida gradiente estocástica. Esta variação utiliza a média dos gradientes dos pesos compartilhados.

2.5 TRABALHOS CORRELATOS

Neste capítulo serão apresentados três trabalhos correlatos, que possuem características semelhantes a proposta desta monografia. A seção 2.5.1 aborda o trabalho de Bartlett et al. (2003) que consiste em um sistema de reconhecimento de expressões faciais em tempo real. A seção 2.5.2 apresenta o trabalho de Tang e Huang (2008), que tem como objetivo reconhecer expressões faciais em imagens 3D. A seção 2.5.3 apresenta um sistema de reconhecimento de expressões faciais para uso em tratamentos de psicoterapia, desenvolvido por Candra et al. (2016). Por fim, a Seção 2.5.4 descreve o trabalho de Amin, Chase e Sinha (2017), que análogo ao trabalho proposto, busca reconhecer expressões faciais utilizando a técnica de aprendizado profundo.

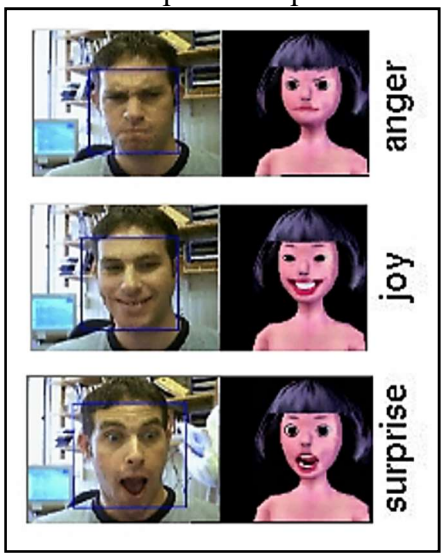
2.5.1 Real time face detection and facial expression recognition

O trabalho Real time face detection and facial expression recognition foi desenvolvido por Bartlett et al. (2003) e tem como objetivo localizar automaticamente faces em um fluxo de vídeo e codificar a expressão visual dinamicamente. Os autores discutem que a comunicação face a face é uma operação em tempo real e com uma escala de tempo em 40 milissegundos. Eles afirmam que o tempo aumenta o nível de incerteza a ser considerado, sendo necessário que os humanos possuam uma rica percepção sensorial ao invés de lentos processos de inferência. O sistema é capaz de detectar sete expressões: neutra, raiva, desgosto, medo, alegria, tristeza e surpresa, diferenciando de outros trabalhos pois opera em tempo real.

Na fase de pré-processamento não é realizada a detecção e alinhamento explícito das características faciais poupando assim o processamento e melhorando o tempo de resposta, que é essencial para aplicações em tempo real. Para realizar a extração das características relevantes para o reconhecimento da região dos olhos os autores utilizaram o *framework* GBoost (FASEL; FORTENBERRY; MOVELLAN, 2005), desenvolvido na linguagem de programação C++, na qual realiza a detecção em tempo real dos olhos em uma imagem, através de um modelo probabilístico. GBoost estende o trabalho de Viola e Jones (2001) que utiliza recursos retangulares simples para detecção da face. O valor de um recurso de dois retângulos é a diferença entre a soma dos pixels dentro de duas regiões retangulares.

Os autores apresentam na fase de classificação uma abordagem utilizando o método Adaboost (FREUND; SCHAPIRE, 1996), que tem como objetivo melhorar o desempenho de algoritmos de aprendizagem fraca utilizando um conjunto de classificadores simples. Algoritmos de aprendizagem fraca possuem uma hipótese de classificação com taxa levemente superior que uma escolha randômica, com precisão mínima de que 50% + 1 (KEARNS; VALIANT, 1994). Em seguida, os classificadores produzidos pelo método Adaboost são combinados em um único classificador, Support Vector Machines (SVM). A Figura 20 ilustra a demonstração do potencial do trabalho desenvolvido, que, para tal, foi construído um espelho de emoção em tempo real. Isto é, um sistema virtual 3D que personifica em tempo real a imitação da expressão emocional de uma pessoa.

Figura 20 - Exemplos do espelho da emoção



Fonte: Bartlett et al. (2003, p 5).

O sistema de Bartlett et al. (2003) foi treinado e testado na base de dados Cohn-Kanade AU-Coded Expression Database (KANADE; COHN; TIAN, 2000). Esta base de dados contém o registro facial de 210 adultos na faixa etária entre 18 e 50 anos de idade, sendo 69% do sexo feminino e 31% masculino, e, 81% Euro-Americanos, 13% Afro-Americanos e 6% de outros grupos étnicos.

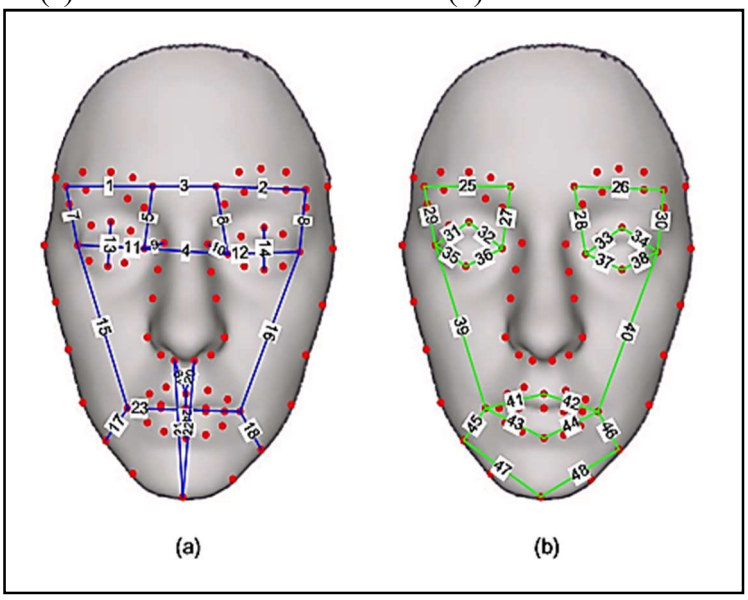
Os experimentos compararam o desempenho do reconhecimento da abordagem de detecção automática com a abordagem de detecção manual, nos quais foi constatado que não houve diferença significativa entre elas. O sistema apresentou nível de precisão de 93% de reconhecimento, na escolha de uma das 7 alternativas de expressões faciais.

2.5.2 3D facial expression recognition based on properties of line segments connecting facial features points

O trabalho 3D facial expression recognition based on properties of line segments connecting facial features points foi desenvolvido por Tang e Huang (2008). Este tem como objetivo reconhecer as seis emoções consideradas básicas e universais através de expressões faciais utilizando a geometria 3D. Os autores discutem que esta abordagem extrai características que são invariantes sob efeito de iluminação ou postura, características estas que são consideradas como obstáculos para o reconhecimento facial em imagem 2D.

Na fase de extração de características foi utilizado uma técnica baseada em propriedades de segmentos de linha que fazem conexão com determinados pontos de características 3D. A Figura 21 ilustra os 96 pontos distintos que são utilizados para a extração das características de distância e inclinação para o reconhecimento da emoção.

Figura 21 - (a) Características de distância. (b) Características de inclinação



Fonte: Tang e Huang (2008, p 3).

As características extraídas que mensuram a distância entre os pontos foram normalizadas, observado que a face de cada indivíduo possui tamanho e proporção diferente. Na fase de classificação das emoções os autores utilizaram um classificador SVM para reconhecer as emoções: raiva, desgosto, medo, alegria, tristeza e surpresa. Esta abordagem não considerou a classificação para expressão facial neutra. Para treinar o classificador SVM foi utilizado o software integrado LibSVM (CHANG; LIN, 2011) implementado na linguagem de programação C++. O LibSVM, em sua versão mais atual (2016), possui extensão para várias outras linguagens de programação.

Este trabalho utilizou, como base de treinamento e teste, a base de dados BU-3DFE (YIN et al., 2006). Esta base de dados é composta de 100 indivíduos, sendo 60% do sexo feminino e 40% do sexo masculino, com variedade étnica, incluindo branco, preto, Leste Asiático, Médio Oriente Asiático, Latino-Americano e outras.

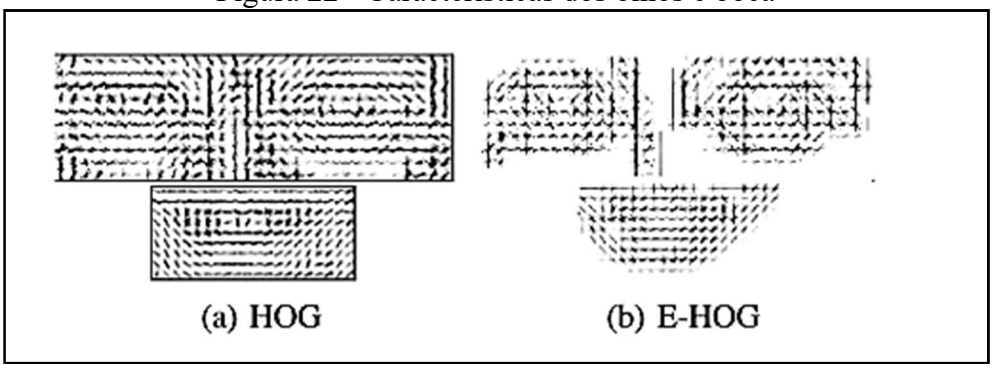
Como conclusão do trabalho, foi constatado que a abordagem produziu o aumento absoluto de 3,5% na taxa média de reconhecimento em relação ao trabalho anterior realizado por Wang et al. (2006). Esta abordagem obteve uma precisão média de 87,1%, sendo que a maior taxa foi de 99,2% para o reconhecimento da expressão facial de surpresa.

2.5.3 Classification of facial-emotion expression in the application of psychotherapy using Viola-Jones and Edge-Histogram of oriented gradient

O trabalho Classification of facial-emotion expression in the application of psychotherapy using Viola-Jones and Edge-Histogram of oriented gradient foi desenvolvido por Candra et al. (2016). Este tem como objetivo principal reconhecer as expressões faciais para auxiliar psicoterapeutas na escolha de métodos de tratamento mais apropriados. Para extração das regiões de interesse, isto é, face, olhos e boca, os autores utilizaram o algoritmo de Viola e Jones (2001). Em seguida, as características extraídas foram combinadas com um extrator de características geométricas, um Histograma de Gradiente Orientado (Histograms of Oriented Gradients - HOG) melhorado, que intitularam de Edge Histograms of Oriented Gradients (E-HOG). Este recurso utiliza a técnica de Canny (1986), algoritmo desenvolvido para detectar as bordas da imagem. Os dados resultantes do processamento da técnica de Canny (1986) são parametrizados como entrada para o processo de extração HOG. A Figura 22 ilustra os descritores retornados pelas abordagens HOG e E-HOG.

Os autores utilizaram para os experimentos a base de dados Cohn-Kanade AU-Coded Expression Database Version 2 (LUCEY et. al. 2010). O sistema realiza a classificação das seis emoções universais e uma emoção adicional, denominada nojo, e não abordada pelos outros trabalhos correlatos. Na fase de classificação, fase final do sistema, foi utilizado um classificador SVM para categorizar os dados reconhecidos em sete classes, sendo elas, as emoções de raiva, desgosto, medo, alegria, tristeza, surpresa e nojo.

Figura 22 - Características dos olhos e boca



Fonte: Candra et al. (2016, p. 425).

Os autores concluíram que utilizando o método proposto (E-HOG) atinge 96,4% de precisão média na identificação das sete expressões, com aprimoramento significativo no tempo de processamento, reduzindo em 83,07% para olhos, 82,20% para boca, 1833,33% para face e 97,67% para olhos e boca juntos. Porém, ao comparar o método HOG com o proposto E-HOG para o reconhecimento da face, constataram uma pequena redução na precisão, de 96,07% para 94,90% respectivamente. Sendo a diferença entre as técnicas de apenas 1,17%, os autores não descartam a utilização da nova abordagem, devido à melhora significava no tempo de processamento. A Figura 23 ilustra a tabela da matriz de confusão do melhor classificador treinado utilizado esta abordagem. As emoções são classificadas na tabela como: raiva (An), desprezo (Co), desgosto (Di), medo (Fe), alegria (Ha), tristeza (Sa) e surpresa (Su).

Figura 23 - Matriz de Confusão - Candra et al. (2016).

	An	Co	Di	Fe	Ha	Sa	Su
An	**97.2%**	0.0%	0.0%	1.4%	0.0%	1.4%	0.0%
Co	0.0%	**90.0%**	0.0%	8.6%	0.0%	1.4%	0.0%
Di	4.3%	0.0%	**95.7%**	0.0%	0.0%	0.0%	0.0%
Fe	1.4%	0.0%	1.4%	**94.4%**	1.4%	0.0%	1.4%
Ha	0.0%	0.0%	0.0%	0.0%	**100.0%**	0.0%	0.0%
Sa	1.4%	0.0%	0.0%	0.0%	0.0%	**98.6%**	0.0%
Su	0.0%	0.0%	0.0%	0.0%	0.0%	1.4%	**98.6%**
Average Accuracy = 96.4%							

Fonte: Candra et al. (2016, p. 426).

2.5.4 Touchy Feely

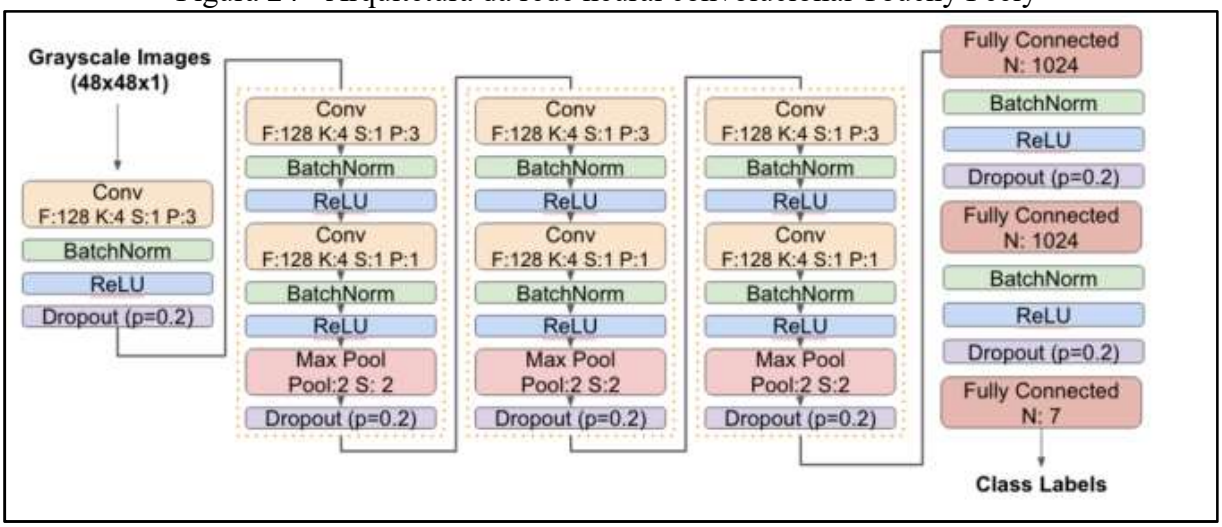
O Touchy Feely foi desenvolvido por Amin, Chase e Sinha (2017) e busca reconhecer emoções através de expressões faciais utilizando a técnica de aprendizado profundo. Segundo

Amin, Chase e Sinha (2017), a utilização de redes neurais convolucionais, na abordagem de identificação de emoções, atinge uma precisão média de 60%.

Para desenvolvimento do trabalho foi utilizado a base de dados Facial Expression Recognition 2013 (FER-2013). Esta base de dados foi desenvolvida por Pierre Luc Carrier e Aaron Courville, e possui um total 35887 imagens, com dimensões de 48x48 pixels (GOODFELLOW et al., 2013). As amostras possuem as seis expressões faciais de Ekman, adicionadas da expressão neutra. As amostras desta base de dados estão distribuídas em: 7215 imagens de alegria; 436 de desgosto; 4097 de medo; 4965 de neutra; 3995 de raiva; 3171 de surpresa; e 4830 de tristeza.

Os autores criaram sua própria rede convolucional para classificar as sete emoções disponíveis na base de dados. Contudo, realizaram várias tentativas de arquitetura e configuração de hiperparâmentros. A arquitetura que apresentou melhor desempenho pode ser vista na Figura 24.

Figura 24 - Arquitetura da rede neural convolucional Touchy Feely



Fonte: Amin, Chase e Sinha (2017, p. 4).

A arquitetura consiste em um bloco convolucional inicial, que possui uma camada de convolução, seguida de uma camada de normalização, posteriormente a camada de ativação, utilizando função Relu, e ao final do bloco uma camada de Dropout. Os três blocos seguintes da rede, são constituídos por um agrupamento do primeiro bloco, exceto pela camada de Dropout que é adicionada ao final do bloco. A camada final da rede, responsável pela classificação, é formada por três camadas densas, intercaladas por uma camada de normalização, uma de ativação e de Dropout. As duas primeiras camadas densas possuem 1024 neurônios na camada oculta e a última possui 7 neurônios, que constituem a saída da rede. Foi utilizado um classificador Softmax, que é uma generalização do classificador de regressão logística binária, adaptado para múltiplas classes para categorizar os dados na saída

da rede. A categorização feita pela função Softmax normaliza a saída dos dados da rede em um intervalo de 0 e 1, configurando assim a distribuição da probabilidade para cada saída da rede. Para a correção dos pesos e retropropagação, foi utilizado a função de perda Cross-Entropy. Esta função calcula a distância entre duas distribuições de probabilidade, a primeira distribuição é a saída gerada pela função Softmax e a segunda distribuição é a matriz contendo a saída esperada da rede, utilizando para o ajuste da descida do gradiente.

Segundo Amin, Chase e Sinha (2017), o trabalho proposto apresentou bons resultados, alcançando a precisão média de 61,05% para a classificação das sete emoções. Analisando os resultados, os autores constataram que a emoção de alegria possui a maior taxa de reconhecimento. Porém o modelo não consegue classificar corretamente as emoções de medo e tristeza.

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são apresentadas as etapas do desenvolvimento do protótipo de reconhecimento em emoções em faces por meio de imagens fundamentado pelos conceitos apresentados no capítulo anterior. São abordados os requisitos do protótipo, seguidos de sua especificação, implementação e discussão dos resultados obtidos.

3.1 REQUISITOS

O Protótipo a ser desenvolvido atenderá os Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) apresentados no Quadro 7 e Quadro 8, respectivamente. Os requisitos são mapeados com os casos de uso ilustrados na Figura 25.

Quadro 7 - Requisitos funcionais

Requisitos funcionais (RF)	Casos de uso (UC)
RF01: realizar a detecção da face em uma imagem	UC01, UC04
RF02: criar uma rede neural profunda	UC03, UC05
RF03: utilizar a técnica de Aprendizado Profundo para o aprendizado de máquina	UC03
RF05: permitir que o usuário insira imagens no sistema	UC01, UC02
RF06: classificar a emoção através das expressões faciais	UC01, UC03, UC04, UC05

Fonte: elaborado pelo autor.

Quadro 8 - Requisitos não funcionais

Requisitos não funcionais (RNF)
RNF01: ser desenvolvido utilizando a linguagem Python
RNF02: utilizar Jupyter Notebook como ambiente de desenvolvimento
RNF03: utilizar o framework Keras para aplicação da Aprendizado Profundo
RNF04: utilizar frameworks OpenCV, Scikit Learn, Numpy, Matplotlib para computação científica e visualização dos dados

Fonte: elaborado pelo autor.

3.2 ESPECIFICAÇÃO

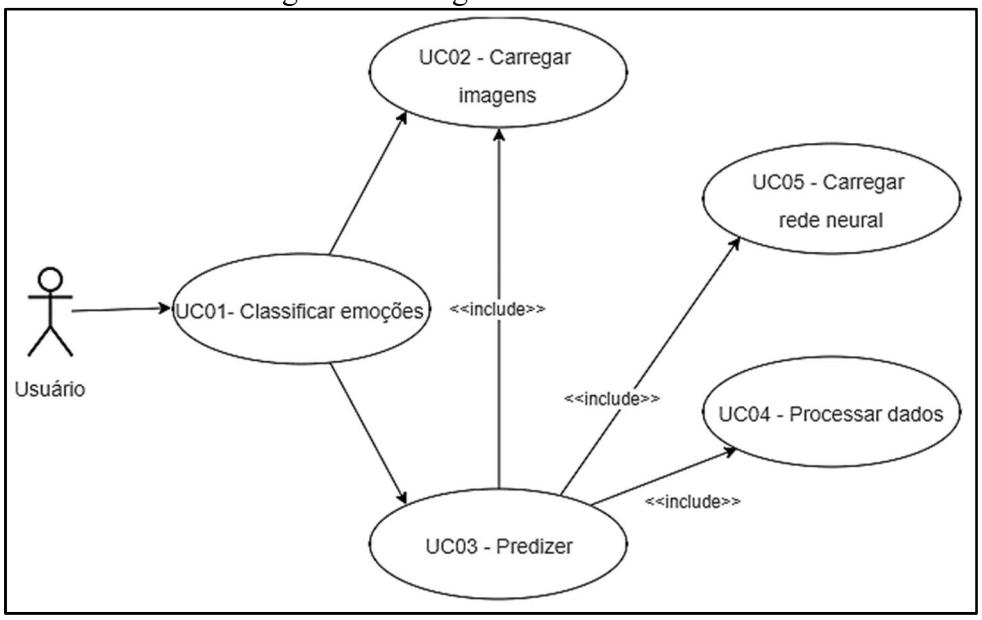
Nesta seção é descrito a especificação do protótipo DeepEmotive e suas funcionalidades. A especificação é composta por um diagrama de casos de uso e um diagrama de atividade. Foi utilizado a ferramenta Draw.io[4] para modelagem com base na Unified Modeling Language (UML).

[4] Ferramenta da empresa Google para criação e edição de diagramas de variados tipos, incluindo modelos UML, baseada em navegador web (DRAW.IO, 2017).

3.2.1 Diagrama de caso de uso

Nesta seção é apresentado um diagrama de caso de uso, ilustrado na Figura 25, demonstrando a utilização do protótipo pelo ator usuário.

Figura 25 - Diagrama de caso de uso



Fonte: elaborado pelo autor.

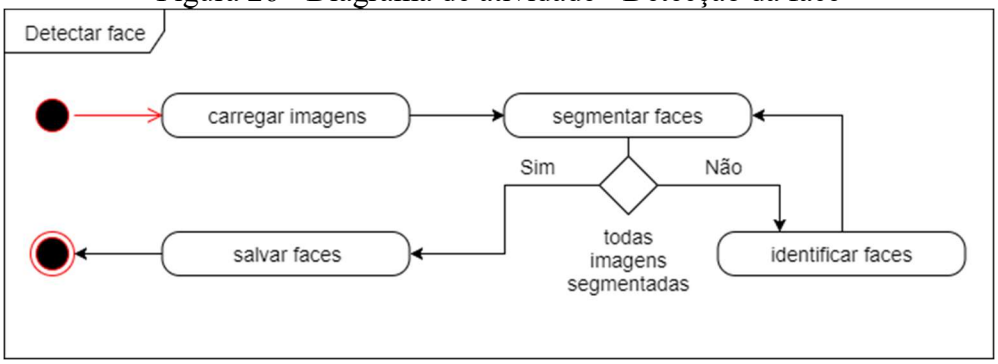
O caso de uso `UC01 - Classificar emoções` disponibiliza ao usuário a possibilidade de classificar automaticamente as emoções em imagens que possuem expressões faciais. Para tanto, é necessário a importação das imagens para o protótipo, ação realizada pelo caso de uso `UC02 - Carregar imagens`. O caso de uso `UC02` permite o usuário informar o caminho absoluto de uma imagem, ou informar o diretório das imagens, as quais deseja realizar a classificação. O caso de uso `UC03 - Predizer` informa ao usuário qual a emoção predita para a imagem, ou imagens, que foi especificado. Para que o caso de uso `UC03` realize sua função é necessário que a imagem, ou imagens, já estejam carregadas no protótipo. O prosseguimento do caso de uso `UC03` é dado pela execução da ação dos casos de uso `UC04 - Processar dados` e `UC05 - Carregar rede neural`. O caso de uso `UC04` realiza a etapa de formatação da estrutura dos dados da imagem, extraindo os pixels e estruturando na forma compatível com a entrada exigida pela rede neural profunda. O caso de uso `UC05` carrega a rede neural profunda e a configura com os pesos do melhor treinamento disponível.

3.2.2 Diagramas de atividades

Nesta seção é apresentado dois diagramas de atividades. A Figura 26 apresenta as atividades para detecção da face em uma imagem. Inicialmente a imagem, ou imagens, são carregadas para o protótipo através da funcionalidade `carregar imagens`, que é realizada

pelo caso de uso `UC02 – Carregar imagens`. A detecção da face ocorre na função `segmentar faces`, que tem a responsabilidade de instanciar o classificador da técnica Viola-Jones.

Figura 26 - Diagrama de atividade - Detecção da face



Fonte: elaborado pelo autor.

Em seguida, inicia a função auxiliar `identificar faces`, que, por sua vez, identifica a face através do classificador e retorna uma *bounding box*[5] com a localização espacial da face na imagem. Com a localização exata, a função `segmentar faces` separa a região da face da imagem original. Esta ação permanece executando até que todas as imagens sejam segmentadas. Por fim, as faces são salvas em um novo conjunto de dados pela função `salvar faces`, para uso do protótipo. A codificação da detecção da face é apresentada no Quadro 9.

[5] *Bounding box* é uma caixa retangular que envolve o objeto detectado. Pode ser formada por quatro ou oito pontos espaciais, dependendo se o objeto tem formato 2D ou 3D, respectivamente.

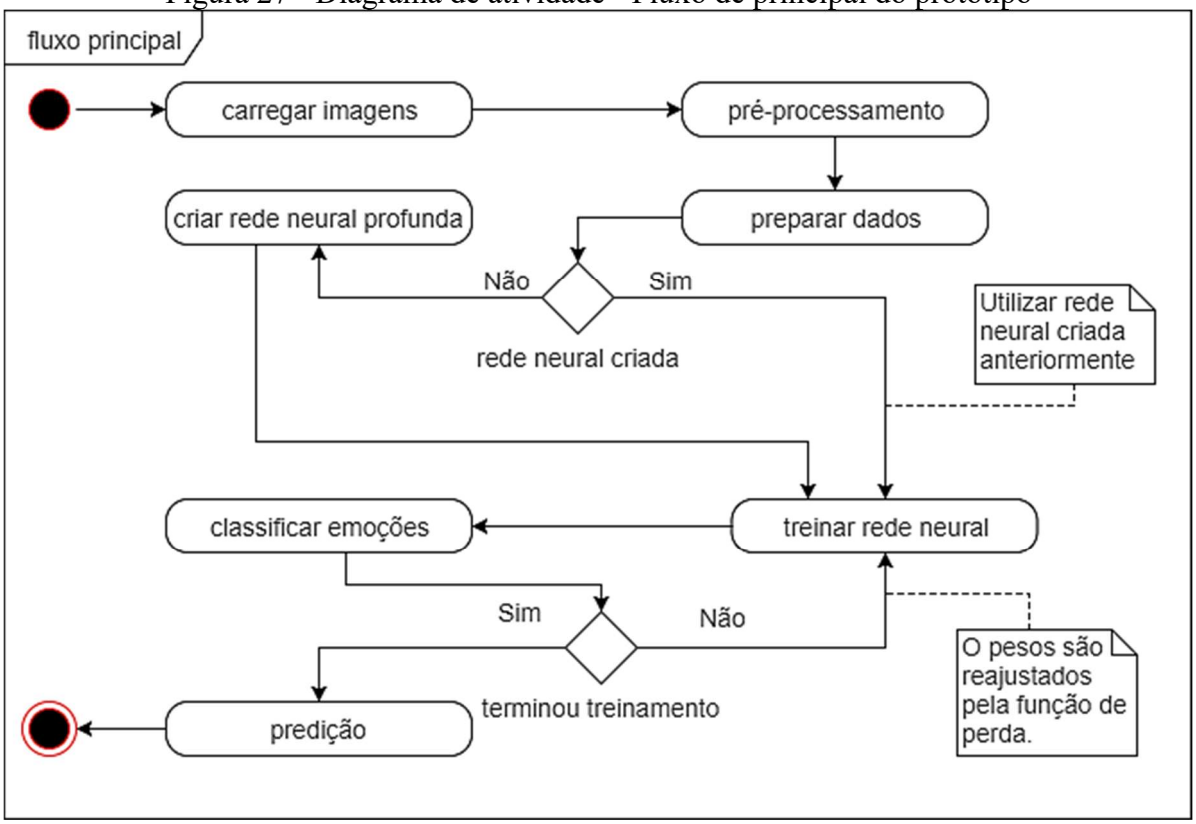
Quadro 9 - Codificação da segmentação da face

```python
def segmentar_faces(imagens, formato='.png'):
    """ Esta função é responsável por segmentar
        a face na imagem. Recebe como entrada uma
        lista de tuplas (imagens,emocao). Retornar a
        uma lista de tuplas (faces,emocao) extraídas de cada imagem.

        Args:
            imagens (list): lista de tuplas (imagens,emocao)

        Returns:
            list: uma lista de tuplas (faces,emocao).
    """
    faces = [] # lista para guardar as faces segmentadas
    face_cascade = _criar_classificador_face(cv2)  # classificador
    print('Segmentando faces, por favor aguarde...')
    for i, (imagem, emocao) in enumerate(imagens):

        box = _identificar_face(np.asarray(imagem),
                                face_cascade) # encontra face
        for (x, y, w, h) in box: # bounding box
            # recorta a região de interesse
            crop_im = imagem.crop((x, y, x+w, y+h)).resize((224,224))
            faces.append((crop_im, emocao)) # insere face an lista

    print('Segmentação de faces concluida com sucesso!')
    print('Quantidade de faces segmentadas: ',len(faces))
    return faces
```

Fonte: elaborado pelo autor.

A Figura 27 apresenta o diagrama de atividade do módulo principal do protótipo. A execução deste modulo parte do carregamento da imagem, ou imagens, para o protótipo. É a mesma funcionalidade desempenhada pelo caso de uso `UC02 - Carregar imagens` descrito anteriormente, no entanto, o caminho a ser informado como parâmetro do carregamento das imagens deve ser do novo conjunto de dados, conjunto que contém as faces segmentadas. Após as imagens estarem disponíveis, aplica-se a funcionalidade `pré-processamento`. Esta é uma função chave para protótipo, pois redimensiona cada imagem para uma dimensão pré-definida. A atividade de `pré-processamento` também aplica, em uma rotina interna, a técnica Data Augmentation, com a finalidade de aumentar o conjunto de dados original, porém sua utilização é opcional, sendo ativado através do *flag* `dataAugmentation`.

Figura 27 - Diagrama de atividade - Fluxo de principal do protótipo



Fonte: elaborado pelo autor.

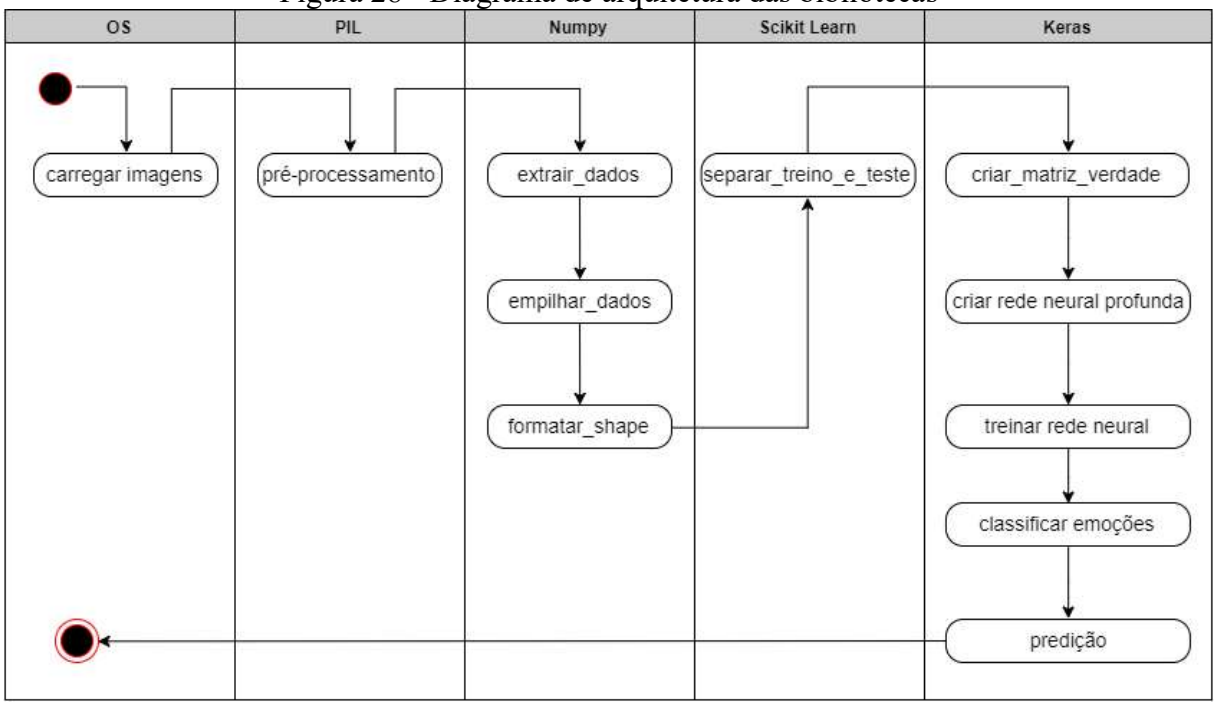
O passo seguinte é realizado pela função `preparar dados`, que estrutura o conjunto de dados para que seja possível utilizá-los como parâmetro do framework de aprendizado profundo. Esta função realiza as seguintes etapas: `extrair_dados`, que realiza a extração de todos os pixels de cada imagem para cálculos e manipulação; `empilhar_dados`, que cria transforma o conjunto de dados, imagens e seus respectivos descritores, em uma pilha de *tuple* de três posições de formato, quantidade, largura, altura; `separar_treino_e_teste`, que separa o conjunto de dados empilhados, de modo randômico, em quatro conjuntos distintos; `formatar_shape`, que adiciona a estrutura de cada conjunto a informação de quantos canais de profundidade serão trabalhados pelo *framework* de aprendizado profundo; `criar_matriz_verdade`, responsável por criar uma matriz verdade para todos os descritores. Todas estas etapas serão abordadas mais detalhadamente na Seção 3.3.

Em seguida, é realizada a verificação da necessidade de treinamento da rede, visto que é possível realizar a classificação em dois cenários: o primeiro que cria uma rede neural nova, e o segundo, que utiliza um modelo já criado anteriormente. Sequencialmente é executada a funcionalidade `treinar rede neural`, o qual, constantemente, identifica e extrai características relevantes para o aprendizado da rede. O treinamento implica na execução da funcionalidade de `classificar emoções`, que categoriza a saída neural em seus descritores,

fazendo a validação das emoções com base na matriz verdade criada anteriormente. A execução do treinamento acontece até um ciclo pré-definido no parâmetro da função. A cada ciclo de treinamento os pesos são reajustados de acordo com a função de perda.

Após o término do ciclo de treinamento, o protótipo está pronto para realizar a fase de predição, executada pela funcionalidade `predicão`. Esta etapa consiste em analisar as imagens de teste, isto é, aquelas que não foram utilizadas na fase de treinamento para serem classificadas de acordo com o conhecimento adquirido pelo algoritmo. Estas novas imagens podem ser inseridas pelo usuário através de um terminal de linha de comandos, ou podem ser imagens da mesma base de dados, separadas anteriormente. A Figura 28 apresenta o diagrama de bibliotecas utilizadas para desenvolvimento do fluxo principal do protótipo, demonstrando qual biblioteca é responsável pela aplicação de qual funcionalidade.

Figura 28 - Diagrama de arquitetura das bibliotecas



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

Neste capítulo são apresentadas as técnicas, ferramentas e a operacionalidade da implementação do protótipo. A seção 3.3.1 apresenta o detalhamento das ferramentas e as técnicas utilizadas. A seção 3.3.2 demonstra o processo operacional do protótipo.

3.3.1 Técnicas e ferramentas utilizadas

Para implementar o trabalho proposto utilizou-se o ambiente de desenvolvimento Jupyter Notebook versão 4.3.0, e a linguagem de programação Python, versão 3.6.2. O

Jupyter Notebook é baseado em uma aplicação web que possibilita a programação em células independentes, as quais podem ser formatadas como células de código, Markdown[6], e de visualização. O protótipo também utiliza as ferramentas: OpenCV, uma biblioteca de processamento de imagem desenvolvida em linguagem de programação C, porém atualmente possui suporte à linguagem Python; Scikit Learn, uma biblioteca com técnicas de aprendizado de máquina desenvolvida na linguagem Python e dispõe de várias funcionalidades, como classificação, regressão, entre outras; Numpy, uma biblioteca Python para manipulação de estruturas de dados numéricos; Matplotlib, uma biblioteca Python para visualização de dados.

O protótipo utiliza, como biblioteca principal, o *framework* Keras, biblioteca Python voltada a aprendizado de máquina profundo. Esta biblioteca implementa funcionalidades de *frameworks* de deep learning, já consolidados na comunidade, entre eles o TensorFlow[7], CNTK [8]e Theano[9]. As técnicas utilizadas no desenvolvimento do protótipo são apresentadas nas subseções a seguir, adotando a ordem dos casos de uso, contemplando todas as funcionalidades.

3.3.1.1 Base de dados

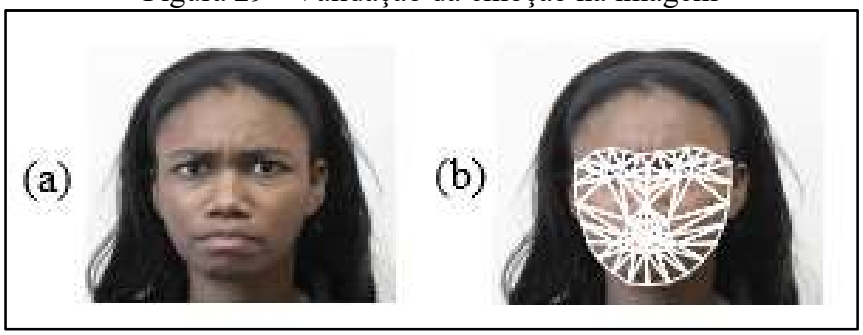
Para o desenvolvimento do protótipo, foi utilizado a base de dados Cohn-Kanade AU-Coded Expression Database Version 2 (LUCEY et. al. 2010), uma versão aprimorada da base de dados Cohn-Kanade AU-Coded Expression Database apresentada nos trabalhos correlatos, devido a ser voltada para o estudo de expressões faciais em imagens. Esta base de dados possui um total de 593 de imagens, totalizando 10,708 frames de um total de 123 indivíduos. Cada registro facial inicia em um frame apresentando a expressão facial neutral e vai até o frame apresentando a emoção. Esta versão da base de dados também realizou a validação da imagem com as AU's do sistema FACS. Um exemplo é apresentado na Figura 29, que ilustra a validação da emoção de raiva.

A escolha por esta base de dados foi ponderada por ter sido utilizada em dois dos três trabalhos correlatos apresentados, bem como por vários outros artigos científicos da área. Esta base de dados contém imagens classificadas para oito emoções, sendo elas: alegria, desgosto, desprezo, medo, neutral, raiva, surpresa e tristeza.

---

[6] Markdown é uma linguagem de marcação, converte seu texto em XHTML válido.
[7] TensorFlow é uma biblioteca de código aberto para aprendizado profundo de propriedade do Google.
[8] CNTK é uma biblioteca aprendizado profundo de propriedade da Microsoft.
[9] Theano é uma biblioteca de aprendizado de máquina, desenvolvida pela Uniservité Montréal no Canadá.

Figura 29 - Validação da emoção na imagem



Fonte: Lucey et al. (2010).

Contudo, para utilização desta base de dados fez-se necessário uma etapa de refinamento, visto que, segundo Lucey et al. (2010), nem todas as imagens tiveram a expressão facial validada pelas AU's. Para este protótipo foi utilizado apenas as imagens que passaram pelo processo de validação, mesmo que, se avaliada por um humano, apresentasse um grau elevado de confiança na classificação. Esta validação foi realizada pelo trabalho de Lucey et al. (2010), os autores compararam cada imagem com os pontos faciais que configuravam positivamente uma AU. Os autores disponibilizam, em conjunto com a base de dados, um arquivo informando quais imagens apresentam pontos válidos. No contexto deste trabalho, para cada indivíduo foi selecionado o primeiro frame, para ser classificado como emoção neutra, e, o último frame, classificando a emoção validada por Lucey et al. (2010), totalizando 453 imagens válidas. Cada imagem da base de dados possui as dimensões 640x490 pixels de largura e altura respectivamente.

3.3.1.2 Carregar imagens para o protótipo

Nesta etapa é realizada a importação das imagens para uso do protótipo. O carregamento é feito pelo método `carregar_imagens(path)`, ilustrado no Quadro 10. O parâmetro da função é o caminho do diretório onde as imagens estão localizadas no disco. A Figura 30 ilustra a estrutura do diretório das imagens.

Figura 30 -Estrutura do diretório de imagens

Fonte: elaborado pelo autor.

Quando uma imagem é carregada para o sistema, também é gerado um descritor para a emoção apresentada. Este descritor é gerado com base no diretório que a imagem está situada. A Tabela 1 ilustra o descritor atribuído para cada emoção. Ao finalizar a função de carregamento das imagens, a função `carregar_imagens(path)` retorna uma estrutura de dados Python, chamada de *Tuple*, contendo uma lista de imagens e uma lista de descritores.

Tabela 1 - Descritores de emoção das expressões faciais de emoção

Emoção	Descritor
Alegria	0
Desgosto	1
Desprezo	2
Medo	3
Neutra	4
Raiva	5
Surpresa	6
Tristeza	7

Fonte: elaborado pelo autor.

Quadro 10 - Função de carregamento das imagens para o protótipo

```
def carregar_imagens(path):
    """ Esta função é responsável por importar as imagens para
        o protótipo. Recebe caminho do diretório onde as imagens
        estão localizadas. Retorna uma lista de tuplas de duas
        posições. A primeira posição é o objeto de imagem e a
        segunda é a emoção da imagem.

        Args:
            path (string): caminho do diretório onde as imagens
            estão localizadas.
        Returns:
            tuple (len=2): tupla contendo objeto imagem e emoção.
    """
    imagens = [] # lista para guardar as faces segmentadas
    classes_emocoes = os.listdir(path)
    # para cada categoria(emoção)
    for emocao in range(len(classes_emocoes)):

    m_path = os.path.join(path, classes_emocoes[emocao])  # path src
        # pega todos os arquivos da emocao
        files = [f for f in os.listdir(m_path)]

        print('Carregando imagens de {}, qtd: {}...'.format(
            classes_emocoes[emocao], len(files)))
        for file in files:  # para cada imagem

            path_img = m_path + '\\' + file #caminho absoluto
            im = Image.open(path_img) # obj PIL
            imagens.append((im, emocao)) # insere face an lista

    print('Imagens carregadas com sucesso!')
    print('Quantidade de imagens: ',len(imagens))
    return imagens
```

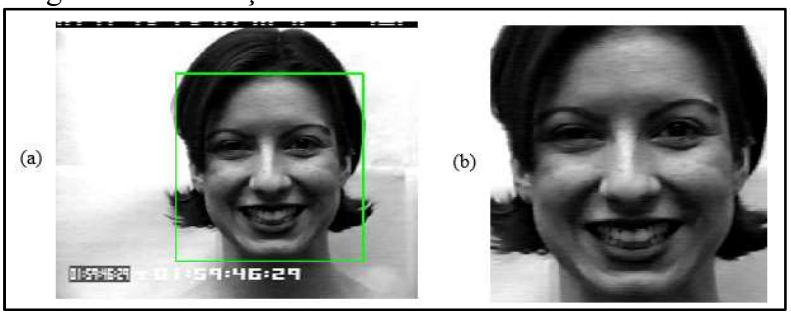
Fonte: elaborado pelo autor.

3.3.1.3 Detecção da face

Esta seção descreve a implementação da atividade `detectar face`, introduzida na seção 3.2. Para detecção da face foi utilizado a biblioteca OpenCV versão 3.3.0, que implementa o algoritmo de Viola e Jones (2001) descrito na seção 2.3.3. Para a identificação da região onde a face está localizada, é realizada através da função `detectMultiScale`, que é responsável pela aplicação núcleos, cálculo das regiões retangulares, e seu retorno é uma *bounding box* que delimita a região de interesse. A região é então classificada como sendo uma face, através da função `CascadeClassifier`, que é a implementação do classificador em cascata. A Figura 31 ilustra a detecção da face em uma imagem, realizada pela ferramenta descrita. Observa-se na imagem que, primeiramente é identificada a área de interesse (a) e

posteriormente é segmentada a região de interesse (b). As faces encontradas na imagem possuem a dimensão de 224 de largura por 224 de altura. Estas novas imagens são armazenadas em um novo conjunto de dados específico para faces.

Figura 31 - Detecção da face utilizado técnica Viola-Jones

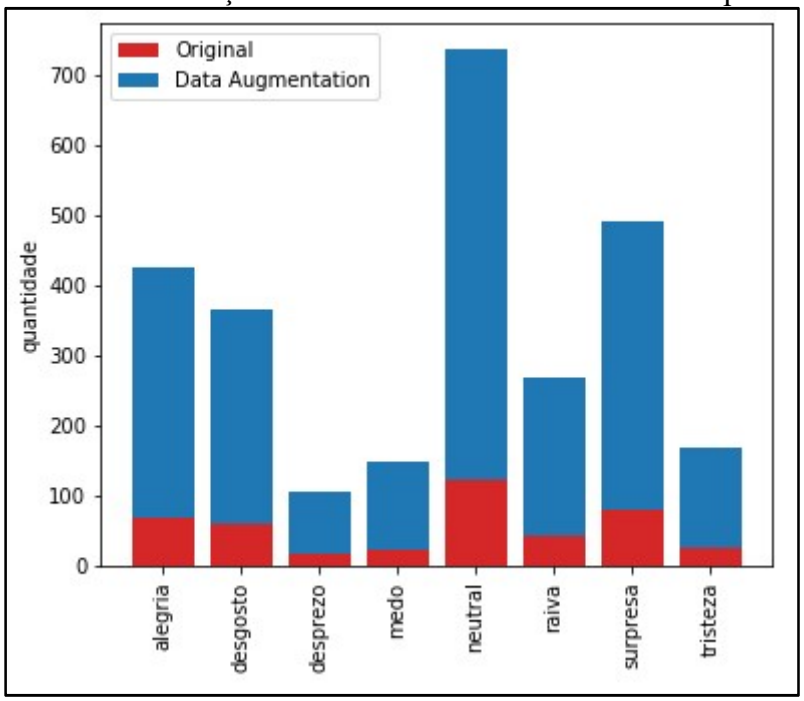


Fonte: elaborado pelo autor.

3.3.1.4 Pré-processamento

A fase de pré-processamento é responsável pela aplicação de transformações nas imagens importadas para o protótipo, redimensionando cada imagem para uma dimensão pré-definida de 98 pixels largura e 98 pixels de altura. Para esta fase existe a possibilidade de ser aplicada a técnica Data Augmentation, ativada através do *flag* `dataAugmentation`, com a finalidade de aumentar o conjunto de dados original, porém sua utilização é opcional. A distribuição das amostras da base de dados pode ser observada na Figura 32.

Figura 32 – Distribuição das amostras da base dados do DeepEmotive



Fonte: elaborado pelo autor.

As amostras originais estão distribuídas em:71 imagens de alegria; 61 de desgosto; 18 de desprezo; 25 de medo; 123 de neutra; 45 de raiva; 82 de surpresa; e 28 de tristeza. Após a

aplicação da técnica de Data Augmentation, a distribuição das amostras ficou em: 426 imagens de alegria; 366 de desgosto; 108 de desprezo; 150 de medo; 738 de neutra; 270 de raiva; 492 de surpresa; e 168 de tristeza.

A aplicação da técnica de Data Augmentation foi baseada no trabalho de Wang (2016), na qual foi utilizada a mesma base de dados para este trabalho. Devido ao conjunto de dados ser relativamente pequeno, no contexto do aprendizado profundo, a técnica auxiliou a criar novas amostras para o aprendizado da rede. A utilização da técnica cria variações nas imagens, aplicando duas transformações, a primeira de rotação em -30°, -15°, 15° e 30° graus, e a segunda de correção de brilho, em cada imagem. A utilização da técnica aumenta a base de dados em 600%, totalizando 2718 imagens. As imagens desta base de dados possuem três canais de cores, RGB (Red, Green, Blue – vermelho, verde e azul). Porém, as imagens são convertidas em escala de cinza para o protótipo proposto, devido ao custo computacional.

3.3.1.5 Preparar dados para treinamento

A etapa `preparar dados`, é responsável por preparar o conjunto de dados para o treinamento, para que seja possível sua utilização pelo framework de aprendizado profundo. Os códigos da implementação das funções auxiliares da fase de `preparar dados` podem ser vistos em maiores detalhes no apêndice A. A primeira função a ser executada é `extrair_dados`, que recebe uma lista de objetos de imagem Python e extrai todos os pixels de cada imagem. Para tanto, essa funcionalidade faz uso de uma função auxiliar denominada `_get_pixels_imagem(imagem)`, que é o cerne da funcionalidade. A extração é ilustrada no Quadro 11. A função auxiliar transforma os pixels da imagem passada como parâmetro em uma lista de pixels. Após a criação da lista, esta é convertida para a *array* de dados Numpy. Em seguida o *array* é redimensionado para um formato pré-definido para trabalhar com o aprendizado profundo, que no contexto deste trabalho, cada imagem possui dimensão de 98x98 pixels de largura e altura.

Quadro 11 - Função para extração dos pixels de uma imagem

```
def _get_pixels_imagem(imagem, shape=(img_cols,img_rows)):
    """ Esta função é responsável por extrair os pixels
        da imagem. Recebe como entrada uma objeto de imagem
        python e retorna um numpy array contendo uma
        lista de lista de todos os pixels. Pode ser
        alterado a dimensão da imagem pelo parametro shape.

        Args:
            imagem (PIL): objeto de imagem python.
            shape (tuple): tupla contendo o dimensão
            da imagem de saida.

        Returns:
            array (np.ndarray): retorna um array numpy.
    """
    # cria uma array numpy com uma lista contendo
    # todos os pixels da imagem
    pixel_values = np.array(list(imagem.getdata()))

    # redimensiona os pixels para formato 98x98
    pixel_values = pixel_values.reshape(shape)
    return pixel_values
```

Fonte: elaborado pelo autor.

A segunda função da etapa de preparação é `empilhar_dados`, que transforma a estrutura do conjunto de dados, imagens e seus respectivos descritores, em uma pilha de *tuple* de três posições, no formato quantidade, largura, altura. Esta transformação é necessária devido a especificação de trabalho do *framework* Keras. Esta transformação é necessária devido a especificação de trabalho do *framework* Keras. A terceira função da etapa é `separar_treino_e_teste(imagens, descritores)`, que separa o conjunto de dados empilhados, de modo randômico, em quatro conjuntos distintos: imagens de treinamento, imagens de teste, descritores de treinamento e descritores de teste. Esta função recebe o conjunto de dados com todos os pixels da imagem e o conjunto de dados com todos os descritores e os separa em dois novos conjuntos, denominados de treino e teste. Para realizar esta separação é utilizado uma função auxiliar da biblioteca Scikit Learn denominada `train_test_split(x,y, test_size, random_state)`, esta, por sua vez separa o conjunto de dados de acordo taxa especificada pelo parâmetro `test_size`. Para o trabalho proposto, a taxa de separação escolhida foi de 80% para treinamento e 20% para teste. A escolha de cada imagem a ser separada é determinada por um fator randômico, definido pelo parâmetro `random_state`. Este parâmetro serve como semente geradora (*seed*) do algoritmo de randomização.

A quarta função desta etapa é `formatar_shape`, que formata o conjunto de imagens de treinamento e de teste, adicionando a quantidade de camadas de profundidade a serem computadas pelo aprendizado profundo no Keras. A formatação dos dados é definida de acordo com framework de background a ser utilizado. No trabalho proposto foi utilizado o TensorFlow como ferramenta de background, então, a indicação do canal de cor será incluída na estrutura de dados após a informação da dimensão da imagem, e. g. quantidade, largura, altura, canais. Nesta etapa os pixels salvos anteriormente como tipo inteiro, variando de 0 a 255, são convertidos para o tipo de ponto flutuante (*float*) e normalizados.

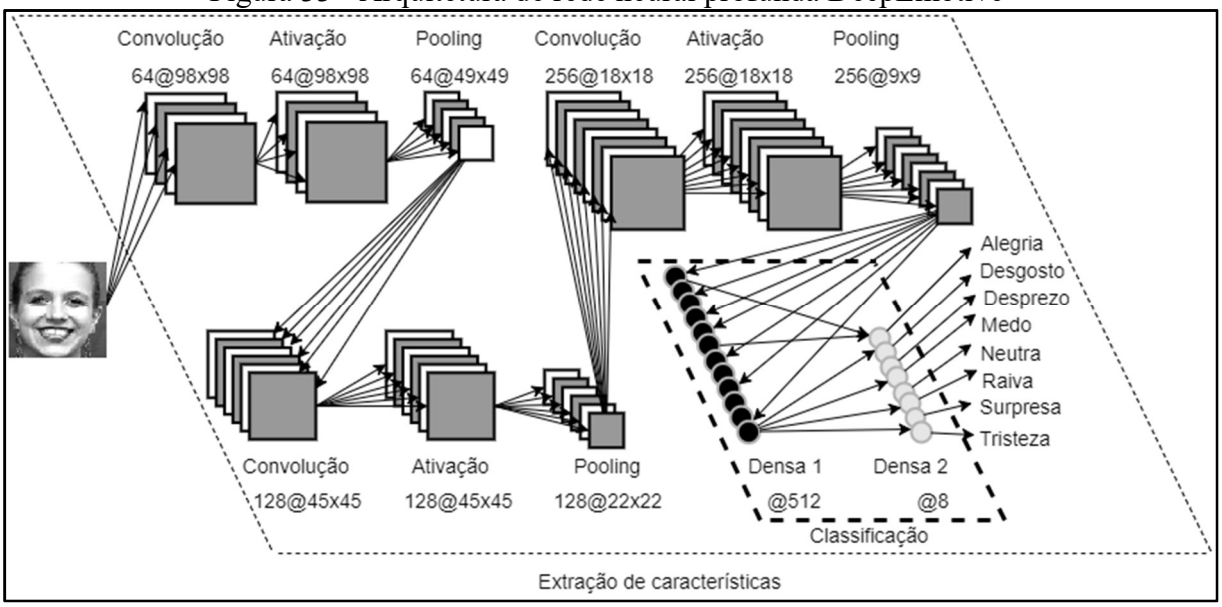
A quinta e última função desempenhada pela etapa da de preparação de dados é a função `criar_matriz_verdade(y_train, y_test)`, responsável por uma matriz verdade, também conhecida como *one-hot*. São utilizadas para otimizar as decisões booleanas. A função recebe como parâmetro os conjuntos de descritores de emoção, treinamento e teste, e aplica a funcionalidade da biblioteca Keras denominada `np_utils.to_categorical`. Esta cria uma matriz de comprimento do conjunto de dados e de largura da quantidade de descritores únicos encontrado no conjunto de dados. Cada linha da matriz possui o *flag* ativo na posição que corresponde o descritor, e. g. a linha [1 0 0 0 0 0 0 0] corresponderia ao descritor 0 (alegria), e a linha [0 0 0 0 0 0 0 1] corresponderia ao descritor 7 (tristeza). No contexto do trabalho proposto, o processamento desta função gerou uma matriz de dimensões (2174, 8).

3.3.1.6 Criar rede neural profunda

Nesta etapa do desenvolvimento é criado a arquitetura da rede neural utilizando as técnicas de aprendizado profundo. Conforme citado anteriormente, a construção de uma rede neural profunda, está sobre a temática da organização hierárquica, a fim de encontrar o melhor modelo que se ajuste ao conjunto de dados da abordagem a ser desempenhada. Para este trabalho, será utilizado o modelo de redes neurais convolucionais.

A rede neural profunda, utilizada neste protótipo, foi definida após inúmeras tentativas descritas na seção de experimentos, e possui a arquitetura ilustrada na Figura 33. A rede é construída pela funcionalidade `criar_deep_cnn_10(num_classes, shape)`, conforme pode ser visto no Quadro 12. Esta função cria as camadas da rede, definindo sua hierarquia. A função recebe dois parâmetros de entrada: o primeiro é `num_classes`, que indica quantidade de categorias que serão atribuídas ao final da rede utilizadas para a classificação dos dados; o segundo parâmetro `shape`, é responsável por parametrizar a entrada da rede neural, este vincula ao processo realizado na etapa de preparar dados.

Figura 33 - Arquitetura de rede neural profunda DeepEmotive



Fonte: elaborado pelo autor.

Inicialmente são definidos alguns parâmetros que serão utilizados pela rede: quantidade de núcleos para cada bloco convolucional; dimensão dos campos receptivos locais; taxa de queda da rede (Dropout) para prevenir a sobreposição (*overfitting*); quantidade de neurônios da primeira camada densa; dimensão dos campos receptivos para camada de subamostragem.

Em seguida é instanciado o objeto `model`, da classe `Sequencial`, disponibilizada pela biblioteca Keras. Este objeto define que a computação dos dados na rede neural será realizada de forma sequencial, ou seja, a saída de primeira camada servirá como parâmetro de entrada para a camada subsequente. A rede possui dois blocos macros: extração de características, que é subdividido em três blocos convolucionais; e classificação, constituído por duas camadas densas.

Cada bloco convolucional é intercalado com uma camada de convolução, uma camada de ativação, e uma camada de *pooling*. No primeiro bloco, a camada de convolução é criada a partir da função `model.add(Conv2D(nucleos_conv1, campos_receptivos_nucleos, input_shape=shape, padding='same', name='Conv1'))`. A função recebe como parâmetro: a quantidade de núcleos que a camada terá, representando a quantidade de mapas de características; a dimensão dos núcleos, que define a dimensão dos campos receptivos de cada mapa; o formato dos dados, abordado anteriormente na etapa prepara dados; o *padding*, que define que a camada em questão irá gerar uma saída com mesmo tamanho da imagem original; e por fim um rótulo para identificar a camada. A camada de ativação pela função `model.add(PReLU(name='PRelu1'))` recebe como parâmetro somente o rotulo identificador

da camada. Após a camada de ativação, é criada a camada de subamostragem, através do método `model.add(AveragePooling2D(pool_size=(campos_receptivos_pooling), name='Pooling1'))`. Esta camada cria os novos mapas de características, agrupando os dados de saída da camada pelo valor médio resultante da convolução, diferentemente da técnica *max pooling*, que agrupa os dados pelo maior valor resultante. A camada de subamostragem recebe como parâmetro a dimensão dos novos campos receptivos de pooling, definidos como 2x2, e o nome para identificação na rede.

Quadro 12 - Criar rede neural profunda

```
def criar_deep_cnn_10(num_classes, shape):
    nucleos_conv1 = 64
    nucleos_conv2 = 128
    nucleos_conv3 = 256
    dropout_1 = 0.25
    dropout_2 = 0.5
    neuronios_camada_densa = 512
    campos_receptivos_nucleos = (5, 5)
    campos_receptivos_pooling = (2, 2)
    model = Sequential()
    # bloco convolucional 1
    model.add(Conv2D(nucleos_conv1, campos_receptivos_nucleos,
                     input_shape=shape, padding='same', name='Conv1'))
    model.add(PReLU(name='PRelu1'))
    model.add(AveragePooling2D(pool_size=(campos_receptivos_pooling), name='Pooling1'))
    # bloco convolucional 2
    model.add(Conv2D(nucleos_conv2, campos_receptivos_nucleos, name='Conv2'))
    model.add(PReLU(name='PRelu2'))
    model.add(AveragePooling2D(pool_size=campos_receptivos_pooling, name='Pooling2'))
    # bloco convolucional 3
    model.add(Conv2D(nucleos_conv3, campos_receptivos_nucleos, name='Conv3'))
    model.add(PReLU(name='PRelu3'))
    model.add(AveragePooling2D(pool_size=campos_receptivos_pooling, name='Pooling3'))
    # bloco interligação
    model.add(Dropout(dropout_1, name='Dropout1'))
    model.add(Flatten(name='Achatamento'))
    # bloco denso 1
    model.add(Dense(neuronios_camada_densa, name='Densa1'))
    model.add(PReLU(name='PRelu4'))
    # bloco denso 2
    model.add(Dropout(dropout_2, name='Dropout2'))
    model.add(Dense(num_classes, activation='softmax', name='predicao'))
    # ajuste fino e funcão de perda
    adam = optimizers.Adam(lr=1e-3, decay=1e-5)
    model.compile(loss = 'categorical_crossentropy',
                  optimizer = adam, metrics = ['accuracy'])
    return model
```

Fonte: elaborado pelo autor.

A computação da camada de subamostragem gera na saída da rede, uma imagem com redução da dimensão em 50%, tornando assim a camada de nível superior mais abstrata. Os dois blocos convolucionais seguintes, são criados semelhantemente a descrição acima, alterando a quantidade de núcleos para 128 e 256, para o bloco 2 e 3, respectivamente.

Após a criação dos blocos convolucionais, é criado um bloco intermediário de ligação, que adiciona uma camada de `Dropout` com taxa de queda de 25%, e uma camada `Flatten`, que realiza o achatamento dos dados. Este achatamento consiste em converter as matrizes de pixels da convolução em um vetor unidimensional, no qual cada pixel servirá como uma entrada para a primeira camada densa.

Em seguida, gera-se a classificação da rede, construindo a primeira camada densa com neurônios totalmente conectados. Esta camada possui 512 neurônios na camada oculta da rede para cada pixel. Em sequência, é adicionado uma nova camada de ativação. Após o processamento da primeira camada densa, é adicionado uma nova camada de `Dropout` com taxa de queda de 50%, tornando ainda maior a generalização das saídas da rede. A segunda camada densa é a saída da rede, na qual é realizada a classificação da emoção. Esta camada recebe pelo parâmetro `num_classes` a quantidade de neurônio que irão compor a saída da rede, i.e., a quantidade de emoções que deseja classificar. No contexto deste trabalho a saída da rede é a representação das oito emoções: alegria, desgosto, desprezo, medo, neutral, raiva, surpresa e tristeza. A ativação dos neurônios desta camada densa se dá pela função logística de ativação Softmax.

Ao final, é definido como será feito a correção dos pesos para o aprendizado da rede. A função de perda, escolhida para o aprendizado da rede, é `categorical_crossentropy`. Esta etapa também consiste em manipular os hiperparâmetros do gradiente de aprendizado para encontrar possíveis melhorias no aprendizado da rede. No trabalho proposto trabalho foi ajustado a taxa de aprendizado (*learning rate*) para $1e-3$ e ajustar o passo da descida do gradiente (*decay*) em $1e-5$. Por fim, é compilado o objeto `model` e retornado. Neste ponto a rede neural profunda está pronta para realizar o treinamento.

3.3.1.7 Treinamento da rede neural profunda

Nesta seção é apresentado a etapa de treinamento da rede neural profunda, conforme apresentada no Quadro 14 é responsável por executar as funcionalidades de ajuste dos pesos da rede neural profunda. Cada imagem do conjunto de dados de treino é convoluída através da rede neutral profunda. É realizado a extração das características em cada bloco de convolução e atualizado os pesos do treinamento. O Quadro 13 apresenta a quantidade de paramentos que a rede treina para cada imagem é de 12.604.808. Com isto, identifica-se o alto custo computacional para o processamento do conjunto de dados, visto que a quantidade total de parâmetros a serem treinados para todo o conjunto de dados é de 34.259.868.144.

Quadro 13 - Quantidade de parâmetros treinados para uma imagem

```
=================================
Total params: 12,604,808
Trainable params: 12,604,808
Non-trainable params: 0
_________________________________
```

Fonte: elaborado pelo autor.

O método `treinar_rede_deep_cnn` recebe como parâmetro: o `model`, instância do objeto da rede neural criada anteriormente; `X_train`, conjunto de dados das imagens de treinamento; `X_teste`, conjunto de dados das imagens de treino; `Y_train`, matriz verdade (*on-hot*) dos descritores de treinamento; `Y_test`, matriz verdade (*on-hot*) dos descritores de teste; `epocas`, quantidade de iterações de treinamento da rede neural; e o `batch`, que define o tamanho do lote de treinamento. Os dados são repassados para a função `fit`, da biblioteca Keras, a qual, realiza de fato o treinamento da rede.

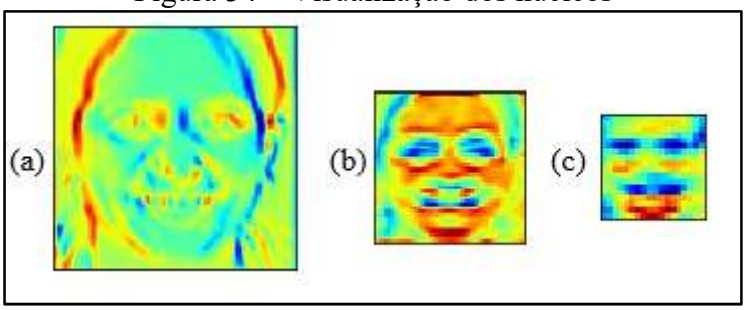
Quadro 14 - Treinar rede convolucional profunda

```
1  def treinar_rede_deep_cnn(model, X_train, Y_train,
2                            X_test, Y_test epocas=20, batch=99):
3      """ Esta função é responsável por realizar o treinamento
4          da rede neural convolucional profunda.
5          Recebe como parâmetro o modelo da rede, a quantidade de
6          epocas de treinamento, e o tamanho do lote de treinamento.
7
8          Args:
9              model (sequencial): modelo da rede neural
10             X_train (array): imagens de treino.
11             Y_train (array): descritores de treino.
12             X_teste (array): imagens de teste.
13             Y_teste (array): descritores de teste.
14             epocas (int): quantidade de epocas de treinamento
15             batch (int): tamanho do lote de treinamento.
16
17         Returns:
18             history: log's do aprendizado da rede
19             model: rede neural profunda completa.
20     """
21     print('Treinando a rede neural profunda...')
22     # % time: contador do tempo de processamento
23     %time history = model.fit(X_train,  # imagens treino
24                               Y_train,  # descritores treino
25                               validation_data=(X_test, Y_test),
26                               epochs= epocas,
27                               batch_size= batch,
28                               verbose=1)
29     print('Treinanmeto finalizado!')
30     return history, model
```

Fonte: elaborado pelo autor.

A cada iteração de treinamento os pesos são validados no conjunto de dados de teste, matrizes *on-hot,* e realizado a correção pela função de perda. Ao final, é retornado o *log* de aprendizado da rede e o modelo treinado, contendo as métricas do aprendizado. A arquitetura do modelo treinado, assim como os pesos da melhor iteração de treinamento, é memorizada e salva em disco, para que seja possível uma posterior utilização, sem a necessidade da aplicação do treinamento. Durante a fase de treinamento também foi realizado uma etapa conhecida como visualização das camadas intermediárias, que consiste em visualizar como os neurônios de cada mapa de característica estão sendo ativados. Isto é realizado através da visualização da saída de cada camada convolucional da rede neural profunda. A Figura 34 ilustra um mapa de características para cada camada da rede do DeepEmotive, sendo (a) a primeira camada de convolução com 64 núcleos, (b) a segunda com 128 núcleos, e (c) a última camada com 256 núcleos. Todos os núcleos podem ser visualizados com mais detalhes no apêndice B.

Figura 34 – Visualização dos núcleos



Fonte: elaborado pelo autor.

3.3.1.8 Predição

A predição é a etapa final do processamento de execução do protótipo, e é responsável por apresentar a classificação de uma ou mais imagens, baseado no aprendizado adquirido pela rede, na fase de treinamento. Esta funcionalidade é executada pela função `model.predict(X, batch_size=32, verbose=1)`, função da biblioteca Keras, apresentada no Quadro 15. A função recebe como primeiro parâmetro os dados da imagem ou imagens na qual se deseja predizer a emoção. Também é setado o tamanho do lote de processamento e o *flag de log* do processamento de predição. O retorno da função é uma matriz com a probabilidade de cada emoção predita para o conjunto de dados passados como entrada. Cada linha da matriz corresponde a uma imagem analisada, e as colunas são as emoções preditas. Para cada imagem é atribuído um valor de probabilidade relativo a uma das oito emoções, distribuído num intervalo entre [0, 1]. A predição é dada pelo maior argumento, ou seja, caso a linha possua os valores x=[0.8, 0, 0, 0, 0, 0, 0, 0.2], então a predição dada para esta imagem

seria a emoção de alegria, pois possui 80% de confiança para ser alegria, e apenas 20% para tristeza.

Quadro 15 - Predição da emoção

```
def predizer(model):
    """Esta função é responsável por realizar a predição
       do conjunto de dados. Recebe como parametro o
       modelo da rede, e retorna uma tupla contendo a
       matriz de probabilidade e a matriz da classificação.

       Args:
           model (sequencial): modelo da rede neural
           treinado.
       Returns:
           np.array: matriz de probabilidade
           np.array: matriz da classificação
    """
    print('Predição da(s) imagem(ns)...\n')
    m_probabilidade = model.predict(X_test, batch_size=32, verbose=1)
    m_predicao = [np.argmax(prob) for prob in m_probabilidade]

    print('Predição completa!')
    return m_probabilidade, m_predicao
```

Fonte: elaborado pelo autor.

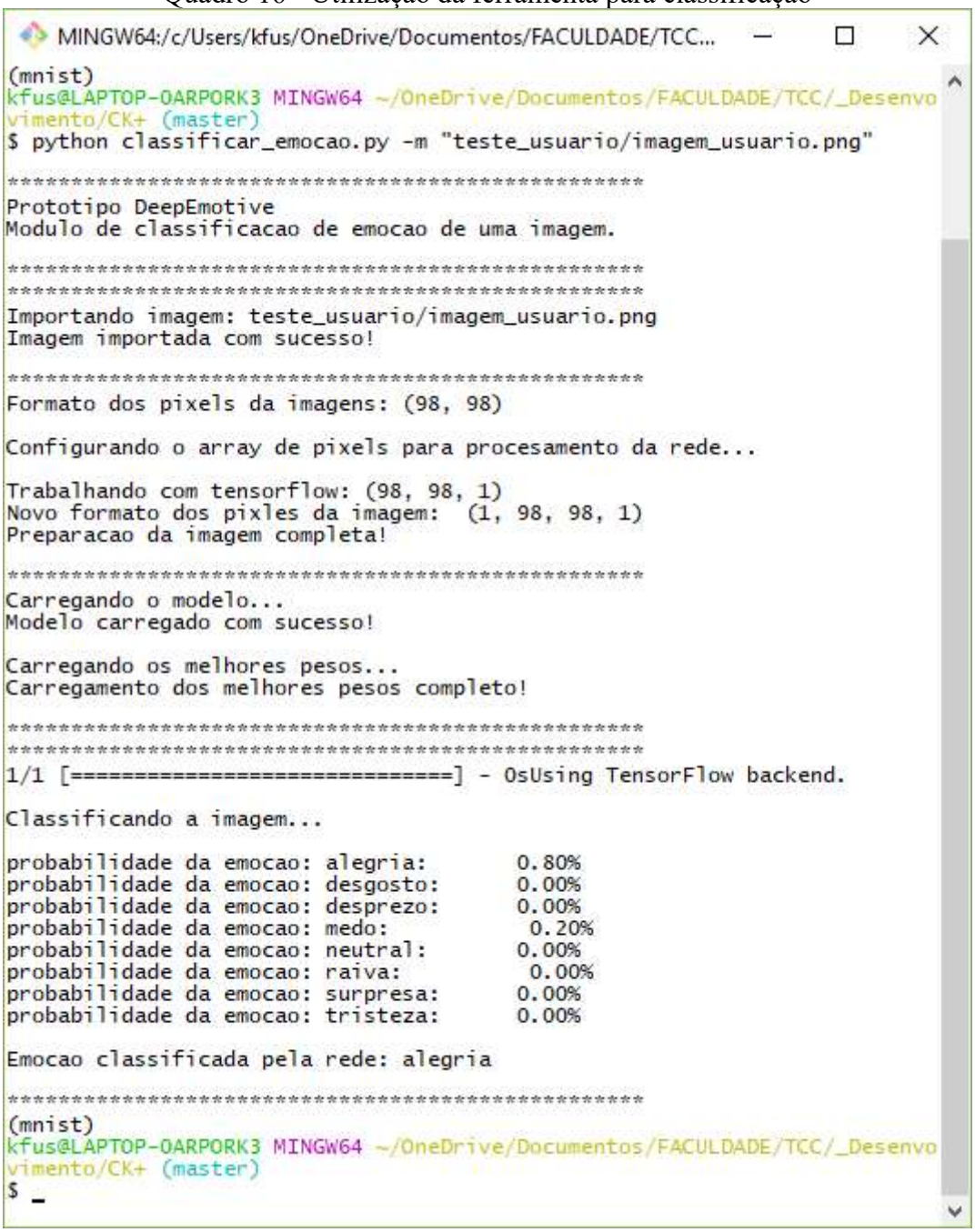
3.3.2 Operacionalidade da implementação

Nesta seção é demonstrado a operacionalidade da implementação do protótipo, abordando o caso de uso `UC1 - Classificar emoções`. O usuário pode testar a classificação de uma ou mais imagens. Para tanto, é necessário que modelo já esteja treinado. Esta funcionalidade pode ser observada no Quadro 16.

O modelo é disponibilizado através do GitHub[10], juntamente com a configuração dos melhores pesos encontrados para essa rede. Para a correta execução do protótipo, é necessário que o ambiente de execução do usuário possua todas as dependências do protótipo. As dependências estão relacionadas no arquivo `requirements.txt`, disponível juntamente com os arquivos de download. A instalação é feita através de um gerenciador de pacotes, como por exemplo Pip ou Conda.

[10] Este protótipo pode ser acessado pela url: https://github.com/diogenesd/deepemotive

Quadro 16 - Utilização da ferramenta para classificação



Fonte: elaborado pelo autor.

O usuário deve realizar o download dos arquivos e, através de um terminal de comandos, navegar até o diretório onde os arquivos foram salvos. Em seguida deve executar o script `classificar_emocao.py` para predizer uma ou mais imagens. Conforme pode ser visto no Quadro 17, a execução do script irá recriar o modelo e configurá-lo com os pesos já treinados. Logo após a configuração (*setup*) do modelo é feita a predição, na qual as imagens são passadas através do classificador da rede. Ao final é apresentado a emoção classificada para cada imagem e a probabilidade para cada emoção, conforme apresentado na parte inferior do Quadro 16. Este script possui dois parâmetros de entrada: `-m`

<CAMINHO_DA_IMAGEM>, que se refere ao caminho absoluto da imagem a ser classificada; e -d <CAMINHO_DO_DIRETORIO>, referente ao diretório contendo todas as imagens a serem preditas. Também foi criado um parâmetro -h (*help*) que auxilia o usuário na operacionalidade da predição.

Quadro 17 - Configuração do modelo treinado para fazer a predição

```
178     print('*' * 50)
179     # Carregando o modelo salvo
180     # Lendo o modelo salvo em arquivo para um novo modelo
181     MODELO_SALVO_JSON = 'deep_emotive_model_tl0_faces.json'
182     print("Carregando o modelo...")
183     json_file = open(MODELO_SALVO_JSON, 'r')
184     loaded_model_json = json_file.read()
185     json_file.close()
186     loaded_model = model_from_json(loaded_model_json)
187     print('Modelo carregado com sucesso!\n')
188
189     MELHORES_PESOS = 'pesos_teste_10-266-0.96.3.hdf5'
190     # Carregando o melhor pesos do checkout para para o novo modelo
191     print("Carregando os melhores pesos...")
192     loaded_model.load_weights(MELHORES_PESOS)
193     # Compilando o modelo que estava salvo
194     # lr: taxa de aprendizado
195     # decay: tamanho do passo da caída do gradiente
196     adam = optimizers.Adam(lr=1e-3, decay=1e-5)
197     # Copilando o modelo
198     # métrica de reconhecimento será precisão.
199     loaded_model.compile(loss = 'categorical_crossentropy',
200                          optimizer = adam, metrics = ['accuracy'])
201     print("Carregamento dos melhores pesos completo!\n")
202     print('*' * 50)
```

Fonte: elaborado pelo autor.

3.4 ANÁLISE DOS RESULTADOS

Neste capítulo é realizado a análise dos resultados obtidos com os testes elaborados para a rede DeepEmotive. Para validar o modelo de rede neural profunda desenvolvida, foram realizados testes sobre a base de dados Cohn-Kanade AU-Coded Expression Database Version 2 de Lucey et. al., (2010). O recurso tecnológico utilizado para o desenvolvimento deste protótipo foi um notebook com processador Intel core i7, 16GB de memória RAM e GPU Nvidia Geforce 920M de 2GB de memória dedica.

A seção 3.4.1 apresenta os resultados obtidos nos testes de definição da arquitetura de rede neural profunda. A seção 3.4.2 apresenta os resultados obtidos aplicando a técnica de transferência de aprendizado. Por fim, a seção 3.4.3 compara os resultados obtidos pelo protótipo com os trabalhos correlatos.

3.4.1 Teste de arquitetura da rede neural profunda

Para testar a arquitetura da rede neural profunda, foram realizados 16 testes, apresentados no Quadro 18.

Quadro 18 - Experimentos de modelos neurais convolucionais

Experimento	Pixels	Campo receptivo	Núcleos	Função ativação	Alteração	Precisão %	Tempo treinamento
Modelo 1	640x490	3x3	32,64, 128,128	ReLU	---	---	---
Modelo 2	150x150	3x3	32, 64, 128	ReLU	Redimensionar	31,08	01h10m21s
Modelo 3	150x150	3x3	32, 64, 128	ReLU	Dropout	37,36	02h34m28s
Modelo 4	150x150	3x3	32, 64, 128	ReLU	Normalização	24,18	02h05m19s
Modelo 5	224x224	5x5	64, 128, 256	PReLU	Sem normalização	52,75	02h55m37s
Modelo 6.1	224x224	5x5	64, 128, 256	PReLU	Otimizador Sgd	27,89	02h04m50s
Modelo 6.2	224x224	5x5	64, 128, 256	PReLU	Otimizador Nadam	28,96	02h05m31s
Modelo 6.3	224x224	5x5	64, 128, 256	PReLU	Otimizador Adagrad	35,96	02h05m25s
Modelo 6.4	224x224	5x5	64, 128, 256	PReLU	Otimizador Adamax	64,85	4h12m21s
Modelo 6.5	224x224	5x5	64, 128, 256	PReLU	Otimizador Adadelta	65,87	06h10m45s
Modelo 6.6	224x224	5x5	64, 128, 256	PReLU	Otimizador Rmsprop	81,65	08h09m04s
Modelo 6.7	224x224	5x5	64, 128, 256	PReLU	Otimizador Adam	85,69	15h01m50s
Modelo 7	98x98	5x5	64, 128, 256	PReLU	Segmentação	90,00	31h05m51s
Modelo 8	98x98	5x5	64, 128, 256	PReLU	Paralelismo	47,05	01h52m29s
Modelo 9	98x98	5x5	64, 128, 256	PReLU	Global MaxPooling	86,97	31h57m09s
Modelo 10	98x98	5x5	64, 128, 256	PReLU	Average Pooling	96,33	34h14m08s

Fonte: elaborado pelo autor.

O `Modelo 1` foi criado contendo quatro blocos de convolução e um bloco para a classificação. Na execução deste teste as imagens estavam na sua dimensão original, isto é, 640x490 pixels de largura e altura, sem a segmentação da face. Cada bloco convolucional possuía uma camada de convolução. O primeiro bloco convolucional possuía 32 núcleos, o segundo 64, o terceiro e quarto 128. A função utilizada para ativação dos neurônios foi ReLU.

Contudo, devido a quantidade de parâmetros a serem treinados, foi necessário abandonar o modelo proposto, pois o processamento seria inviável para o recurso tecnológico disponível.

A definição do `Modelo 2` pode ser vista no Quadro 19 e na terceira linha do Quadro 12. A primeira coluna, *Layer (type)*, apresenta a identificação dos blocos da rede neural, assim como o tipo de camada, a segunda coluna, *Output Shape*, apresenta o formato dos dados e a quantidade de núcleos extratores na saída da respectiva camada, e a terceira coluna, *Param #*, apresenta a quantidade de parâmetros computados pela camada. As entradas da rede foram alteradas para a dimensão 150x150 pixels de largura e altura, onde foi reduzido para três blocos convolucionais, a fim de reduzir o processamento. Cada bloco convolucional foi formado por uma camada de convolução, uma de ativação, e uma de subamostragem, e o bloco de classificação foi formado por duas camadas densas `Classi_Densa1`, `Classi_Densa2`. As três camadas convolucionais, `BL01_Conv`, `BL02_Conv` e `BL03_Conv`, possuem 32, 64, e 128 núcleos, respectivamente, com dimensão espacial de 3x3. A função de ativação utilizada nos blocos convolucionais foi a ReLU, identificado por `BL01_Relu`, `BL02_Relu` e `BL03_Relu`. O tipo de camada de subamostragem utilizado foi Max Pooling, identificados por `BL01_Polling`, `BL02_Polling` e `BL03_Polling`, com campos receptivos desta de dimensão 2x2 pixels de largura e altura. O início do bloco de classificação é identificado por `Ligacao`, que achata as matrizes dos dados da imagem, convertendo em um vetor de uma dimensão (1-D). A primeira camada densa, `Classi_Densa1`, possui 1024 neurônios, e a segunda camada densa, `Classi_Densa2`, tem 8 neurônios. Entre as camadas densas possui uma camada de ativação ReLU, identificada por `Classi_Relu`. O classificador linear utilizado na última camada densa foi Softmax. A função de perda para o aprendizado da rede foi definida como `categorical_crossentropy`. A rede foi treinada por 100 iterações e a precisão obtida por este modelo foi de 31,08%.

Quadro 19 – Testes – Primeira rede convulocional

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Bl01_Conv (Conv2D)           (None, 148, 148, 32)      320
_________________________________________________________________
BL01_Relu  (Activation)      (None, 148, 148, 32)      0
_________________________________________________________________
Bl01_Pooling (MaxPooling2D)  (None, 74, 74, 32)        0
_________________________________________________________________
BL02_Conv (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
BL02_Relu (Activation)       (None, 72, 72, 64)        0
_________________________________________________________________
BL02_Pooling (MaxPooling2D)  (None, 36, 36, 64)        0
_________________________________________________________________
Bl03_Conv (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
Bl03_Relu (Activation)       (None, 34, 34, 128)       0
_________________________________________________________________
Bl03_Pooling (MaxPooling2D)  (None, 17, 17, 128)       0
_________________________________________________________________
Ligacao (Flatten)            (None, 36992)             0
_________________________________________________________________
Classi_Densa1 (Dense)        (None, 512)               18940416
_________________________________________________________________
Classi_Relu (Activation)     (None, 512)               0
_________________________________________________________________
Classi_Densa2 (Dense)        (None, 8)                 4104
=================================================================
Total params: 19,037,192
Trainable params: 19,037,192
Non-trainable params: 0
_________________________________________________________________
```

Fonte: elaborado pelo autor.

O `Modelo 3` diferencia-se do modelo anterior apenas pela inclusão de duas camadas de Dropout. Conforme pode ser visto no Quadro 20, a primeira camada de Dropout, `Dropout1`, posicionada após a última camada de subamostragem, com taxa de queda de 25% e a segunda camada de Dropout, identificada por `Dropout2`, intercalando as duas camadas densa, com a taxa de queda em 50%. Esta alteração aumentou a precisão da rede para 37,36%.

Quadro 20 - Testes – Inclusão das camadas de Dropout

Bl03_Pooling (MaxPooling2D)	(None, 17, 17, 128)	0
Dropout1 (Dropout)	(None, 17, 17, 128)	0
Ligacao (Flatten)	(None, 36992)	0
Classi_Densa1 (Dense)	(None, 512)	18940416
Classi_Relu (Activation)	(None, 512)	0
Dropout2 (Dropout)	(None, 512)	0
Classi_Densa2 (Dense)	(None, 8)	4104

Fonte: elaborado pelo autor.

O `Modelo 4` é criado incluindo uma camada de normalização em cada bloco convolucional. Esta camada de normalização está posicionada entre a camada de ativação e subamostragem. Esta normalização altera os valores que serão considerados posteriormente na camada de subamostragem, pois não permite que os reajustes dos pesos sofram uma alteração brusca, normalizado assim as saídas. Contudo, a aplicação desta técnica reduziu a taxa de precisão do protótipo para 24,18%.

O `Modelo 5` sofreu grandes modificações nos hiperparâmetros, comparado com os anteriores. Foram removidas as camadas de normalização, pois foi observado através dos testes anteriores que não aprimoraram a precisão. As dimensões das imagens na entrada da rede foram alteradas para 224x224 pixels de largura e altura, obtendo assim mais paramentos para o aprendizado. A quantidade de núcleos para cada camada convolucional foi alterada para 64, 128 e 256, respectivamente, e o tamanho dos campos receptivos de cada núcleo mudou para dimensão 5x5 pixels de largura e altura. A camada de ativação foi alterada para o tipo PReLU. Os demais parâmetros, e camadas, permaneceram iguais ao `Modelo 4`. Esta alteração aumentou a taxa de precisão para 52,75%.

O `Modelo 6.1`, consistiu em realizar pequenos ajustes nos hiperparâmetros do treinamento para otimizar a função de perda. Existem vários otimizadores para esta técnica, e são apresentados na Tabela 2. Para que fosse possível realizar os testes, foi criado um modelo para cada otimizador. Baseado nos testes, o otimizador com melhor desempenho foi o `Modelo 6.7` com o otimizador `Adam`. Este otimizador foi desenvolvido por Kingma e Ba (2014). Como parâmetro do otimizador foi alterado o *learning rate* para $1e-3$ e a descida do gradiente para $1e-5$. Esta alteração trouxe um aumento considerável na precisão do protótipo, alterando para 85,69%.

Tabela 2 – Testes - Otimizadores e precisão apurada

Modelo	Otimizador	Precisão %
Modelo 6.1	Sgd	27,89
Modelo 6.2	Nadam	28,96
Modelo 6.3	Adagrad	35,96
Modelo 6.4	Adamax	64,85
Modelo 6.5	Adadelta	65,87
Modelo 6.6	Rmsprop	81,65
Modelo 6.7	Adam	85,69

Fonte: elaborado pelo autor.

Após a constatação de que o `Modelo 6.7` de arquitetura apresentou resultados satisfatórios, iniciou então o ajuste nas imagens do banco de dados. Até o momento, para os modelos acima citados, não havia sido realizado a etapa de segmentação da face, e por este motivo aplicou-se a etapa de segmentação.

O `Modelo 7` possui entrada da rede neural com dimensões de 98x98 pixels de largura e altura, pois são extraídas características apenas da região da face segmentada. Com esta aprimoração no conjunto de dados, a precisão da rede aumentou para 90%. A Tabela 3 a presenta a precisão para cada emoção classificada pelo `Modelo 7`. O treinamento da rede neural deste modelo levou 1 dia, 3 horas, 51 minutos e 9 segundos. O próximo modelo testado, `Modelo 8`, consiste em alterar o tipo de processamento da rede neural profunda. Foi construída uma rede na qual a saída processada por uma camada, era utilizada para alimentar duas camadas em paralelo, e posteriormente reorganizadas para o modo sequencial. Contudo, este modelo de rede não apresentou taxa de precisão maior que o `Modelo 7`, alcançando apenas 47,05%. Acredita-se que, para esse tipo de rede, seja necessário mais tempo de pesquisa e experimentação.

Tabela 3 – Testes - Precisão da rede neural com as faces segmentadas

Emoção	Precisão %
Alegria	99
Desgosto	93
Desprezo	79
Medo	100
Neutra	81
Raiva	89
Surpresa	99
Tristeza	70

Fonte: elaborado pelo autor.

A próxima arquitetura criada, `Modelo 9`, sofreu alterações nas camadas de subamostragem. Foram realizados testes alterando a técnica de Max Pooling para Global Max Pooling. Isto é, ao invés da rede possuir uma camada de subamostragem em cada bloco convolucional, existe apenas uma camada de subamostragem, situada ao fim de todos os blocos de convolução. Esta técnica cria um mapa global com o maior valor dentre todas as

camadas convoluidas. Porém, ao aplicar esta técnica, observou-se que a mesma reduziu a taxa de precisão para 86,97%.

Por fim, foi criado o último modelo de rede neural profunda, `Modelo 10`, onde sua arquitetura pode ser vista no Quadro 21. Neste modelo foi alterado a técnica da camada de subamostragem para Average Pooling. Esta técnica consiste em criar a subamostragem a partir da média dos valores resultantes da convolução, ao contrário do `Modelo 8`, que obteve o maior valor dentre os resultados. Com o uso desta técnica, a taxa de precisão alcançou 96,33%, sendo considerado o modelo mais preciso para o escopo deste trabalho.

Quadro 21 - Rede neural DeepEmotive

Layer (type)	Output Shape	Param #
Bl01_Conv (Conv2D)	(None, 98, 98, 64)	1664
Bl01_PReLU (PReLU)	(None, 98, 98, 64)	614656
Bl01_Pooling (AveragePooling	(None, 49, 49, 64)	0
Bl02_Conv (Conv2D)	(None, 45, 45, 128)	204928
Bl02_PRelu (PReLU)	(None, 45, 45, 128)	259200
Bl02_Pooling (AveragePooling	(None, 22, 22, 128)	0
Bl03_Conv (Conv2D)	(None, 18, 18, 256)	819456
Bl03_PRelu (PReLU)	(None, 18, 18, 256)	82944
Bl03_Pooling (AveragePooling	(None, 9, 9, 256)	0
Dropout1 (Dropout)	(None, 9, 9, 256)	0
Ligacao (Flatten)	(None, 20736)	0
Classi_Densa1 (Dense)	(None, 512)	10617344
Classi_PRelu (PReLU)	(None, 512)	512
Dropout2 (Dropout)	(None, 512)	0
Classi_Densa2 (Dense)	(None, 8)	4104
Total params: 12,604,808		
Trainable params: 12,604,808		
Non-trainable params: 0		

Fonte: elaborado pelo autor.

A Figura 35 ilustra a predição das emoções através das expressões faciais realizada pela rede. As imagens foram selecionadas aleatoriamente e as emoções preditas pela rede

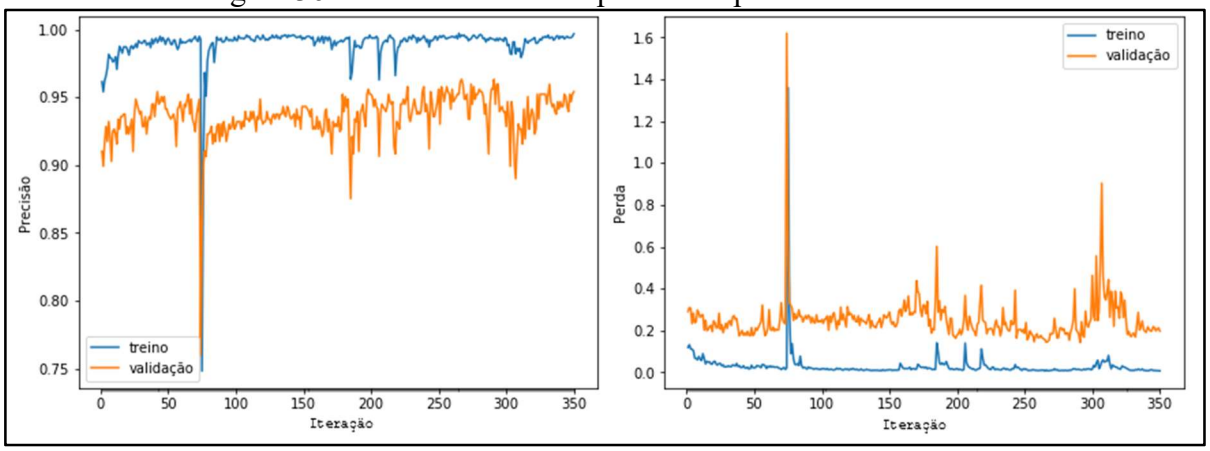
treinada baseada no `Modelo 10`. É possível observar que o algoritmo consegue predizer relativamente bem as emoções baseado nas expressões faciais.

Figura 35 - Emoções preditas pela rede DeepEmotive na base de dados Cohn-Kanade

Fonte: elaborado pelo autor.

A Figura 36 apresenta o gráfico do treinamento da rede, ilustrando a evolução da precisão e a taxa de perda. Observa-se no gráfico que o aprendizado passa por constantes variações de precisão, dado pela correção dos pesos e também pela técnica do Dropout, que em uma determinada iteração pode acabar eliminado as ligações da rede que cotiam informação correta. Existe também uma grande queda no aprendizado na iteração 75, aproximadamente, porém não foi possível identificar o motivo nos testes realizados.

Figura 36 - Gráfico da taxa de precisão e perda do treinamento

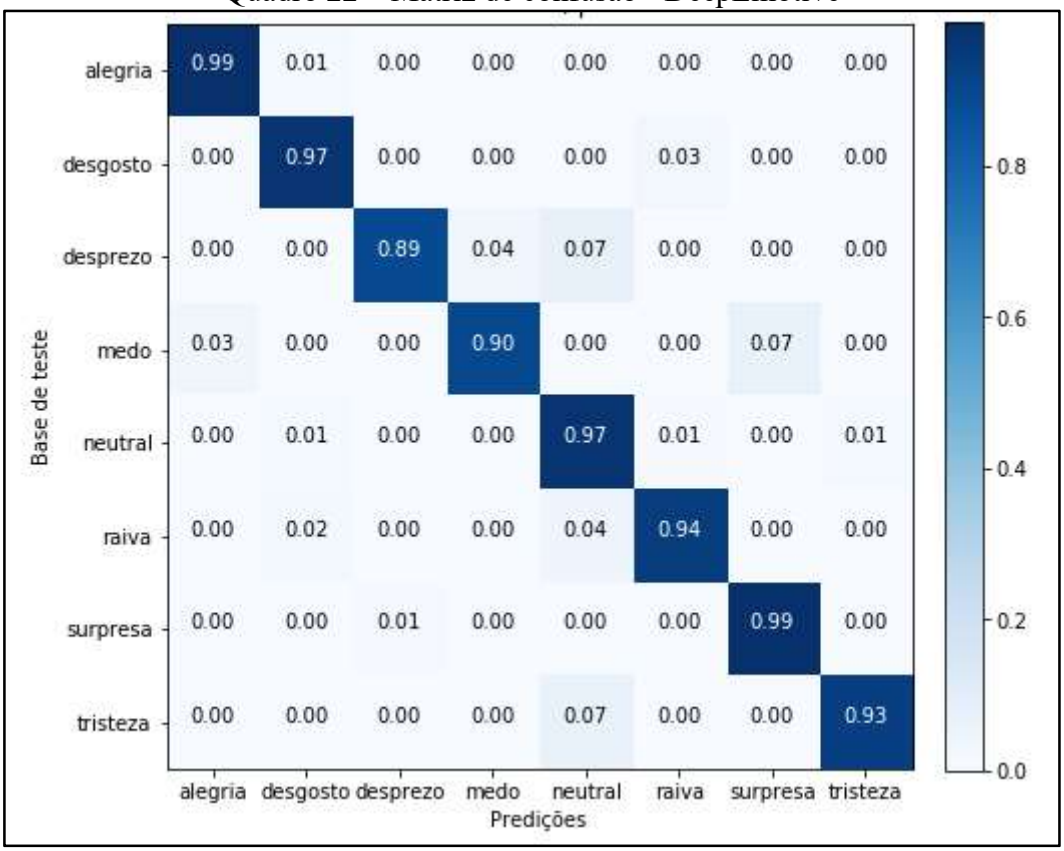


Fonte: elaborado pelo autor.

O Quadro 22, apresenta a matriz de confusão do protótipo do `Modelo 10`, gerado a partir da predição do conjunto de dados de teste. O eixo x da matriz representa as emoções preditas pelo protótipo e o eixo y representa as emoções reais. Com os resultados, foi possível observar que a emoção classificada com a menor taxa de precisão é a emoção de desprezo. Um dos motivos pode ser devido à baixa quantidade de amostra desta emoção, apenas 108, conforme já observado na Figura 32. Outro motivo, é que esta emoção também possui uma

taxa de erro considerável para emoções de medo e neutra, 0,4% e 0,7% respectivamente. O mesmo é visto na predição da emoção de medo, que possui a taxa de erro de 0,3% e 0,7% para alegria e surpresa, respectivamente. Isto significa que é necessário mais características para aumentar a precisão da classificação destas emoções, talvez aprofundando mais a rede ou obtendo mais amostras para extrair características que diferenciem melhor estas emoções. Outra análise importante é que as emoções de alegria e surpresa obtiveram a maior média de precisão, sendo 99% para ambas. Observa-se também que a emoção de tristeza possui uma taxa de erro a ser considerada, 0,7% para a emoção neutra. Isto significa que, para estas emoções a rede conseguiu identificar todas as características necessárias para a classificação. O motivo provável se deve a estas emoções possuírem expressões mais fortes que as diferenciam das demais.

Quadro 22 – Matriz de confusão - DeepEmotive



Fonte: elaborado pelo autor.

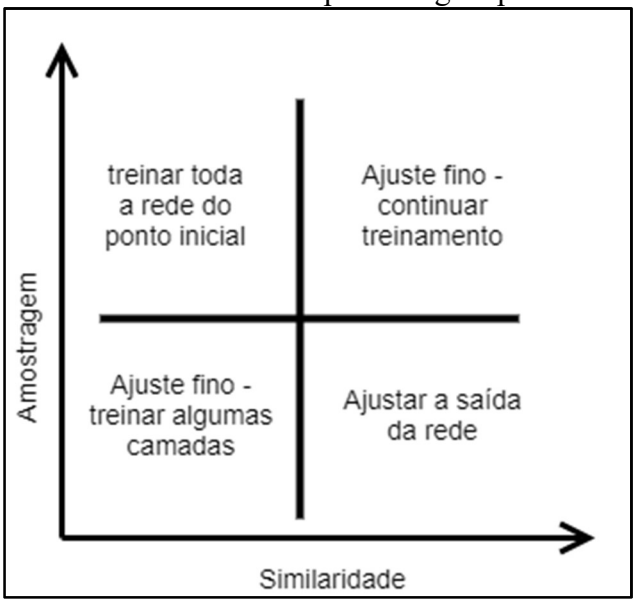
3.4.2 Transferência de aprendizado

Outra análise realizada para a validação deste protótipo foi a Transferência de Aprendizado ou Transfer Learning. A transferência de aprendizado consiste em utilizar um conhecimento adquirido na resolução de um problema e aplicá-lo a um novo problema relacionado a este. (PAN; YANG, 2010). Comumente são utilizadas quatro categorias de

transferência de aprendizado com redes neurais profundas: `ajustar saída da rede`, `ajuste fino – continuar treinamento`, `ajuste fino – treinar algumas camadas`, e, `treinar toda rede do ponto inicial`. A segunda e terceira categorias são conhecidas como ajuste fino (*fine tuning*). Existem várias redes neurais profundas disponíveis para uso desta técnica, como por exemplo a AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), VGG16 ou VGG19 (SIMONYAN; ZISSERMAN, 2014),

A definição de qual a melhor categoria de transferência de aprendizado a ser utilizada é baseada em dois aspectos, na quantidade de amostra e na similaridade, que podem ser observados na Figura 37. A primeira categoria, `ajustar saída da rede`, consiste em utilizar uma rede neural, treinada anteriormente, e ajustar apenas a última camada densa, responsável pela classificação. Para tanto, é necessário que a nova amostra de dados seja pequena e com alta similaridade de características do problema a ser resolvido. A segunda categoria, `ajuste fino – continuar treinamento`, executa a continuação do aprendizado da rede, isto é, inicia a rede com os pesos já treinados e continua a retropropagação a partir da nova amostra de dados, porém é necessária uma quantidade grande de amostra e alta similaridade. A terceira categoria, `ajuste fino – treinar algumas camadas`, realiza pequenas alterações nas camadas da rede neural já treinada, bloqueando algumas camadas e liberando outras para serem treinadas novamente. Esta categoria é aplicada quando se tem poucos dados e baixa similaridade. A quarta categoria, `treinar toda rede do ponto inicial`, é a transferência de arquitetura como um todo, aproveitando a estrutura da rede e treinando-a do ponto inicial a partir do novo conjunto de dados. Esta categoria é utilizada quando existe uma grande quantidade de amostras, porém pouco similar.

Figura 37 – Diagrama de transferência de aprendizagem para redes neurais profundas



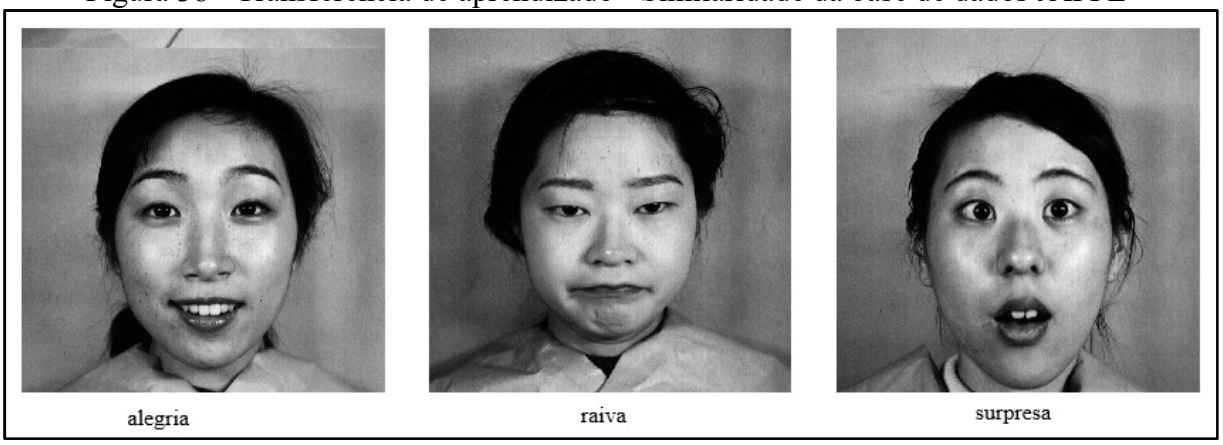
Fonte: elaborado pelo autor.

Neste trabalho, buscando validar a estrutura e o treinamento da rede neural DeepEmotive, foi realizado testes abordando duas categorias de transferência de aprendizado, primeira categoria e quarta categoria. As motivações pela escolha destas duas categorias são detalhadas nas subseções a seguir. Para executar os testes, foram selecionadas duas novas bases de dados, Japanese Female Facial Expression (JAFFE) e Facial Expression Recognition 2013 (FER-2013).

3.4.2.1 Resultados obtidos com a base de dados JAFFE

A base de dados JAFFE, criada por LYONS et al., 1998, possui um total de 213 amostras de sessenta mulheres japonesas. As amostras possuem as expressões faciais das seis emoções de Ekman et al. (1987), e também expressões da emoção neutra. As amostras desta base de dados estão distribuídas em: 31 imagens com expressões de alegria; 29 de desgosto; 32 de medo; 30 neutras; 30 de raiva; 30 de surpresa; e 31 de tristeza. Esta base de dados se enquadra na primeira categoria de transferência de aprendizado, pois ela é pequena e bem similar à base de dados utilizada para treinamento da rede DeepEmotive. A Figura 38 ilustra algumas imagens da base de dados JAFFE, a fim de comprovar a similaridade com a base de dados treinada pela rede DeepEmotive.

Figura 38 - Transferência de aprendizado - Similaridade da base de dados JAFFE



Fonte: elaborado pelo autor.

Para aplicar a transferência de aprendizado nesta base de dados foi necessário alterar a última camada da rede DeepEmotive, camada de classificação, para que contivesse apenas 7 neurônios de saída, representando as emoções disponíveis, pois a base JAFFE não possui as expressões de desprezo. O Quadro 23 apresenta a comparação dos resultados obtidos com o trabalho que originou a base de dados JAFFE, o trabalho de Ducan, Shine e English (2016), que utiliza a mesma base de dados, e o protótipo DeepEmotive.

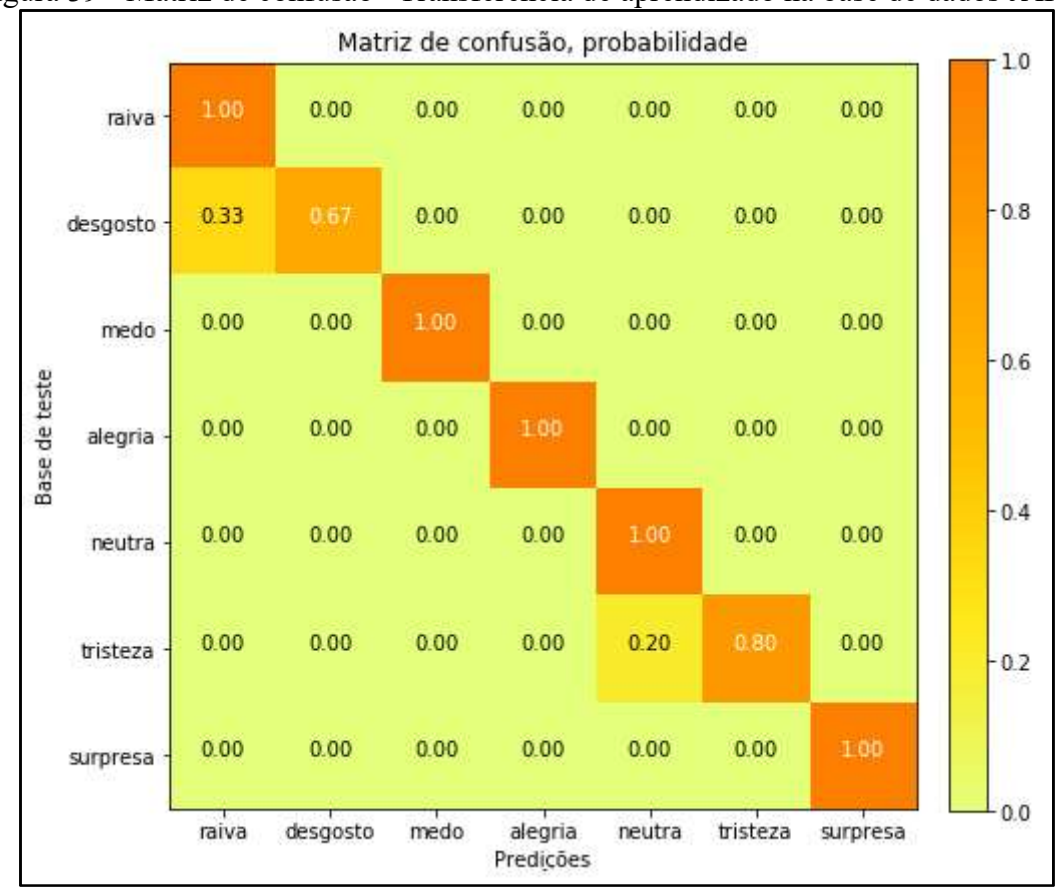
Quadro 23 – Transferência de aprendizado - Comparação da base de dados JAFFE

Trabalho	Lyons et al. (1998)	Ducan; Shine; English (2016)	DeepEmotive (2017)
Técnica	Filtros Gabor	Aprendizado Profundo	Aprendizado Profundo
Precisão	67%	57,1%	93,02%

Fonte: elaborado pelo autor.

Observa-se que, o resultado desempenhado pela rede DeepEmotive foi satisfatório, alcançando a precisão relativamente maior do que os trabalhos elencados, de 93,02%. Conforme pode ser visto na matriz de confusão ilustrada pela Figura 39, duas emoções tem uma alta taxa de perda: a emoção de desgosto, com 0,33% categorizada como raiva; e a emoção tristeza, com taxa de erro de 0,20% categorizada como neutra. Isto significa que a rede DeepEmotive precisa aprimorar o treinamento para estas emoções, extraindo características de novas amostras para ser possível diferenciar as emoções.

Figura 39 - Matriz de confusão –Transferência de aprendizado na base de dados JAFFE



Fonte: elaborado pelo autor.

3.4.2.2 Resultados obtidos com a base de dados FER-2013

A segunda base de dados utilizada para transferência de aprendizado foi a Facial Expression Recognition 2013 (FER-2013). Esta base de dados se enquadra na quarta categoria de transferência de aprendizado, pois ela possui 28709 imagens, porém é dissimilar para com a base de dados utilizada para o treinamento da rede DeepEmotive, pois não possui imagens validadas pelas AU's e a face nas imagens possuem ângulos diferentes. As amostras desta base de dados estão distribuídas em: 7215 imagens de alegria; 436 de desgosto; 4097 de medo; 4965 de neutra; 3995 de raiva; 3171 de surpresa; e 4830 de tristeza. Para demonstrar a dissimilaridade desta base de dados, são apresentadas algumas imagens na Figura 40. As imagens desta base de dados apresentam problemas já consolidados na área de visão computacional, como oclusão, perspectiva, ruídos, bem como imagens que não fazem parte do contexto do tema abordado no trabalho. Por este motivo, estas imagens interferem no resultado final do aprendizado. Segundo Sarkar (1991), a aparência de um objeto pode ser radicalmente alterada devido às características relevantes estarem esclarecidas ou desfocadas. O oposto, uma imagem com boa iluminação, pode reduzir algum ruído e diminuir o tempo de processamento.

Figura 40 – Transferência de aprendizado – Dissimilaridades encontradas na base de dados FER-2013

Fonte: elaborado pelo autor.

A base de dados FER-2013, foi utilizada pela competição The ICML 2013 Workshop on Challenges in Representation: A report on three machine-learning contests, promovida pela plataforma Kaggle[11] para o reconhecimento de emoções através de expressões faciais em imagens. O vencedor da competição, Tang (2013), utilizou a técnica de aprendizado profundo com redes convolucionais. Para realizar os testes foi necessário realizar alterações na rede DeepEmotive, configurando a saída da rede para classificar apenas sete emoções. Houve a necessidade também de alterar o tamanho da entrada padrão da rede DeepEmotive, visto que, as imagens da base de dados FER-2013, possui dimensões de 98x98 pixels de largura e altura. Por fim, foi refeito o treinamento da rede, sem a utilização dos pesos anteriores. O Quadro 24 apresenta a comparação dos resultados obtidos pelo vencedor da competição ICML 2013 e o protótipo DeepEmotive criado neste trabalho.

Quadro 24 – Transferência de aprendizado - Comparação da base de dados FER-2013

Trabalho	**Tang (2013)**	**DeepEmotive (2017)**
Técnica	Aprendizado profundo	Aprendizado profundo
Classificador	L2-SVM	Softmax
Precisão	71,16%	60,62%

Fonte: elaborado pelo autor.

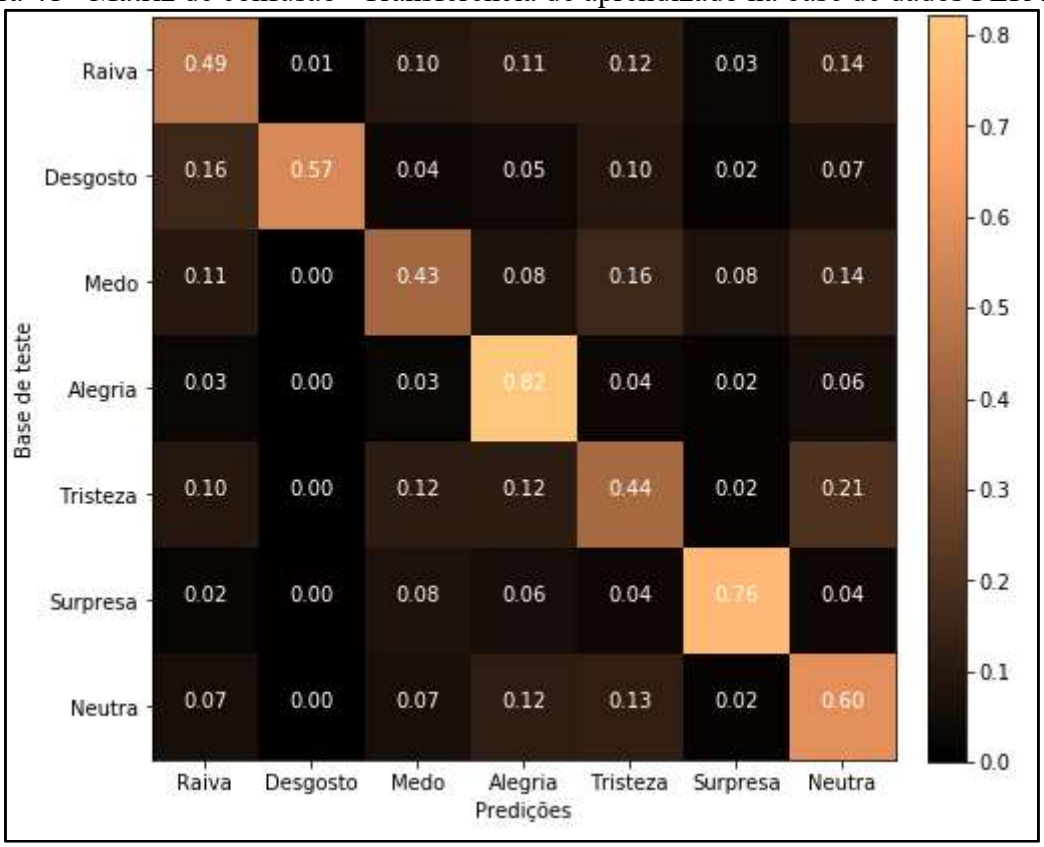
O resultado desempenhado pela rede DeepEmotive foi menor que vencedor da competição, alcançando a precisão de 60,62%. Porém, a rede DeepEmotive provou possuir um algoritmo robusto, tendo em vista que a precisão de classificação é superior à taxa de escolha randômica de 50% (KEARNS; VALIANT, 1994). A diferença pode ser atribuída as técnicas utilizadas, no processamento dos dados e no classificador. O trabalho de Tang (2013) utiliza um classificador L2-SVM12, e o protótipo DeepEmotive utiliza o classificador Softmax. O trabalho de Tang (2013) apresenta comparativos abordando os dois classificadores comparados, L2-SVM e Softmax.

A Figura 41 ilustra a matriz de confusão criada pela rede DeepEmotive. Observa-se que a classificação de duas emoções tem uma alta taxa de perda para várias emoções, sendo

[11] Kaggle é uma plataforma web de competições de Data Science. Atualmente é propriedade da empresa Google.
[12] L2-SVM é uma generalização do classificador SVM, que busca encontrar a margem máxima entre os pontos de classes diferentes.

que três emoções apresentam taxa abaixo de 50%. A taxa de erro é distribuída entre várias emoções, provando que a rede DeepEmotive não possui características suficientes para predizer com exatidão as imagens desta base de dados. É necessário reestruturar a rede ou alterar os hiperparâmentros em novos testes.

Figura 41 - Matriz de confusão –Transferência de aprendizado na base de dados FER-2013



Fonte: elaborado pelo autor.

3.4.3 Comparação dos resultados obtidos pelo protótipo DeepEmotive com os correlatos

Nesta seção é apresentada a comparação do trabalho proposto com os trabalhos correlatos descritos no capítulo anterior, bem como suas principais características. Conforme pode ser observado no Quadro 25, todos os trabalhos realizam o reconhecimento das seis emoções consideradas básicas e universais. O trabalho de Bartlett et al. (2003) acrescenta a expressão neutra, os trabalhos de Candra et al. (2016) e Amin, Chase e Sinha (2017) acrescenta a expressão de desprezo, o trabalho proposto (Deepemotive) acrescenta as expressões neutra e de desprezo.

Quadro 25 - Comparativo dos trabalhos correlatos

Características / Trabalho	Bartlett et al. (2003)	Tang; Huang, (2008)	Candra et al. (2016)	Amin; Chase; Sinha (2017)	DeepEmotive (2017)
Dimensão das imagens	2D	3D	2D	2D	2D
Extrator de características	GBoost, Viola-Jones	Distância entre Pontos	Viola-Jones, E-HOG	Aprendizado profundo	Viola-Jones, Aprendizado profundo
Classificador	SVM	SVM	SVM	Softmax	Softmax
Emoções reconhecidas	7 (Ekman et al. 1987) + Neutra	6 (Ekman et al. 1987)	7 (Ekman et al. 1987) + Desprezo	7 (Ekman et al. 1987) + Neutra	8 (Ekman et al. 1987) + Desprezo e Neutra
Tempo real	Sim	Não	Não	Não	Não
Precisão	93%	87,1%	96,4%	61,05%	96,33%

Fonte: elaborado pelo autor.

Observa-se também que, o trabalho de Tang e Huang (2008) não utiliza a técnica de Viola-Jones na fase de extração de características, os autores utilizam o cálculo da distância euclidiana dos pontos faciais. O trabalho de Amin; Chase; Sinha (2017) também não utiliza a técnica Viola-Jones, é aplicado o aprendizado profundo em uma base de dados onde a face já está segmentada. O trabalho de Tang e Huang (2008) é o único, entre os correlatos, que aborda o reconhecimento de emoções através do espaço 3D, porém apresenta a menor taxa precisão.

Dentre os correlatos, apenas o trabalho de Amin, Chase, Sinha (2017), utiliza a mesma técnica do protótipo proposto, realizando a extração de características através de redes neurais convolucionais, utilizado o aprendizado profundo. Similar ao trabalho proposto, Amin, Chase, Sinha (2017) também utilizam o classificador Softmax, enquanto os demais trabalhos utilizam o classificador SVM. A abordagem de Bartlett et al. (2003) diferencia-se por realizar o reconhecimento das emoções em tempo real. O projeto de Candra et al. (2016) apresentou a maior taxa média de precisão, porém com uma pequena diferença de 0,07% da rede proposta, o DeepEmotive.

4 CONCLUSÕES

A proposta inicial deste trabalho foi desenvolver um protótipo capaz de reconhecer automaticamente as seis emoções básicas e universais: raiva; desgosto; medo; alegria; tristeza; surpresa, através da análise de expressões faciais. Para tanto, foram utilizadas imagens oriundas da base de dados Cohn-Kanade AU-Coded Expression Database Version 2 (LUCEY et. al. 2010). Foi utilizado a linguagem de programação Python, a ferramenta Jupyter Notebook, assim com os *frameworks* Keras, OpenCV, Scikit Learn, Numpy, Matplotlib.

O trabalho realizado alcançou os objetivos pré-estabelecidos, bem como, apresenta resultados positivos e limitações. Este propôs reconhecer a face em imagens utilizando técnicas de visão computacional e processamento digital de imagem e realizou este objetivo através da utilização do algoritmo de Viola-Jones implementado pela biblioteca OpenCV. As faces identificadas foram segmentadas e deram origem a uma nova base de dados. O trabalho realizado visou extrair as características faciais relevantes para identificar as expressões faciais utilizando técnicas de visão computacional e processamento digital de imagem e realizou este objetivo através da aplicação da convolução em imagens e utilização das bibliotecas Scikit Learn e Numpy para computação científica e manipulação dos dados. Este trabalho objetivou implementar um classificador de emoções utilizando redes neurais artificiais com a tipologia de Aprendizado Profundo e realizou este objetivo através da criação de redes neurais convolucionais profundas, implementadas no framework Keras. O trabalho realizado é capaz de reconhecer oito emoções através da identificação das expressões faciais com precisão de 96,33% atingindo o objetivo geral proposto inicialmente. As duas emoções reconhecidas e que não estavam na proposta inicial são desprezo e neutra.

Adicionalmente, foi implementada a técnica de Transferência de Aprendizado, com o objetivo de validar o aprendizado obtido pelo classificador de emoções criado, bem como para investigar suas limitações. Os resultados obtidos foram satisfatórios quando aplicado a transferência de aprendizado na base de dados Japanese Female Facial Expression (JAFFE), atingindo uma precisão de 93,02%. O JAFFE possui imagens similares a base de dados utilizadas para o treinamento da rede DeepEmotive. Com este resultado é possível concluir que o DeepEmotive é eficaz no reconhecimento automático de expressões faciais de outras bases de dados que sejam relativamente similares a base de dados utilizada para o mesmo, porém não se limitando somente a esta. Este pode servir como referência para a comunidade tecnológica na geração de novos algoritmos computacionais, que, baseado nesta arquitetura reconhece automaticamente as emoções através de expressões faciais.

Uma limitação apresentada pelo protótipo pode ser observada quando a técnica de transferência de aprendizado é aplicada na base de dados Facial Expression Recognition 2013 (FER-2013). O protótipo apresentou baixa assertividade, alcançado 60,62% de precisão. Uma das razões da baixa assertividade é que a base de dados FER-2013 possui imagens dissimilares às bases de dados citadas anteriormente, contendo imagens com problemas que ainda são desafiadores para a área de visão computacional, como oclusão e ruído. No entanto, o resultado foi superior à taxa de escolha randômica de 50%. Outra limitação apresentada no desenvolvimento do protótipo é o custo computacional requerido para aplicação da técnica de aprendizado profundo, necessitando de um bom recurso tecnológico para o processamento. Os testes desempenhados neste trabalho possuíam tempo duração, na fase de treinamento, variando entre 1 hora, para a pior taxa de precisão e 1 dia, 10 horas, 14 minutos, e 8 segundos para a melhor taxa. Isto limitou a quantidade de características que a rede é capaz de aprender, visto que, para que fosse possível a execução do algoritmo, as imagens precisam estar em baixa escala e também utilizando apenas um canal de cor.

Conclui-se que a técnica de aprendizado profundo de máquina pode ser utilizada para a identificação de emoções através de expressões faciais em imagens 2D, com um percentual fidedigno de acurácia. Tornando assim desnecessária a marcação explicita dos pontos relevantes a serem extraídos para obter o conhecimento, fase que é realizada por um especialista no domínio do problema. O trabalho proposto também disponibiliza um modelo de predição treinado, o qual através do conhecimento adquirido, pode predizer emoções em imagens de face humana. A criação do modelo preditivo agrega também relevância social ao protótipo, podendo ser utilizado em diversas áreas da sociedade, como por exemplo nas áreas de entretenimento, auxiliando na interação homem máquina, no marketing, através da informação do estado emocional dos clientes para auxiliar na venda, e no monitoramento de pacientes, predizendo sintomas que possuam correlação com o estado emocional, como a depressão.

4.1 EXTENSÕES

O projeto apresentado atingiu os objetivos propostos, porém, existem aspectos que podem ser melhorados para o aprimoramento do protótipo DeepEmotive:

a) aprimorar o conhecimento da rede neural, através da transferência de aprendizado, aumentando a generalização do conhecimento da rede;
b) implementar a estrutura proposta por este trabalho, em um ambiente tecnológico com maiores recursos, a fim de fornecer mais características a serem aprendidas

pela rede;

c) treinar a rede neural em uma nova base de dados, que possua uma distribuição de amostras mais homogênea, para que seja possível aprimorar a precisão do reconhecimento das emoções de desprezo e medo;
d) desenvolver uma API para distribuição do conhecimento adquirido pela rede, tornando possível sua utilização por outras áreas de estudo;
e) fazer experimentos utilizando outro tipo de classificador e verificar a precisão em uma base de dados dissimilar como a FER-2013;
f) complementar o protótipo com a funcionalidade de reconhecimento em tempo real;
g) utilizar o modelo preditivo, criado por este trabalho, em conjunto com outras abordagens para análise de sentimento, como análise de emoção em frames de vídeos.

REFERÊNCIAS

AMIN, Dhruv; CHASE, Patrick; SINHA, Kirin. **Touchy Feely**: An Emotion Recognition Challenge. Palo alto: Stanford. 2017. Disponível em: < http://cs231n.stanford.edu/reports.html>. Acesso em: 10 out. 2017.

BARLOW, John S. Electroencephalography: Basic Principles, Clinical Applications and Related Fields. **Jama: The Journal of the American Medical Association,** [S.l.], v. 250, n. 22, p.3108-3108, 9 dez. 1983.

BARTLETT, Marian Stewart et al. Real Time Face Detection and Facial Expression Recognition: Development and Applications to Human Computer Interaction. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION WORKSHOP, 1., 2003, Madinson. **Anais...** [S.l.]: IEEE, [2003]. v. 5, p. 53 - 53.

BASMAJIAN, John V. **Muscles Alive:** Their Functions Revealed by Electromyography. 2. ed. [S.l.]: Williams & Wilkins, 1985.

BENGIO, Yoshua; COURVILLE, Aaron; VINCENT, Pascal. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v. 35, n. 8, p. 1798-1828, 2013.

BOOTH, Paul A. **An Introduction to Human-computer Interaction**. [S.l.] Psychology Press, 1989.

BOROD, Joan C. (Ed.). **The neuropsychology of emotion**. New York: Oxford University Press, 2000.

CANNY, John. A computational approach to edge detection. **IEEE Transactions on pattern analysis and machine intelligence** [S.l.], v. 8, n. 6, p. 679-698, 1986.

CANDRA, Henry et al. Classification of facial-emotion expression in the application of psychotherapy using Viola-Jones and Edge-Histogram of Oriented Gradient. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY (EMBC), 38., 2016, Orlando. **Anais... [**S.l.]: IEEE, 2016. p. 423 - 426.

CHANG, Chih-chung; LIN, Chih-jen. LIBSVM: A library for support vector machines. **ACM Transactions On Intelligent Systems And Technology, [**S.l.], v. 2, n. 3, p.1-27, 2011.

CISCO SYSTEMS. **VNI Complete Forecast Highlights Tool:** 2015 Year in Review. 2015. Disponível em: <http://www.cisco.com/c/m/en_us/solutions/service-provider/vni-forecast-highlights.html>. Acesso em: 19 mar. 2016.

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

DARWIN, Charles; LORENZ, Konrad. **A expressão das emoções no homem e nos animais.** Sao Paulo: Companhia das Letras, 2000. Tradução de: The expression of the emotions in man and animals.

DICARLO, James j.; ZOCCOLAN, Davide; RUST, Nicole C. How Does the Brain Solve Visual Object Recognition? **Neuron,** [S.l.], v. 73, n. 3, p.415-434, fev. 2012.

DRAW.IO. **Diagrams for everyone, everywhere.** 2017. Disponível em: <https://about.draw.io/>. Acesso em: 08 out. 2017.

DUNCAN, Dan; SHINE, Gautam; ENGLISH, Chris. **Facial Emotion Recognition in Real Time**. Palo alto: Stanford. 2016. Disponível em: < http://cs231n.stanford.edu/reports.html>. Acesso em: 10 out. 2017.

EKMAN, Paul. **Emotions Revealed**: Recognizing Faces and Feelings to Improve Communication and Emotional Life. New York: Times Books, 2003.

_____. Facial expressions. In: DALGIEISH, Tim; POWER, Mick (Ed.). **Handbook of Cognition and Emotion.** Midsomer Norton: John Wiley & Sons, 1999. Cap. 16. p. 301-320.

EKMAN, Paul et al. Universals and cultural differences in the judgments of facial expressions of emotion. **Journal Of Personality And Social Psychology,** [S.l.], v. 53, n. 4, p.712-717, 1987.

EKMAN, Paul; FRIESEN, Wallace V. Measuring facial movement. **Environmental Psychology And Nonverbal Behavior,** [S.l.], v. 1, n. 1, p.56-75, 1976.

FASEL, Ian; FORTENBERRY, Bret; MOVELLAN, Javier. A generative framework for real time object detection and classification. **Computer Vision And Image Understanding,** [S.l.], v. 98, n. 1, p.182-210, abr. 2005.

FOLEY, Gretchen N.; GENTILE, Julie P. Nonverbal Communication in Psychotherapy. **Psychiatry (edgmont),** [S.l.], v. 7, n. 6, p.38-44, jun. 2010.

FREUND, Yoav; SCHAPIRE, Robert E. Experiments with a New Boosting Algorithm. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 12., 1996, Bari. **Proceedings...** Bari: [s.n], 1996. p. 148 - 156.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento digital de imagens.** 3. ed. São Paulo: Pearson, 2010.

GOODFELLOW, Ian J. et al. Challenges in Representation Learning: A Report on Three Machine Learning Contests. In: INTERNATIONAL CONFERENCE, 20., 2013, Daegu. **Proceedings...** Heidelberg: Springer, 2013. p. 117 - 124.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning.** Cambridge: MIT Press, 2016.

GLOROT, Xavier; BORDES, Antoine; BENGIO, Yoshua. Deep Sparse Rectifier Neural Networks. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND STATISTICS, 14., 2011, Fort Lauderdale. **Proceedings...** Fort Lauderdale: PMLR, 2011. v. 15, p. 315 - 323.

HORADAM, Kathy J. **Hadamard Matrices and Their Applications.** [S.l.]: Princeton University Press, 2012.

HE, Kaiming et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: INTERNATIONAL CONFERENCE ON COMPUTER VISION, 15., 2015, Santiago. **Proceedings...** Santiago: IEEE, 2015. p. 1026 - 1034.

LYONS, Michael et al. Coding Facial Expressions with Gabor Wavelets. In: INTERNATIONAL CONFERENCE ON AUTOMATIC FACE AND GESTURE RECOGNITION, 3., 1998, Nara. **Proceedings...** Nara: IEEE, 1998. p. 200 - 205.

JELINEK, Frederick. **Statistical Methods for Speech Recognition.** [S.l.]: MIT Press, 1997.

KANADE, Takeo; COHN, Jeffrey F.; TIAN, Yingli. Comprehensive database for facial expression analysis. In: AUTOMATIC FACE AND GESTURE RECOGNITION, 4., 2000, Crenoble. **Proceedings...** Crenoble: IEEE, 2000. p. 46 – 53.

KEARNS, Michael; VALIANT, Leslie. Cryptographic limitations on learning Boolean formulae and finite automata. **Journal Of The Acm,** [S.l.], v. 41, n. 1, p.67-95, jan. 1994.

KINGMA, Diederik P.; BA, Jimmy. Adam: A Method for Stochastic Optimization. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 3., 2015, San Diego. **Proceedings...** San Diego: ICLR, 2015. p. 1 - 15.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 25., 2012, Lake Tahoe. **Proceedings...** Lake Tahoe: Curran Associates Inc, 2012. p. 1097 - 1105.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature,** [S.l.], v. 521, n. 7553, p.436-444, 27 maio 2015.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings Of The IEEE,** [S.l.], v. 86, n. 11, p.2278-2324, 1998.

LINDEBERG, Tony et al. A computational theory of visual receptive fields. **Biological Cybernetics,** [S.l.], v. 107, n. 6, p.589-635, nov. 2013.

LUCEY, Patrick et al. The Extended Cohn-Kanade Dataset (CK+). In: COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION - WORKSHOPS, 3., 2010, San Francisco. **Proceedings...** San Francisco: IEEE, 2010. p. 94 - 101.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens.** Rio de Janeiro: Brasport, 1999.

MARQUES, Jorge. S. Reconhecimento de Padrões: métodos estatísticos e neuronais. 2ª. ed. Lisboa: IST Press, 2005.

MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The Bulletin Of Mathematical Biophysics,** [S.l.], v. 5, n. 4, p.115-133, dez. 1943.

MURRAY, Iain R.; ARNOTT, John L. Toward the simulation of emotion in synthetic speech: A review of the literature on human vocal emotion. **The Journal Of The Acoustical Society Of America,** [S.l.], v. 93, n. 2, p.1097-1108, fev. 1993.

NEVEROVA, Natalia et al. Learning Human Identity From Motion Patterns. **IEEE Access,** [S.l.], v. 4, p.1810-1820, abr. 2016.

OPENCV. Open Source Computer Vision Library. **OPENCV**. Disponível em: <http://opencv.org/>. Acesso em: 08 outubro 2017.

PAN, Sinno Jialin; YANG, Qiang. A Survey on Transfer Learning. **IEEE Transactions On Knowledge And Data Engineering,** [S.l.], v. 22, n. 10, p.1345-1359, out. 2010.

PICARD, Rosalind W. **Affective Computing.** Cambridge: MIT Press, 1997.

REEVE, Johnmarshall. **Motivacão e Emocão.** 4. ed. Rio de Janeiro: LTC, 2006.

ROSENBLATT, Frank. **Principles of Neurodynamics:** Perceptrons and the Theory of Brain Mechanisms. Washington: Spartan Books, 1962.

ROTENSTEIN, Lisa S. et al. Prevalence of Depression, Depressive Symptoms, and Suicidal Ideation Among Medical Students. **JAMA,** [S.l.], v. 316, n. 21, p.2214-2236, dez. 2016.

SANDBACH, Georgia et al. Static and dynamic 3D facial expression recognition: A comprehensive survey. **Image And Vision Computing,** [S.l.], v. 30, n. 10, p.683-697, out. 2012.

SARKAR, Nalini Ranjan. Machine vision for quality control in the food industry. In: FUNG, Daniel Y. C.; MATTHEWS, Richard F. (Ed.). **Instrumental Methods for Quality Assurance in Foods.** 44. ed. [s.l.]: Crc Press, 1991. Cap. 7. p. 167-189.

SCHERER, Dominik; MÜLLER, Andreas; BEHNKE, Sven. Evaluation of pooling operations in convolutional architectures for object recognition. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS, 20., 2010, Thessaloniki. **Proceedings...** Thessaloniki: Springer, 2010. p. 92 - 101.

SCHULTZ, David et al. Emotion knowledge in economically disadvantaged children: Self-regulatory antecedents and relations to social difficulties and withdrawal. **Development And Psychopathology,** [S.l.], v. 13, n. 1, p.53-67, mar. 2001.

SERRA, Jean Paul. **Image analysis and mathematical morphology.** Orlando: Academic Press, 1983.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 3., 2015, San Diego. **Proceedings...** San Diego: ICLR, 2015. p.15.

SOLEYMANI, Mohammad et al. Analysis of EEG Signals and Facial Expressions for Continuous Emotion Detection. **IEEE Transactions On Affective Computing,** [S.l.], v. 7, n. 1, p.17-28, 1 jan. 2016.

SRIVASTAVA, Nitish et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. **Journal Of Machine Learning Research,** Toronto, v. 15, n. 1, p.1929-1958, jun. 2014.

SZELISKI, Richard. **Computer Vision:** Algorithms and Applications. London: Springer, 2011.

TANG, Hao; HUANG, Thomas S. 3D facial expression recognition based on properties of line segments connecting facial feature points. In: INTERNATIONAL CONFERENCE ON AUTOMATIC FACE & GESTURE RECOGNITION, 8., 2008, Amsterdam. **Proceedings...** [S.l.]: IEEE, 2009. p. 1 - 6.

TANG, Yichuan. Deep Learning using Linear Support Vector Machines. In: CHALLENGES IN REPRESENTATION LEARNING, 20., 2013, Daegu. **Proceedings...** [S.l.]: ICML, 2013. p. 15 - 21.

TIAN, Ying-li; KANADE, Takeo; COHN, Jeffrey F. Recognizing action units for facial expression analysis. **IEEE Transactions On Pattern Analysis And Machine Intelligence,** [S.l.], v. 23, n. 2, p. 97-115, fev. 2001.

TURING, Alan. M. I. Computing machinery and intelligence. **Mind,** [S.l.], v. 59, n. 236, p.433-460, 1950.

VIOLA, Paul; JONES, Michael. Rapid object detection using a boosted cascade of simple features. In: COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 9., 2001, Kauai. **Proceedings...** [S.l.]: IEEE, 2003. v. 1, p. 511 - 518.

WANG, Jun et al. 3D Facial Expression Recognition Based on Primitive Surface Feature Distribution. In: COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 13., 2006, New York. **Proceedings...** [S. L.]: IEEE, 2006. p. 1399 - 1406.

WANG, Sherrie. **Facial affect detection using convolutional neural networks**. Palo alto: Stanford. 2016. Disponível em: < http://cs231n.stanford.edu/reports.html>. Acesso em: 10 out. 2017.

YIN, Lijun et al. A 3D facial expression database for facial behavior research. In: INTERNATIONAL CONFERENCE ON AUTOMATIC FACE AND GESTURE RECOGNITION, 7., 2006, Southampton. **Proceedings...** [S. L.]: IEEE, 2006. p. 211 - 216.

ZHOU, Yi-tong; CHELLAPPA, Rama. Computation of optical flow using a neural network. In: INTERNATIONAL CONFERENCE ON NEURAL NETWORKS, 1., 1988, San Diego. **Proceedings...** [S.l.]: IEEE, 1988. v. 2, p. 71 - 78.

APÊNDICE A – Códigos auxiliares da etapa de preparar dados

Nesta seção é apresentado os códigos auxiliares implementados para a fase de preparação de dados. No Quadro 26 pode ser visto o código da função `extrair_dados`. O Quadro 27 apresenta a função `empilhar_dados`. O Quadro 28 traz a função auxiliar `separar_treino_e_teste`. No Quadro 29 pode ser visto o código da função `formatar_shape`. Por fim, o Quadro 30 apresenta a função `criar_matriz-verdade`.

Quadro 26 - Função auxiliar para extrair dados da imagem

```
def extrair_dados(imagens):
    """ Esta função é responsável por extrair os pixels de
        todas as imagem da base de dados.
        Recebe como entrada uma lista de objeto de
        imagem python e retorna uma lista de
        numpy.ndarray contendo todos os pixels
        de todas as imagens.

        Args:
            imagens (list): lista objeto de imagem python.

        Returns:
            array (np.ndarray): retorna uma array numpay.
    """
    data = [] # para armazenar os dados
    array_emocoes = [] # para armazenar os descritores

    print("Extraindo os dados...")
    # para cada imagem
    for imagem in range(len(imagens)):
        # extrai o array de pixels da imagem
        data.append(_get_pixels_imagem(imagens[imagem][0]))
        # extrai a emocao que já estava definida na tupla
        array_emocoes.append(imagens[imagem][1])

    print("Quantidade de imagens extraidas:",len(data))
    print("Quantidade de emocoes extraidas:",len(array_emocoes))
    print("Extração de pixels completa!")
    return data, array_emocoes
```

Fonte: elaborado pelo autor.

Quadro 27 – Função auxiliar para empilhar os dados

```
def empilhar_dados(dados):
    """ Esta função é responsável por empilhar os dados
        para o formato que exigido para trabalho com o
        framework Keras. Recebe como entrada lista de
        dados e retorna uma tupla contendo a
        quantidade de dados, dimensão x, e dimensão y.

        Args:
            dados (list): lista de dados.

        Returns:
            tuple: retorna uma tupla com
            (quantidade, largura, altura).
    """
    print("Empilhando conjunto de dados:(quantidade, largura, altura)...")
    # faz uso da fucionalidade numpy de empilhamento
    dados_empilhados = np.stack(dados)
    print("novo formato dos dados:", dados_empilhados.shape)
    return dados_empilhados
```

Fonte: elaborado pelo autor.

Quadro 28 - Função auxiliar para separar dados de treino e teste

```
def separar_treino_e_teste(imagens, descritores):
    """ Esta função é responsável por separa os dados
        em conjunto de treino e teste. Recebe como
        entrada tupla contendo todos os dados da
        imagens e uma tupla contendo os descritores.
        Retorna uma tupla contendo imagens de treino,
        descritores de treino, imagens de teste e
        descritores de teste.

        Args:
            imagens (tuple): tupla de dados .

        Returns:
            tuple: tupla contendo dados de treino, dados de teste.
    """
    # test_size=0.2 = percentual de separação 20% para teste
    # random_state = definindo a semente para escolha aleatoria
    print('Separando os dados em grupo de treino e teste...')
    X_train, X_test, y_train, y_test = train_test_split(X, y,
                    test_size=0.2, random_state= int(qtd_imagens))
    print('Imagens separadas para treino: ',X_train.shape)
    print('Imagens separadas para teste: ',X_test.shape)
    print('Emoções separadas para treino: ',y_train.shape)
    print('Emoções separadas para teste: ',y_test.shape)
    return X_train, X_test, y_train, y_test
```

Fonte: elaborado pelo autor.

Quadro 29 - Função auxiliar apara formatar a estrutura dos dados

```
def formatar_shape(X_train, X_test, canais=1):
    """ Esta função é responsável por formatar os dados
        para trabalhar com o deep learning, de acordo com
        o framework que está em background (tensorflow(tf)
        ou Theano(th)).
        Recebe como parametro a quantidade de canais de cor
        que as imagens possuem.

        Args:
            X_train (tuple): imagens de treinamento
            X_train (tuple): imagens de teste
            canais (int): quantidade de canais de profundidade.
        Returns:
            tuple: tupla contendo dados de treino, dados de teste,
            e o shape dos dados, formatados para deep learning.
    """
    print("Adicionando canais de cores na estrutura de dados...")
    # O framework theano trabalha com a quantidade de canais
    # de cores a frente da dimesão da imagem
    if K.image_data_format() == 'channels_first': # = th
        shape = (canais, img_cols, img_rows)
        #shape = (1, img_cols, img_rows)
        print("Trabalhando com theano:", shape)
    # O framework tensorflow trabalha com a quantidade de canais
    # de cores apos a dimesão da imagem
    else:  # channel_last = tf
        shape = (img_cols, img_rows, canais)
        #shape = (img_cols, img_rows, 1)
        print("Trabalhando com tensorflow:",shape)
    print("Novo formato para estrutura de dados contendo o canal de cor: ",shape)
    print('Alterando o formato das imagens, colocando a estrutura com canal de cor...')
    X_train = X_train.reshape((X_train.shape[0],) + shape).astype('float32')
    X_test = X_test.reshape((X_test.shape[0],) + shape).astype('float32')
    print('Novo formato das imagens separadas para treino: ',X_train.shape)
    print('Novo formato das imagens separadas para teste: ',X_test.shape)
    # cada pixels é normalizado = pixel = pixel / 255
    print('Normalizando os dados...')
    X_train /= 255
    X_test /= 255
    return X_train, X_test, shape
```

Fonte: elaborado pelo autor.

Quadro 30 - Função auxiliar para criar a matriz verdade dos descritores

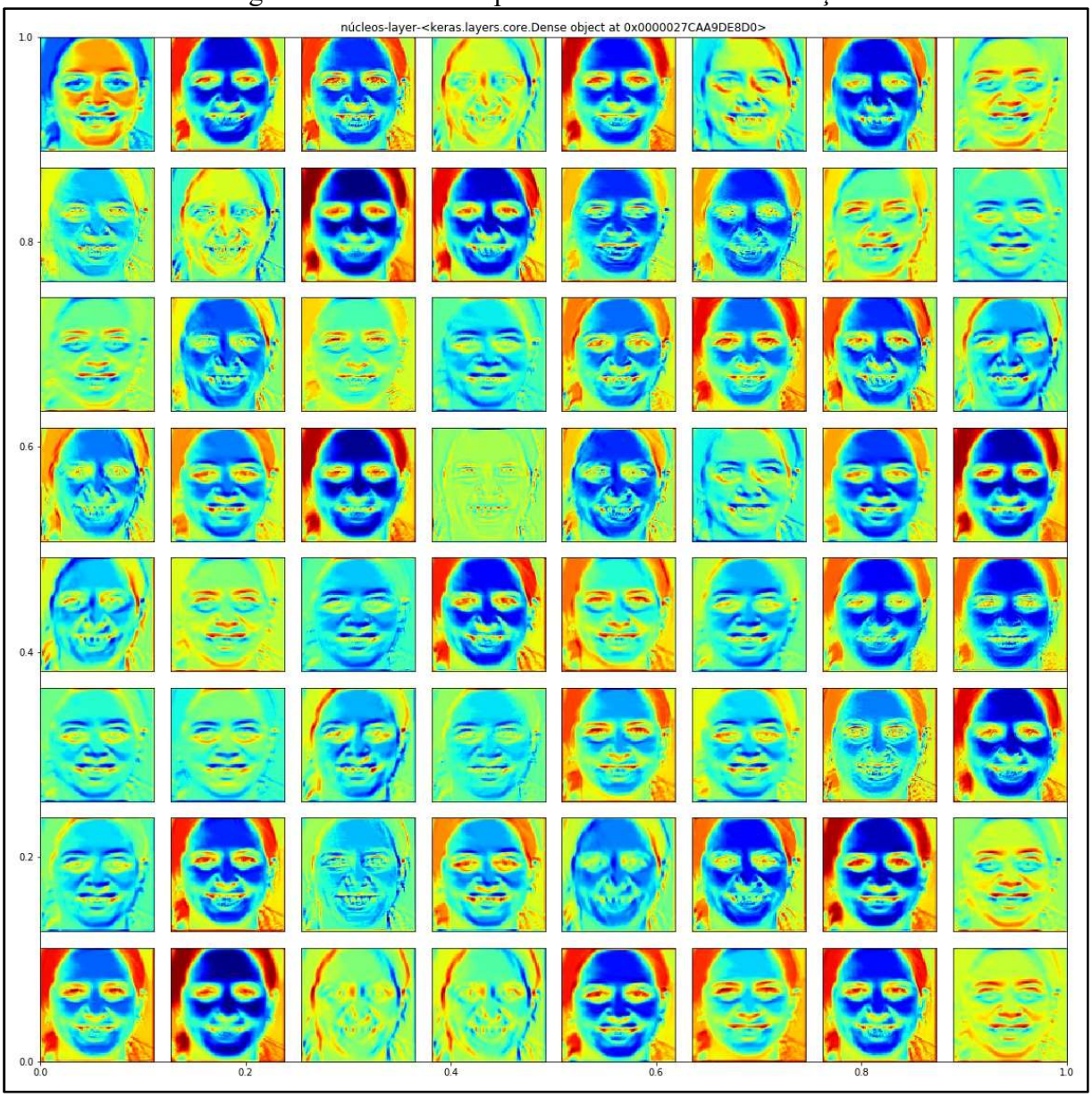
```
def criar_matriz_verdade(y_train, y_test):
    """ Esta função é responsável por criar uma matriz
        verdade para todos os descritores do conjunto de dados.
        Recebe como parametro os descritores de treino e teste.

        Args:
            Y_train (tuple): descritores de emoção de treinamento
            Y_test (tuple): descritores de emoção de teste

        Returns:
            tuple: tupla contendo as matrizes verdade dos
            descritores de treino, dados de teste,
    """
    print('Criando a matriz de classificação binária...')
    # Classificando os descritores de treino
    Y_train = np_utils.to_categorical(y_train)
    # Classificando os descritores de teste
    Y_test = np_utils.to_categorical(y_test)

    # verificando a quantidade de classes encontradas
    n_treino = Y_train.shape[1]
    print('Classes encontradas para imagens: ', n_treino)

    # verificando a quantidade de classes encontradas
    n__teste = Y_test.shape[1]
    print('Classes encontradas para emocoes: ', n__teste)
    return Y_train, Y_test
```

Fonte: elaborado pelo autor.

APÊNDICE B – Visualização das camadas intermediárias da rede neural profunda

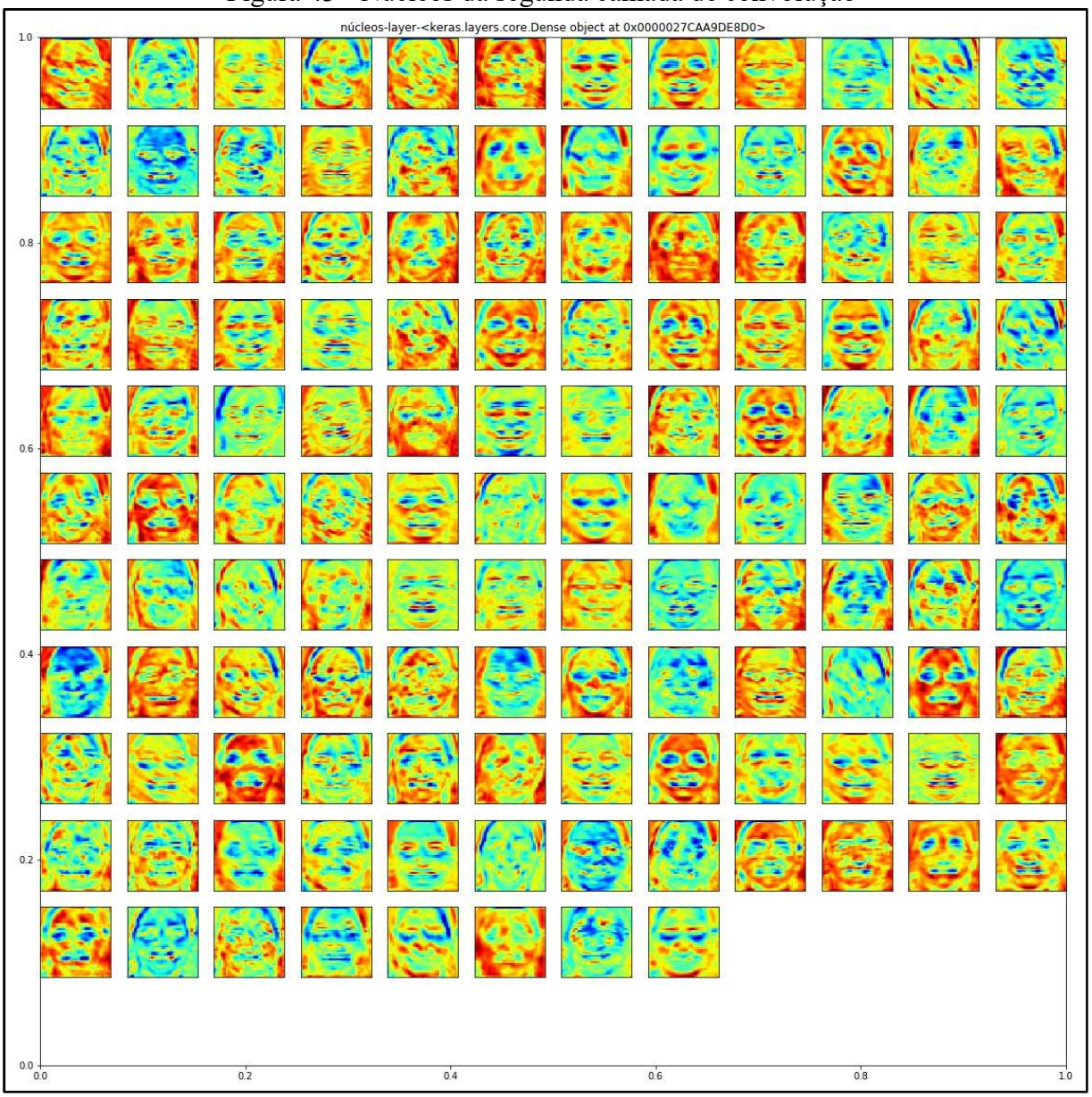
A seguir é apresentado a visualização dos mapas de características criados pela rede neural profunda DeepEmotive e obtidos através das camadas intermediárias. A Figura 42 ilustra os 64 núcleos da primeira camada de convolução, a Figura 43 ilustra os 128 núcleos da segunda camada, e a Figura 44 ilustra os 256 núcleos da última camada de convolução.

Figura 42 - Núcleos da primeira camada de convolução



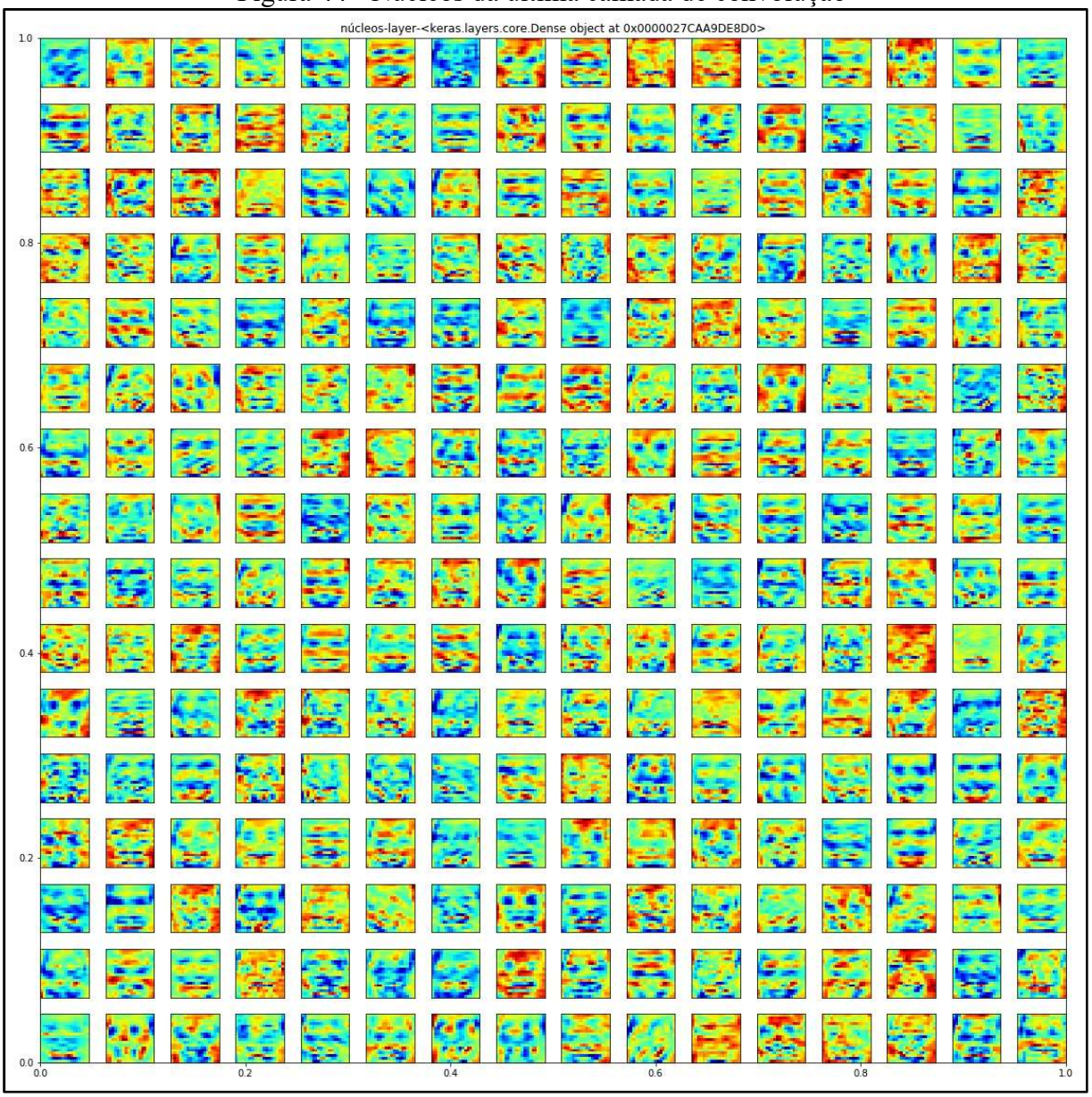
Fonte: elaborado pelo autor.

Figura 43 - Núcleos da segunda camada de convolução



Fonte: elaborado pelo autor.

Figura 44 - Núcleos da última camada de convolução



Fonte: elaborado pelo autor.