UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA E
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

ANDRÉ LUCAS ZANELLATO
CRISTIAN BORTOLINI FERREIRA

SISTEMA DE RECONHECIMENTO DE EXPRESSÕES FACIAIS

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2017

ANDRÉ LUCAS ZANELLATO
CRISTIAN BORTOLINI FERREIRA

SISTEMA DE RECONHECIMENTO DE EXPRESSÕES FACIAIS

Trabalho de Conclusão de Curso de Engenharia de Computação apresentado ao Departamento Acadêmico de Informática e Departamento Acadêmico de Eletrônica da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Engenheiro de Computação".

Orientador: Prof. Valfredo Pilla Júnior

Co-orientador: Prof. Dr. Gustavo Benvenutti Borba

CURITIBA

2017

André Lucas Zanellato
Cristian Bortolini Ferreira

SISTEMA DE RECONHECIMENTO DE EXPRESSÕES FACIAIS

Este trabalho de conclusão de curso foi julgado e aprovado como requisito parcial para obtenção do grau de Engenheiro de Computação pela Universidade Tecnológica Federal do Paraná.

Curitiba, 23 de Maio de 2017.

Prof. Valfredo Pilla Júnior
Prof. Orientador

Prof. Dr. Gustavo Benvenutti Borba
Prof. Co-orientador

Prof. Dr. Leyza Baldo Dorini

Prof. Dr. André Eugênio Lazzaretti

RESUMO

ZANELLATO, André Lucas; FERREIRA, Cristian Bortolini. SISTEMA DE RECONHECIMENTO DE EXPRESSÕES FACIAIS. 71 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática e Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

As expressões faciais fornecem muitas informações sobre as emoções e a interação social de indivíduos. Sistemas capazes de realizar a detecção automática das expressões faciais de pessoas podem ser usados para diversas aplicações, como: detecção de fatiga, identificação de pessoas suspeitas causadoras de riscos, interfaceamento com telefones celulares, ambientes de aprendizado, ambientes de entretenimento e atendimento ao cliente. Este projeto consiste em um sistema capaz de classificar a expressão facial presente em uma imagem frontal da face humana, resultando em uma das 7 expressões faciais: felicidade, tristeza, medo, raiva, surpresa, aversão e neutro. Para a realização dessa tarefa foi escolhido o treinamento de Aprendizado Profundo de duas camadas: a Rede Neural Convolucional, capaz de extrair características de imagens, e a Rede Interconectada para a classificação. Ambas redes foram implementadas com auxílio da biblioteca *Caffe* desenvolvida pela Universidade de Berkeley, disponível em *Python* e C++, e treinada com a base de dados *Cohn-Kanade* estendida. Para tal, foi necessário fazer um aumento artificial dos dados, já que uma Rede Neural Profunda precisa de uma quantidade significativa de dados. Além disso, antes do treinamento com as sete expressões faciais foi realizado um teste com somente três delas, para testar a arquitetura da Rede escolhida, a *AlexNet*, sofrendo algumas alterações para melhor se encaixar nas imagens disponíveis no banco escolhido. Após resultados que apoiaram a hipótese de bom resultado, realizamos o treinamento com as sete expressões. Os resultados obtidos em relação a classificação são avaliados de acordo com uma matriz de confusão, a qual permite analisar o comportamento dessas classificações para casos em que elas foram bem ou mal sucedidas. Por fim, foi implementado um sistema de reconhecimento em tempo real, utilizando o resultado do treinamento.

Palavras-chave: Rede, Neural, Convolucional, Reconhecimento, Expressão, Facial

ABSTRACT

ZANELLATO, André Lucas; FERREIRA, Cristian Bortolini. FACIAL EXPRESSION RECOGNITION SYSTEM. 71 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática e Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Facial Expressions give out a lot of information on emotions and social interactions of individuals. Systems capable of detecting facial expressions could be used for an wide range of applications, such as: fatigue detection, suspicious activity detection, cellphone integration, learning environments, entertainment and client intearction. This project consists of developmenting a system capable to recogninize and classify facial expression presented in an frontal image of an human face, resulting in one of the 7 facial expressions: happiness, sadness, fear, anger, surprise, disgust and neutral. In order to do this task, the method choosen was the training of a Deep Neural Network made of two layers: a Convolutional Neural Network, capable of extraction features from an image, and a Fully-Connected Neural Network in charge of the classification. Both networks were implemented with the help of the Caffe framework, developed by Berkeley University, avaliable in Python and C++ and trained with the extended Cohn-Kanade database. For said database, it was necessary to perform database augmentation, since Deep Neural Networks require a lot of data. Before training with the 7 facial expressions, there was a preliminary training with only three, to test the architetcure. The choosen architecture was the winner of the 2012 ImageNet Challange, called Alexnet, modified slightly to better fit our data. After results that supported the hipothesis of a good result, we did the training with the 7 expressions. The obtained results are displayed in confusion matrixes, which allows the analysis of the classifications in cases where they were right or wrong. In the end, we implemented a system which uses the neural network that we trained to recognize expressions in real time.

Keywords: Convolutional, Neural, Network, Facial, Expression, Recognition

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

MLP	*Multi-layer Perceptrion* (*Perceptron* de Múltiplas Camadas)
CNN	*Convolutional Neural Network* (Rede Neural Convolucional)
CK+	*Extended Cohn-Kanade Dataset*
VP	Verdadeiro Positivo
FP	Falso Positivo
VN	Verdadeiro Negativo
FN	Falso Negativo
S	Sensibilidade
E	Especificidade
VPP	Valor Preditivo Positivo
VPN	Valor Preditivo Negativo

SUMÁRIO

1 INTRODUÇÃO

O reconhecimento de expressões faciais é um processo natural para o ser humano. Essa é uma forma de comunicação não verbal que possibilita identificar o estado emocional de um indivíduo.

Entre duas pessoas, a comunicação não verbal enriquece a interação pela quantidade de informações que podem ser passadas. Criar um sistema inteligente com a habilidade de reconhecer e interpretar algumas dessas informações possibilitará melhores compreensões e entendimento dos interesses, intensões, humores e emoções das pessoas. Isso permite que sistemas possam ser mais interativos com seus usuários e com melhores respostas de uso, da mesma maneira que isso ocorre na interação entre uma comunicação "cara a cara" (CALVO; D'MELLO, 2010). Isso permite que a interação homem-máquina seja mais natural e facilitada. A detecção automática de expressões faciais é uma maneira não invasiva para atingir esse objetivo (LYONS et al., 1998).

Publicidade e propaganda é uma das áreas que conseguiriam tirar grandes proveitos de um sistema de reconhecimento de expressões faciais. Informações como nível de atenção e a relação entre a emoção causada e a emoção esperada podem ser usadas para determinar a qualidade de um anúncio, propaganda ou vídeo. Essas informações seriam adquiridas de maneira automática e em grande escada através de tal sistema. Outras áreas que se beneficiariam com essa forma de interação humano-máquina são: ambientes de ensino, entretenimento, atendimento ao cliente, jogos eletrônicos, segurança, software educacionais e monitoramento de motorista (MISHRA et al., 2015).

Do ponto de vista de um sistema computacional, o reconhecimento de expressões faciais é um problema mais complexo, que nos traz o desafio de variações anatômicas de indivíduo para indivíduo, além de mudanças nas próprias expressões. Sendo assim, é necessária uma técnica com grande poder de generalização mas que mantenha a especificidade para resolver tal problema. Considerando avanços recentes no desenvolvimento de arquiteturas de Redes Neurais Convolucionais, principalmente no desafio *ImageNet Large*

Scale Visual Recognition Challange(RUSSAKOVSKY et al., 2015), foi decidido implementar uma Rede Neural Profunda composta por uma Rede Neural Convolucional e uma Rede Neural Interconectada.

Em 2012, o ganhador do desafio citado foi uma arquitetura de Rede Neural Convolucional chamada *AlexNet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Esta arquitetura foi muito importante para renovar o interesse em Redes Neurais Convolucionais na comunidade científica, interesse que se mantém hoje, com ganhadores dos desafios em anos posteriores baseando-se nesta arquitetura. Este fato nos chamou a atenção e aproveitamos a oportunidade de aprendizado para experimentar uma arquitetura reconhecida, a *AlexNet*. Para a realização deste projeto, foi decidido buscar ferramentas para facilitar alterações na arquitetura e no processo de treinamento. Portanto, foi utilizado um conjunto de ferramentas disponibilizado pela faculdade Berkeley, chamado *Caffe*.

A Rede Neural Profunda precisa passar por um processo de treinamento, onde é fornecido exemplos de expressões faciais, onde no caso são imagens previamente categorizadas. Dessa forma, a rede neural é capaz de observar esses exemplos e "aprender"a classificar as imagens em suas respectivas categorias.

Para podermos realizar esse treinamento, é necessário um banco de dados, no nosso caso um banco de imagens frontais de faces humanas. Encontramos no *Cohn-Kanade* estendido um banco de dados que satisfez todas as nossas necessidades, possuindo imagens frontais de faces humanas divididas nas sete categorias que nos interessam: neutro, raiva, aversão, medo, felicidade, tristeza e surpresa. Após a definição deste, foi descoberta a necessidade de um aumento artificial no número de dados para a utilização no processo de treinamento já que Redes Neurais Convolucionais precisam de um grande número de exemplos para generalizar de forma satisfatória. Sendo tal processo longo e sendo o treinamento também um processo longo, escolheu-se a realização de um primeiro teste com três categorias ao invés de sete (Pilla Jr et al., 2016).

Quando o teste com somente três categorias teve resultados favoráveis à utilização da arquitetura, foram repetidos o processo de aumento artificial e treinamento com as sete categorias, desta vez utilizando-se da técnica da validação cruzada, para uma análise mais profunda dos resultados obtidos. Tais resultados são apresentados com uma tabela de confusão para cada uma das 10 redes resultantes deste processo de treinamento, bem como os filtros das mesmas. Além disso, utilizamos o resultado desse processo em um sistema de reconhecimento em tempo real, que possui seus resultados apresentados aqui.

O trabalho é apresentado de maneira a contextualizar as técnicas de aprendi-

zado de máquina, em especial Redes Neurais e Redes Neurais Convolucionais, em seguida discute-se o reconhecimento de expressões faciais, inclusive como foram definidas as categorias utilizadas. Por fim, apresentamos o desenvolvimento do trabalho, desde o processo de escolha do banco de dados, seu aumento artificial de dados (ambos na seção 5.1), a escolha da arquitetura e sua adequação aos dados disponíveis (na seção 3.5) e os dados colhidos durante e após o processo de treinamento da Rede Neural Profunda (na seção 6).

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Este projeto tem por objetivo o desenvolvimento de um sistema capaz de classificar a expressão facial presente em uma imagem estática da face humana, resultado em uma das sete emoções base: neutro, raiva, aversão (ou desgosto), medo, felicidade, tristeza e surpresa.

1.1.2 OBJETIVOS ESPECÍFICOS

Existem etapas a serem seguidas para que seja possível desenvolver um sistema para realizar a classificação de expressões faciais. Estas etapas são importantes para definir o funcionamento do sistema, parâmetros de entrada, parâmetros de saída, técnica abordada para o reconhecimento de padrões, arquitetura desenvolvida e processo de treinamento.

Antes de desenvolver o sistema, é necessário a obtenção de um banco de dados contendo imagens de expressões faciais. A normalização de algumas características dessas imagens também é importante, pois elas ajudam a determinar como a arquitetura deve ser desenvolvida para conseguir interpretar tais imagens com mais facilidade. Esse processo é detalhado na Seção 5.1.

O desenvolvimento do sistema de reconhecimento através de uma Rede Neural Convolucional, que será explicada posteriormente na Seção 5.2. Este desenvolvimento se dá pelo uso de um *deep learning framework* chamado *Caffe*.

Após a satisfazibilidade da arquitetura, será realizado um último treino da rede neural utilizando uma técnica mais demorada e eficaz para validação do sistema. Essa técnica se chama validação cruzada, apresentada na Seção 5.2.

O projeto foi concluído após serem realizados testes e análise de resultados da

última Rede Neural Convolucional treinada. Esses dados são usados para decidir se o sistema pode ser validado para uso de acordo com o objetivo na qual ele foi desenvolvido.

2 APRENDIZADO DE MÁQUINA

Para definirmos aprendizado de máquina, é necessário definir o que quer-se dizer com aprendizado dentro do nosso contexto. Uma definição utilizada é a que: "Um programa de computador aprende com a experiência E com relação a tarefas T, medido através do índice P se seu desempenho em T, medido por P, aumenta de maneira correlata com E". Esta definição é ampla, mas bem útil para entendermos aprendizado de máquina no contexto utilizado neste trabalho (MITCHELL, 1997, p. 50).

Aprendizado de máquina nos permite realizar tarefas muito diversas desde detecção de anomalias, transcrição (transformar uma imagem em texto, por exemplo), redução de ruído, entre outros (SEBASTIANI, 2002; EGMONT-PETERSEN; RIDDER; HANDELS, 2002).

Os índices de desempenho também variam com a tarefa. Um exemplo é a utilização de 0-1 *loss* (perda 0-1), que nos diz para cada exemplo se o algoritmo acertou a classificação ou não. Estes testes de desempenho devem ser utilizados em dados separados dos dados que serão utilizados para treinamento do modelo, para garantir a generalização do mesmo, já que o interesse é de que o algoritmo se comporte bem em situações do mundo real e não seja somente capaz de um bom desempenho com os dados já existentes no *dataset* escolhido.

O treinamento destes algoritmos pode ser definido de maneira ampla como supervisionado ou não supervisionado. A diferença entre os dois é simples: o conjunto de dados no supervisionado foi previamente categorizado, de maneira que cada um dos exemplos presentes possui um rótulo associado ao mesmo e durante o processo de aprendizagem, o resultado é comparado com o rótulo e reajustado baseado no erro apresentado. Já no não supervisionado, os dados não possuem rótulos e o algoritmo deve definir por si qual a característica que será utilizada para agrupar ou classificar os dados. Esta distinção não é sempre fácil de ser realizada e podem existir técnicas que ficam entre as duas definições, sendo chamados de métodos semi-supervisionados (ZHU et al., 2003).

Como dito, o desejado é que o algoritmo se comporte bem quando apresentado a dados nunca antes vistos e não somente naqueles que foi treinado. Esta característica é chamada de generalização. Para determinar se o algoritmo possui essa característica, é necessário que se separe partes do *dataset* para que o algoritmo não aprenda com esta parte. Desta maneira, pode-se simular uma experiência única e evitar que aconteça *overfitting*, termo utilizado quando o modelo não possui generalização. Isto ocorre quando o algoritmo é treinado de maneira exagerada, quando seus parâmetros são muito específicos para o *dataset* ou por falta de exemplos, entre outros. Isto leva o modelo a se comportar bem durante o treinamento, mas não ter utilidade real.

Um dos problemas da generalização é que, dados todos os problemas possíveis, o desempenho médio de dois algoritmos é idêntica quando aplicado a todos estes (WOLPERT; MACREADY, 1997). Este é o chamado *No Free Lunch Theorem* (Não existe almoço de graça, em tradução livre). Felizmente, não estamos interessados em um algoritmo que resolve qualquer problema e sim um problema específico.

Existem algumas decisões nessa etapa. O problema será abordado como classificação, os dados utilizados serão rotulados, não rotulados ou uma mistura entre os dois. Se a ideia é prever a possibilidade ou apontar uma faixa de valores possíveis. Essas decisões são baseadas no que será modelado, no *dataset* disponível, no tempo de treinamento que é desejado e do desejo do(s) projetista(s).

A decisão de utilizar Redes Neurais Convolucionais neste trabalho se deve ao desempenho recente das mesmas em relação a reconhecimento em imagens, demonstrado pelo desempenho das mesmas no ILSVCR de 2012 (RUSSAKOVSKY et al., 2015).

3 REDES NEURAIS E REDES NEURAIS CONVOLUCIONAIS

3.1 O NEURÔNIO

Sistemas computacionais de redes neurais tentam imitar o funcionamento de cérebros. Estes são capazes de receber dados e padrões inconsistentes, fazer sentido dessa informação e aprender a identificá-los futuramente para, por exemplo, reconhecer expressões faciais. Apesar de estarmos longe de compreender o funcionamento complexo de um cérebro por completo, já se conhece muitos princípios básicos a respeito da sua estrutura e funcionamento. Transmissores químicos dentro do cérebro alteram o potencial elétrico dos neurônios devido a um estímulo. Se esse potencial ultrapassa um limite, o neurônio dispara um pulso com certa força e duração pelo seu axônio. Esse disparo se chama de sinapse. Os axônios se dividem e se conectam a outros neurônios. Reproduzindo em um computador os princípios desta estrutura, pode-se criar um algoritmo capaz de aprender (MARSLAND, 2009).

O neurônio sozinho só realiza ou não sinapses de acordo com o estímulo recebido. Os mesmos não aprendem nada por conta própria. Portanto, antes de "construir um cérebro dentro de um computador", é preciso entender como um neurônio é capaz de "aprender". Como comentando anteriormente, os neurônios possuem conexões com outros neurônios e as sinapses tem forças específicas. A regra de Hebb diz que a força das sinapses é proporcional com a relação em que os neurônios conectados disparam simultaneamente. Porém quando neurônios conectados nunca disparam simultaneamente, suas conexões vão enfraquecendo ou até desaparecem. Esse condicionamento é conhecido como potencialização a longo prazo ou neuro-plasticidade (MARSLAND, 2009).

Para que seja possível construir um sistema computacional que imita o raciocínio de um cérebro, é preciso saber como criar uma modelagem matemática de neurônios e de conjuntos de neurônios. McCulloch e Pitts apresentaram, em 1943, uma modelagem de neurônio formado por um conjunto de entradas (x_i, representando estímulos externos obtidos de outros neurônios) com diferentes pesos ($\mathbf{W}_i$), um somador (equivalente ao

acúmulo de potenciais elétricos na célula) e uma função de ativação. Nesse modelo, as entradas (x_i) podem admitir qualquer número (inclusive valores negativos), e a saída (o) do neurônio são binárias, recebendo valores 1 ou 0. A função de ativação é encarregada de decidir se deve ocorrer uma sinapse para o dado conjunto de entradas (MARSLAND, 2009; BISHOP, 2006; THEODORIDIS; KOUTROUMBAS, 2003). A Figura 1 ilustra a modelagem matemática de um neurônio.

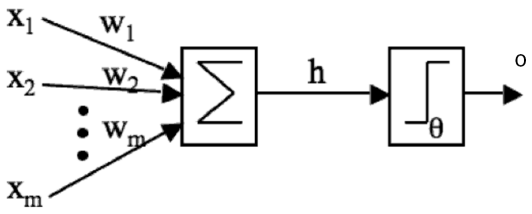


Figura 1: Modelo matemático de um neurônio
Fonte: (MARSLAND, 2009)

Quando um neurônio recebe, ou não, sinapses de outros neurônios, seus sinais são multiplicados pelos pesos de suas conexões e somados. A função de ativação é encarregada de decidir se o neurônio deve disparar ou não de acordo com a força resultante dos sinais (h). A Equação 1 é encarregada de calcular a força total (MARSLAND, 2009; BISHOP, 2006).

$$h = \sum_{i=1}^{n} \mathbf{W}_i x_i \tag{1}$$

Na modelagem de McCulloch e Pitts, a decisão da função de ativação é binária. Isso significa que se a amplitude do sinal (h) for acima de um nível, o neurônio dispara (MARSLAND, 2009; BISHOP, 2006), conforme representado pela Equação 2.

$$o = g(h) = \begin{cases} 1 : h > \theta \\ 0 : h \leq \theta \end{cases} \tag{2}$$

3.2 O *PERCEPTRON*

Quando um conjunto de neurônios de McCulloch e Pitts são organizados em uma rede para receber um conjunto de entradas com pesos definidos, tem-se uma rede *Perceptron*, ilustrada na Figura 2. No lado esquerdo, representados por círculos cinza, estão as

entradas. Todas elas estão conectadas com cada um dos neurônios de McCulloch e Pitts à direita, representados por círculos pretos. Essas conexões têm seus pesos específicos. As saídas dos neurônios também representam as saídas do sistema (MARSLAND, 2009; BISHOP, 2006).

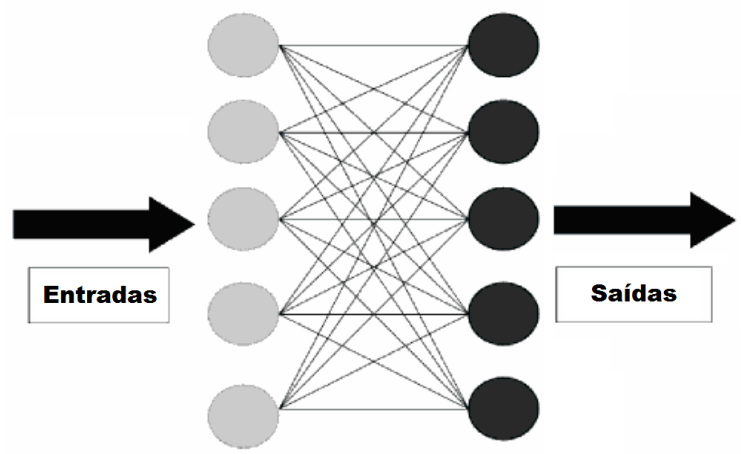


Figura 2: A rede *perceptron*
Fonte: Adaptado de (MARSLAND, 2009)

Nesta estrutura, o número de entradas é definido pela quantidade de conexões de cada neurônio, e o número de saídas é definido pela quantidade de neurônios (MARSLAND, 2009). Para melhor criar uma modelagem matemática da rede, é feita a representação dos pesos como $\mathbf{W}_{ij}$, sendo i a conexão do neurônio j. Computacionalmente, todos os pesos da rede neural podem ser representados por um *array* bidimensional. Esses pesos inicialmente são definidos com valores aleatórios pequenos. Depois, esses pesos sofrerão alterações, o que consiste no processo de treinamento, ou aprendizado, da rede neural. Quando as entradas da rede neural são alimentadas com um conjunto de informações, cada neurônio irá (ou não) realizar a sinapse. Ao visualizar os resultados obtidos nas saídas, é determinado se o padrão está correto ou não. O padrão desejado deve ser conhecida previamente ao treinamento, pois é a partir dessa informação que é decidido quais alterações são necessárias fazer para que a rede neural consiga reconhecer os padrões com que estamos a alimentado. Caso os neurônios tenham disparado corretamente, não é necessário fazer alterações nos pesos. Porém se algum neurônio disparou (ou não) quando não devia, deve-se fazer alterações para que suas sinapses tornem-se corretas (MARSLAND, 2009; BISHOP, 2006; THEODORIDIS; KOUTROUMBAS, 2003).

Primeiramente, antes de alterar os valores dos pesos, é preciso saber a força necessária que faça o neurônio disparar. Supondo um neurônio k, onde seus pesos ($\mathbf{W}_{ik}$)

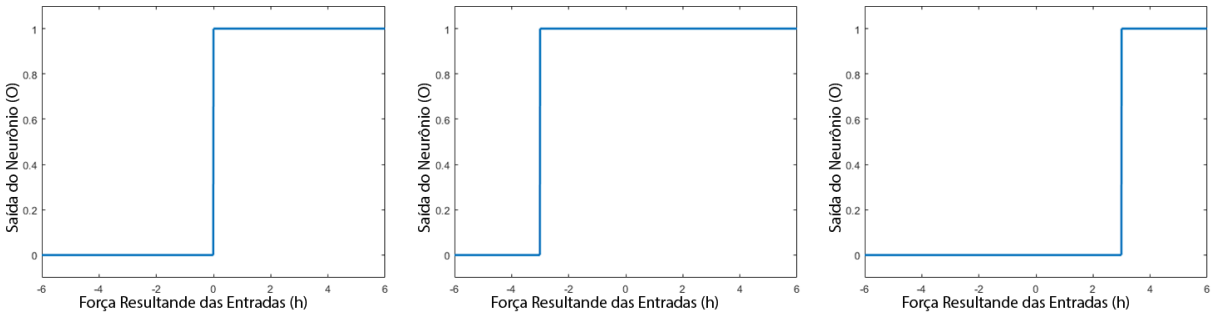
serão alterados, que atingiu força y_k, deveria atingir força t_k para ser ativado. O erro (e), diferença de força necessária para ativação do neurônio, é dada pela Equação 3. Isto também vale caso o neurônio não devesse disparar, com y_k maior que t_k (MARSLAND, 2009; BISHOP, 2006).

$$e = t_k - y_k \tag{3}$$

Como a entrada do neurônio pode receber valores negativos, é necessário multiplicar o valor de erro pelo valor da entrada para que o sinal seja corrigido antes de alterar o peso $\mathbf{W}_{ik}$. Isso impede que o valor erro divirja descontroladamente no caso de entradas negativas. Também é definida a velocidade de aprendizado da rede neural. Esse parâmetro é chamado de taxa de aprendizagem (η). Este valor é importante para que a rede neural consiga se adaptar para o quanto seus pesos devem mudar. Se η for muito grande, a rede neural pode alterar seus pesos de forma exagerada, fazendo que os pesos nunca convirjam. Se η for muito pequeno, o processo de treinamento fica muito demorado. Esse parâmetro não é necessariamente constante durante o treinamento, permitindo uma elasticidade durante o treinamento (MARSLAND, 2009; BISHOP, 2006) . A Equação 4 mostra a regra de atualização do valor de um peso na rede neural.

$$\mathbf{W}_{ij} \leftarrow \mathbf{W}_{ij} + \eta(t_j - y_j)x_i \tag{4}$$

Caso o valor x_i de entrada for 0, o peso $\mathbf{W}_{ij}$ não irá ser alterado. Nesse caso, uma saída é alterar o limite da função de ativação. Porém isso adiciona um peso computacional grande. Uma saída comumente adotada é a utilização do "*bias*"e deixando o limite de todas as funções de ativação igual a 0. O *bias* é uma entrada fixa diferente de 0, com peso ajustável como os outros. Quando um conjunto de entradas é 0, apenas o peso do *bias* irá mudar. Portanto, como o limite da função de ativação é 0, o valor obtido multiplicando o *bias* com o seu peso é o novo limite da função. Comumente, o valor de *bias* é -1, fazendo que apenas seu peso seja o limite (MARSLAND, 2009; BISHOP, 2006). A Figura 3 apresenta diferentes gráficos para o valor de saída da função de ativação em relação com o potencial de entrada, com diferentes valores de pesos da conexão de *bias*, porém com *bias* fixo em -1.



(a) Peso do *bias* $= 0$ (b) Peso do *bias* $= -3$ (c) Peso do *bias* $= 3$

Figura 3: A função de ativação para diferentes pesos da conexão de *bias* e entrada de *bias* $= -1$ (Saída do neurônio vs potencial de entrada)

Em resumo, o treino de um *Perceptron* consiste em:

1. Inicialização - Definição dos pesos $\mathbf{W}_{ij}$ aleatoriamente com valores pequenos.
2. Aprendizado - Processo iterativo de atualização dos pesos $\mathbf{W}_{ij}$.
   - Computação da ativação dos neurônios utilizando a Equação 5.

$$y_j = \begin{cases} 1 : \sum \mathbf{W}_{ij} x_i > 0 \\ 0 : \sum \mathbf{W}_{ij} x_i \leq 0 \end{cases} \tag{5}$$

   - Atualização dos valores dos pesos da rede neural utilizando a Equação 6.

$$\mathbf{W}_{ij} \leftarrow \mathbf{W}_{ij} + \eta(t_j - y_j)x_i \tag{6}$$

3.3 O *PERCEPTRON* DE MÚLTIPLAS CAMADAS

A arquitetura apresentada na Seção anterior (3.2) de rede neural só permite a solução de problemas linearmente separáveis, o que não ocorre na maior parte dos casos. Para que seja possível resolver problemas não-lineares, o modelo que deve ser adotado é conhecido como *perceptron* de múltiplas camadas (MLP - *Multi-layer Perceptrion*), nele são adicionados novos neurônios em camadas dentro da rede neural, entre as camadas de entrada e de saída, criando assim conexões que saem de neurônios e entram em outros neurônios. Essas novas camadas de neurônios são chamadas de camadas intermediárias ou camadas escondidas (*hidden layer*). A Figura 4 representa uma rede *perceptron* de múltiplas camadas. Nesta representação, existe uma única camada intermediária, mas é possível fazer implementações de mais camadas (MARSLAND, 2009; THEODORIDIS; KOUTROUMBAS, 2003).

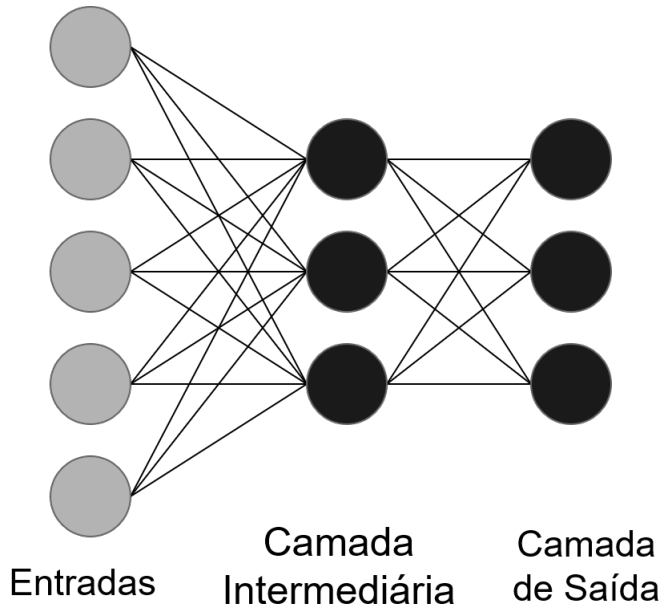


Figura 4: A rede *perceptron* de múltiplas camadas
Fonte: Adaptado de (MARSLAND, 2009)

Apesar do treinamento também consistir de alterações nos pesos das conexões da rede neural com objetivo de minimizar o erro, a maneira com que isso é feito não é a mesma que no perceptron, onde é calculado apenas o erro de potencial para que um neurônio ativasse quando necessário. No caso dos *Perceptrons* de Múltiplas Camadas, esse valor de erro pode não significar nada. Supondo que existam dois neurônios conectados com erros e seus erros são de mesma magnitude porém de sinais opostos. A soma desses erros é 0, fazendo que a rede acredite que não existe erro. Para solucionar esse problema, é utilizada outra função para calcular o erro de potencial de disparo de um neurônio. A solução mais utilizada para que seja possível detectar erro em uma sequência de neurônios é a soma dos erros quadráticos. A Equação 7 descreve essa função, onde ξ é soma dos erros quadráticos, y_k é a força atingida pelo neurônio e t_k é a força necessária para ativação (MARSLAND, 2009).

$$\xi = \frac{1}{2}\sum_{k=1}^{n}(t_k - y_k)^2 \tag{7}$$

Mesmo assim, esse cálculo não é o suficiente para determinar exatamente em quais camadas estão ocorrendo erros. Foi em 1986 que uma solução foi introduzida por Rumelhart, Hilton e McClelland (RUMELHART; HINTON; WILLIAMS, 1986). Essa solução já existia, porém não havia sido aplicada em redes neurais até o momento, a retropro-

gressão de erro, com a retropropagação de erro é possível identificar em quais neurônios e em quais camadas ocorrem erros. Anteriormente, era fácil identificar o neurônio que apresentava problemas porque cada um era conectado diretamente com a saída do sistema e nunca entre si. Com os neurônios tendo interconexão, se a a resposta de um neurônio de saída está incorreta, o problema pode estar no próprio neurônio ou na interconexão de neurônios que chegam até ele. A solução utilizada para conseguir identificar o erro em diferentes níveis da rede neural faz aplicação da Regra da Cadeia (MARSLAND, 2009), descrita pela Equação 8.

$$\frac{\partial \xi}{\partial \mathbf{W}_{jk}} = \frac{\partial \xi}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial h_k}\frac{\partial h_k}{\partial \mathbf{W}_{jk}} \tag{8}$$

Dessa forma, é possível isolar os neurônios da camada de saída e das camadas intermediárias e identificar o erro de seus pesos. Como o sistema agora trabalha com diferenciação, existe um problema em utilizar uma função de ativação que assume somente dois valores (0 e 1), sendo necessário trabalhar com continuidade. Por esse motivo, a função de ativação é substituída pela função Sigmoide, descrita pela Equação 9 e Figura 5. Com isso, as saídas dos neurônios passam a ter valores contínuos entre 0 e 1. (BISHOP, 2006).

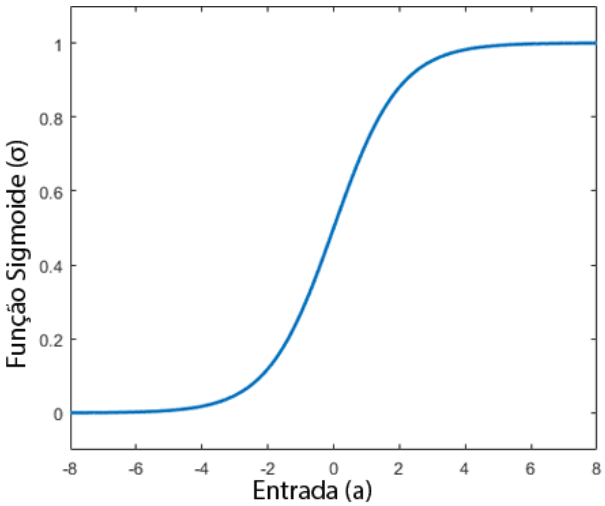


Figura 5: Gráfico da Função Sigmoide

$$\sigma(a) = \frac{1}{1 + exp(-a)} \tag{9}$$

Este é um dos motivos pelos quais a utilização dessa função torna o processo mais prático. A derivada dessa função é facilmente calculável e o resultado é uma função de si mesma, representada pela Equação 10 (BISHOP, 2006).

$$\frac{\partial \sigma(a)}{\partial a} = \sigma(a)(1 - \sigma(a)) \tag{10}$$

Dessa maneira, Rumelhart, Hilton e McClelland (1986) conseguiram definir equações que descrevem os valores de erro. A Equação 11 apresenta o cálculo para o valor de erro nos neurônios da camada de saída. A Equação 12 para o valor de erro nos neurônios das camadas escondidas. Nessas equações, δ_{ok} (erro de *output*) representa o erro no neurônio k na camada de saída, δ_{hj} o erro no neurônio j em uma camada intermediária (*hidden layers*) e a_j é o resultado do neurônio j utilizando a função Sigmoide (MARSLAND, 2009; BISHOP, 2006).

$$\delta_{ok} = (t_k - y_k)y_k(1 - y_k) \tag{11}$$

$$\delta_{hj} = a_j(1 - a_j)\sum_k \mathbf{W}_{jk}\delta_{ok} \tag{12}$$

Agora é possível definir a regra que atualiza os valores de qualquer neurônio na rede neural. A Equação 13 apresenta essa regra. O erro δ pode ser δ_{ok} ou δ_{hj}, dependendo de qual camada pertence o neurônio que está tendo seus pesos ajustados. Também, o valor de entrada x_j pode corresponder à saída de outros neurônios, podendo ser representada por a_i (MARSLAND, 2009; BISHOP, 2006).

$$\mathbf{W}_{ij} \leftarrow \mathbf{W}_{ij} + \eta\delta_j x_j \tag{13}$$

Como a variação que será aplicada nos pesos é um produto de elementos que incluem a entrada, é necessário adicionar entradas de *bias* na rede neural. Sua funcionalidade é idêntica aos *Perceptrons*. Porém, por se tratar de uma função contínua, é realizada a translação da função ao invés de modificar o limite de ativação. Isso permite que a saída do neurônio seja maior com menores potenciais aplicados nele (MARSLAND, 2009).

Em resumo, o treino de um *Perceptron* de Múltiplas Camadas consiste em:

1. Inicialização - Definição dos pesos x_{ij} aleatoriamente com valores pequenos.
2. Aprendizado - Processo iterativo de atualização dos pesos x_{ij}.
   - **Processo progressivo** - Computação da ativação dos neurônios utilizando a Equações 14. Lembrando que alguns neurônios estão conectados com as

entradas da rede neural, enquanto outros estão conectados às saídas de outros neurônios. Portanto, o valor de entrada desse neurônio pode ser o valor de saída de outro.

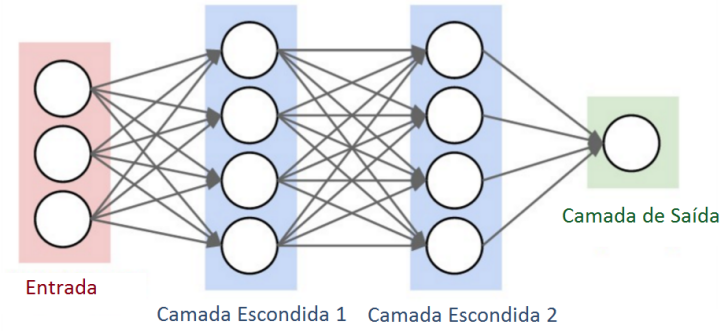
$$y_j = \sigma(\sum x_{ij} y_i) \tag{14}$$

- **Processo regressivo** - Primeiramente é calculado o valor de erro de cada neurônio da rede neural, utilizando a Equação 11 para os neurônios de saída e a Equação 12 para os neurônios nas camadas escondidas. Por fim é feita a atualização dos valores dos pesos da rede neural utilizando a regra na Equação 13.

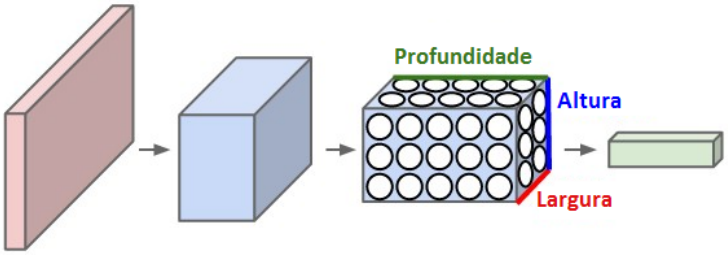
3.4 REDES NEURAIS CONVOLUCIONAIS

Redes Neurais Convolucionais (em inglês *Convolutional Neural Network* - CNN) são nomeadas dessa maneira porque realizam a operação matemática de convolução (ou correlação). Isso torna a rede neural em um tipo especializado para processar dados em matrizes, permitindo que seja utilizada com imagens (LECUN; BENGIO, 1995; GOODFELLOW; BENGIO; COURVILLE, 2016b). As CNNs são similares com as redes neurais interconectadas discutidas anteriormente. Ambas são feitas de neurônios com pesos adaptáveis e *bias*. A diferença mais importante é a organização desses neurônios. No caso do *Perceptron* de Múltiplas Camadas, cada neurônio era conectado com todas as entradas ou neurônios da camada anterior. No caso da CNN, o neurônio só se conecta com neurônios ou entradas de posições vizinhas, o que torna a operação de cálculo do potencial de ativação em uma operação de convolução (STANFORD, 2016). Cada camada da CNN é dividida em três partes: Camada Convolucional, Camada *ReLU* (*Rectified Linear Units*) e Camada de *Pooling* (STANFORD, 2016).

A camada Convolucional têm a particularidade de ter os neurônios organizados em 3 dimensões: largura, altura e profundidade. O termo "profundidade", nesse caso, não se refere à profundidade da rede neural (número de camadas da rede), mas se refere à terceira dimensão de neurônios dentro de uma única camada da rede neural. A figura 6 compara a organização de neurônios em uma rede neural vista anteriormente com a de uma camada convolucional (STANFORD, 2016).



(a) Organização de neurônios em um *Perceptron* de Múltiplas Camadas



(b) Organização de neurônios em uma camada convolucional

Figura 6: Comparação da organização de neurônios em um *Perceptron* de Múltiplas Camadas com uma camada convolucional

Fonte: Adaptado de (STANFORD, 2016)

Esta camada consiste de "filtros que aprendem", também chamados de *kernels*. Suas larguras e alturas, apesar de iguais entre os filtros da mesma camada, podem ser definidas como desejadas. A profundidade desses filtros deve ser igual à profundidade da entrada da camada. Por exemplo, imagens RGB na entrada requerem uma camada de filtros $N \times N \times 3$, ou seja, profundidade três (STANFORD, 2016).

Durante o processo progressivo, cada filtro é deslizado ao longo da imagem de entrada, calculando o produto de um elemento da imagem com um elemento do filtro em qualquer posição, ou seja, é efetuada a convolução entre o filtro e a imagem. Como saída desse processo, é gerado um mapa de ativação bidimensional que fornece as respostas desse filtro em cada posição (STANFORD, 2016; GOODFELLOW; BENGIO; COURVILLE, 2016b). A Equação 15 é a representação matemática da convolução em duas dimensões, onde $\mathbf{I}$ é a imagem, $\mathbf{K}$ é o filtro e $\mathbf{S}$ é o resultado da convolução.

$$\mathbf{S}(i,j) = \sum_m \sum_n \mathbf{I}(i+m, j+n)\mathbf{K}(m,n) \tag{15}$$

Durante o processo de aprendizagem, os filtros são ajustados de maneira que detectam quando algum tipo de característica visual na imagem, como uma borda ou uma textura. Os mapas de ativação, gerados por cada filtro, são agrupados ao longo da dimensão de profundidade, produzindo uma saída. Se a camada tem 12 filtros, é gerada uma imagem de profundidade 12 (STANFORD, 2016). Conforme essas camadas são empilhadas, esse conjunto de camadas consegue detectar características mais complexas (GOODFELLOW; BENGIO; COURVILLE, 2016b).

Utilizando a interpretação de neurônios, cada *pixel* do mapa de ativação gerado é a saída de um neurônio. Cada neurônio está conectado com uma pequena região de entrada e os pesos de todos os neurônios na mesma profundidade são iguais, já que eles representam o mesmo filtro realizando a multiplicação em outra região da entrada. A Figura 7 ilustra a conexão de uma região da imagem de entrada com os neurônios de alguns filtros da camada de convolução.

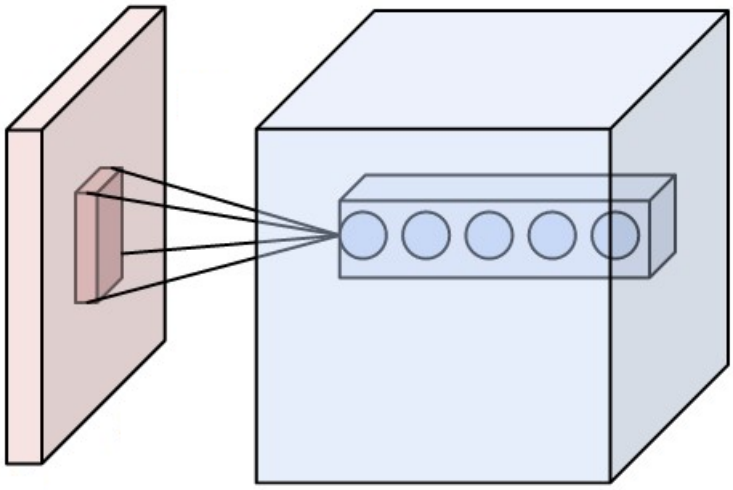


Figura 7: Conexão de uma região da imagem de entrada com os neurônios de alguns filtros da camada de convolução

Fonte: Adaptado de (STANFORD, 2016)

Nesta camada, é necessário definir três hiperparâmetros que estabelecem o volume da saída da camada: i) A profundidade da saída: corresponde com a quantidade de filtros utilizados, onde cada um aprende a identificar uma característica diferente da imagem de entrada da camada. ii) O passo (*stride*): define o espaço deslocado pelo filtro entre um instante do deslizamento e outro. Quando o passo é 1, o filtro desliza 1 *pixel* por vez. iii) O tamanho do preenchimento por zeros (*zero-padding*): em alguns casos é conveniente preencher ao redor da imagem de entrada com zeros (STANFORD, 2016).

A camada *ReLU* contêm todas as funções de ativação dos neurônios. Algumas interpretações não consideram a *ReLU* como uma camada na rede neural, mas somente como a função de ativação incorporada na camada convolucional, apesar de serem implementadas como etapas separadas. Nessa camada é aplicada uma função não linear em uma região do mapa de ativações sem alterar o tamanho da saída da camada convolucional (STANFORD, 2016). Comumente é utilizada a função $max(0, x)$, onde é encontrado o maior valor na região desejada (GOODFELLOW; BENGIO; COURVILLE, 2016b). A Figura 8 exemplifica o funcionamento da camada *ReLU* utilizando a função *max*.

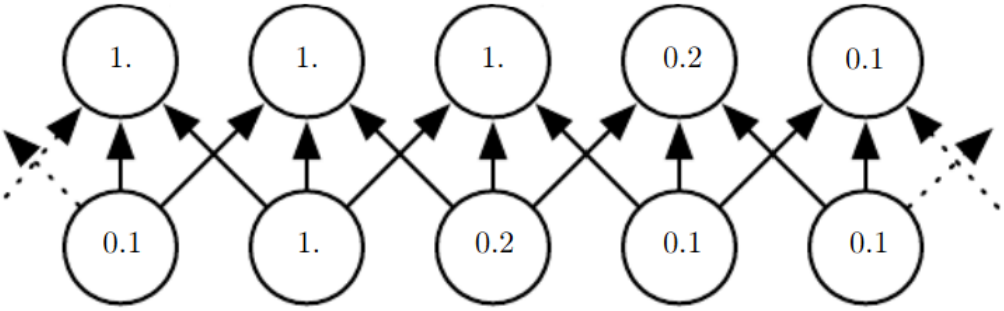


Figura 8: Exemplo do funcionamento da camada *ReLU* utilizando a função *max*
Fonte: Adaptado de (GOODFELLOW; BENGIO; COURVILLE, 2016b)

A camada de *Pooling* realiza a subamostragem da sua imagem de entrada. Dessa maneira, o mapa de ativação obtido pela camada *ReLU* é resumido com as informações mais relevantes em um determinado espaço. Isso também permite pequenas variâncias espaciais da imagem de entrada sejam ignoradas. Usa-se o termo *Max Pooling* quando existe uma camada *ReLU* de função *max* e uma camada de *Pooling* conectadas nesta respectiva ordem (STANFORD, 2016). A Figura 9 exemplifica o processo de *Max Pooling*.

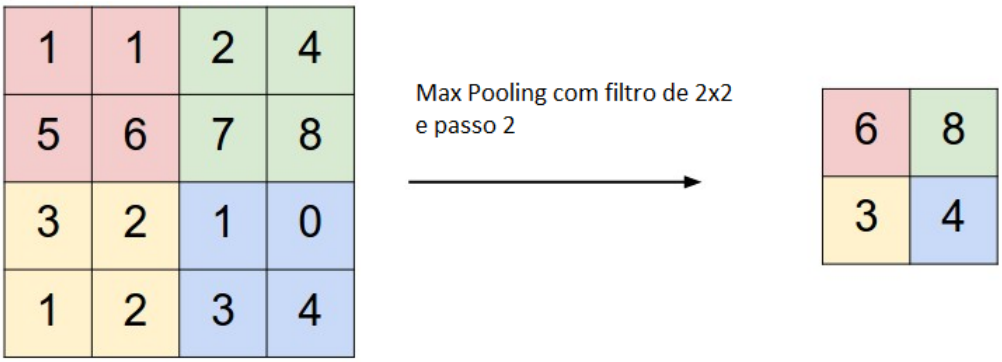


Figura 9: Exemplo do processo realizado pela camada *Max Pooling*
Fonte: Adaptado de (STANFORD, 2016)

A saída de uma Rede Neural Convolucional é um mapa de ativações das características de uma imagem de entrada. Para que seja possível fazer a classificação dessas características extraídas, a Rede Neural Profunda faz uso da Rede Neural Interconectada. Ou seja, em uma Rede Neural Profunda, A Rede Neural Convolucional apenas extrai as características de uma imagem de entrada e a Rede Neural Interconectada as classifica (GOODFELLOW; BENGIO; COURVILLE, 2016b). A Figura 10 ilustra uma Rede Neural Profunda composta pela Rede Neural Convolucional e um Rede neural Interconectada.

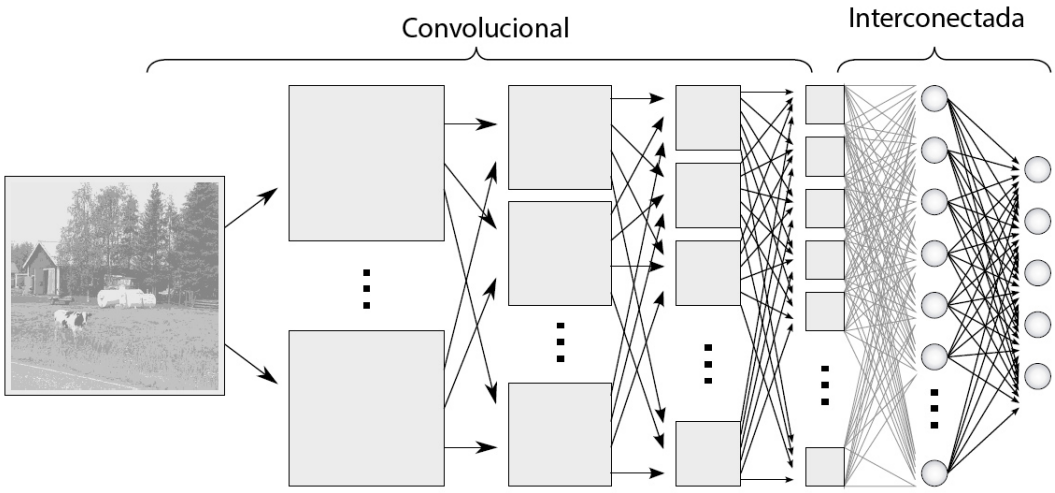


Figura 10: Rede Neural Profunda composta pela Rede Neural Convolucional e pela Rede Neural Interconectada

Fonte: Adaptado de (BONN, 2010)

3.5 ARQUITETURA ALEXNET

A *AlexNet* é uma arquitetura de Rede Neural Profunda formada por uma Rede Neural Convolucional e uma Rede Neural Interconectada, com oito camadas que passam pelo processo de treinamento. As cinco primeiras camadas são convolucionais com camadas de *max pooling* entre essas cinco camadas e em seguida três camadas interconectadas. Essa profundidade da Rede tem um impacto direto na sua acurácia sendo que a retirada de qualquer uma das cinco camadas convolucionais implica em um aumento significativo do erro (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). A arquitetura da *AlexNet* está representada na Figura 11.

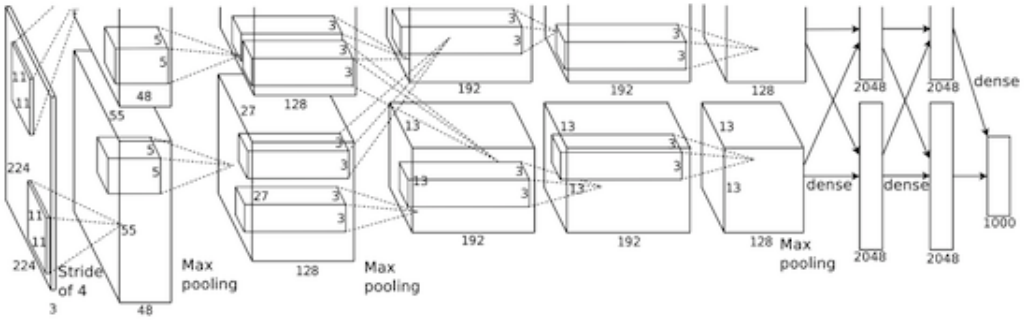


Figura 11: Arquitetura da AlexNet
Fonte: (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

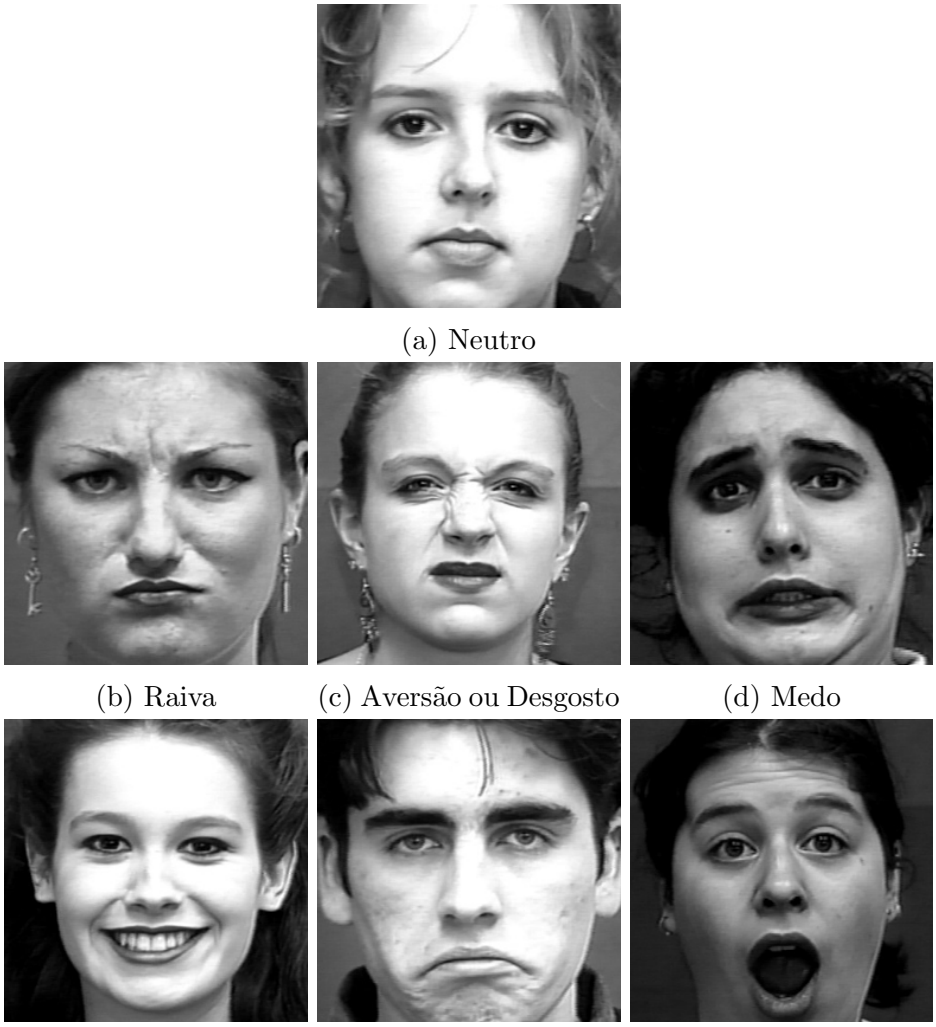
A *AlexNet* teve resultado surpreendente no *ImageNet* (*ImageNet Large Scale Visual Recognition Challange*) em 2012 (RUSSAKOVSKY et al., 2015), com uma diminuição do erro de 45.7% para 37.5% com a rede classificando mil classes de imagens (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Desde então, a *AlexNet* virou referência de arquitetura de Redes Neural Profunda para classificação de imagens.

4 RECONHECIMENTO DE EXPRESSÕES FACIAIS

Os seres humanos realizam o reconhecimento de expressões faciais naturalmente como uma forma de comunicação não verbal possibilitando identificar o estado emocional de um indivíduo. Apesar de existirem outras formas de comunicação não verbal, usadas com o mesmo objetivo, como gestos, tons de voz e postura, as expressões faciais são reconhecidas como a forma mais importante (BUSSO et al., 2004).

Os estudos sobre expressões faciais tem seus primeiros grandes resultados em pesquisas de Darwin (DARWIN, 1899), onde mostra-se o impacto da comunicação não verbal na evolução de espécies. De acordo com a teoria de Darwin, a comunicação através da identificação de emoções em expressões faciais é muito mais rápida do que a comunicação verbal, o que traz grandes vantagens para a espécie humana. Sob esta perspectiva evolutiva, as expressões faciais são universais para todos os seres humanos. Esta teoria foi reconhecida em pesquisas posteriores e é aceita pela psicologia, sendo classificada em sete expressões faciais: neutro, raiva, aversão (ou desgosto), medo, felicidade, tristeza e surpresa, (BUSSO et al., 2004; KELTNER et al., 2003); ilustradas na Figura 12.

Apesar das sete expressões faciais citadas serem mundialmente reconhecidas, existem diferenças culturais que influenciam a maneira com que são apresentadas. Na cultura ocidental, pessoas têm a tendência de negar suas expressões e seus próprios sentimentos. Enquanto em países do Leste Asiático, é mais importante que as expressões emocionais sejam controladas, sendo importante para manter relações harmoniosas, de modo que os indivíduos não impõem seus sentimentos sobre os outros. Isso foi visto em resultados de pesquisas que indicam que japoneses tendem em mascarar suas emoções negativas como: aversão, medo, tristeza e raiva, mas apresentam essas expressões livremente quando estão sozinhos (MARKUS; KITAYAMA, 1991; HEINE et al., 1999). Estudos indicam que expressões faciais podem ser controladas, porém não com sucesso absoluto. O músculo orbicular do olho (ao redor dos olhos) é mais difícil de controlar que o zigomático maior (ao redor da boca) (DUCHENNE; CUTHBERTSON, 1990), duas regiões importantes para realização das expressões faciais.



(a) Neutro

(b) Raiva (c) Aversão ou Desgosto (d) Medo

(e) Felicidade (f) Tristeza (g) Surpresa

Figura 12: Expressões faciais básicas.
Fonte: Adaptado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

Existem alguns fatores que estão dentro da capacidade humana de reconhecimento das expressões faciais mas dificultam a implementação em um sistema inteligente. Alguns exemplos são: fisionomia, identidade facial, personalidade, etnia, iluminação e posição da cabeça (SAMAL; IYENGAR, 1992; TURK; PENTLAND, 1991). Várias regiões do rosto podem ser analisadas para tentar determinar a expressão facial. As principais são as regiões da boca, sobrancelha e olhos, porém outras regiões, como a região das bochechas e nariz podem auxiliar na classificação. Dentro dessas regiões podem ser analisadas posições relativas de certos elementos (movimento), mudanças de textura, mudanças de tamanho, mudanças na coloração da pele e até mesmo o tempo decorrido de tais mudanças (LYONS et al., 1998).

Existem algumas abordagens que podem ser utilizadas na implementação de um

sistema computacional capaz de classificar expressões faciais. Alguns trabalhos fizeram a classificação a partir de imagens estáticas (BARTLETT et al., 1996; PADGETT; COTTRELL, 1995; GU et al., 2012), onde foi preciso realizar a análise morfológica de uma imagem estática e comparar os resultados com padrões esperados de cada expressão. Nessa abordagem, apesar de obter resultados satisfatórios, alguns detalhes podem ser confundidos com outras expressões pela falta de informações de movimento que são perdidas ao utilizar apenas imagens estáticas. Outros trabalhos fazem a comparação morfológica entre *frames* (LYONS et al., 1998), o que permitiu saber a posição e tamanho relativos de algumas regiões do rosto que se alteram entre uma expressão e outra. Apesar de complementar algumas informações, morfologias sutis e movimentos da cabeça são difíceis de serem analisadas e classificadas. A abordagem de fluxo óptico, utilizado em outros trabalhos (BARTLETT et al., 1996; KENJI, 1991; YACOOB; DAVIS, 1996), conseguem extrair mais informações que as abordagens anteriores. Essa abordagem não deixa de ser uma comparação de *frames*, porém, ao determinar o fluxo óptico, é possível analisar qual o movimento realizado pelos músculos do rosto, o tempo decorrido do movimento e movimentos da cabeça, permitindo que áreas mais sutis sejam analisadas mais facilmente.

5 DESENVOLVIMENTO

Esta seção tem por objetivo apresentar o processo de desenvolvimento do projeto, que consiste no desenvolvimento de um sistema inteligente capaz de classificar imagens estáticas contendo expressões faciais. Este processo foi dividido em duas etapas:

1. Obtenção e melhoramento do banco de dados com expressões faciais;
2. Desenvolvimento e treino da Rede Neural Convolucional.

5.1 BANCO DE DADOS

Para que seja possível treinar uma CNN e obter níveis de acurácia aceitáveis, é necessário um banco de dados com número considerável de imagens. A quantidade mínima de imagens necessárias varia dependendo da aplicação, complexidade, número de classes e variabilidade da informação. Enquanto algumas redes neurais são treinadas com um banco de aproximadamente 15 mil imagens (WU; PENG; CHEN, 2016), outras são treinadas com aproximadamente 3 milhões de imagens (LEE; PARK; KIM, 2016).

Neste projeto, foi utilizado o banco de dados *Extended Cohn-Kande Dataset* (CK+) (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010), contendo 10558 imagens de 123 pessoas. Este banco contém sequências de imagens ilustrando a transição do estado neutro para uma expressão facial. Para os fins desse projeto, as imagens com uma expressão facial ambígua, ou seja, na transição de uma expressão para outra, foram retiradas do banco. O banco original contém ainda uma 8ª expressão (desprezo), também retiradas do banco. A decisão de não incluir a 8ª expressão facial veio pelo fato do banco de dados conter um número muito baixo de imagens representando essa expressão. Após a remoção dessas imagens, o banco teve 3406 imagens restantes. O banco contém algumas peculiaridades que fizeram com que fosse necessário um processo de normalização das imagens. A grande maioria das suas imagens estão em escada de cinza. O posicionamento das pessoas nas imagens não segue nenhum padrão (algumas pessoas estão mais próximas,

algumas estão no centro da imagem e algumas mostram os ombros). Essas variações são informações irrelevantes para o objetivo do projeto, que é classificar expressões faciais, mas podem afetar o treinamento da CNN.

Primeiramente, todas as imagens foram convertidas para escala de cinza. Em seguida, foi desenvolvido um *script* capaz de extrair uma regiões de interesse quadrada contendo apenas as cabeças dos indivíduos, utilizando o algoritmo de Viola-Jones (VIOLA; JONES, 2004) presente no MATLAB para detectar faces (MATLAB, 2010; MATHWORKS, 2016). Por fim, todas as imagens quadradas foram redimensionadas para o tamanho 256x256 *pixels*. Assim, todas as imagens têm os rostos centralizados em imagens de tamanho 256x256 *pixels* em escala de cinza (8 bits por pixel). Como o redimensionamento das imagens manteve a proporção entre largura e comprimento, não houve alterações na morfologia das imagens. A Figura 13 exemplifica as normalizações feitas.

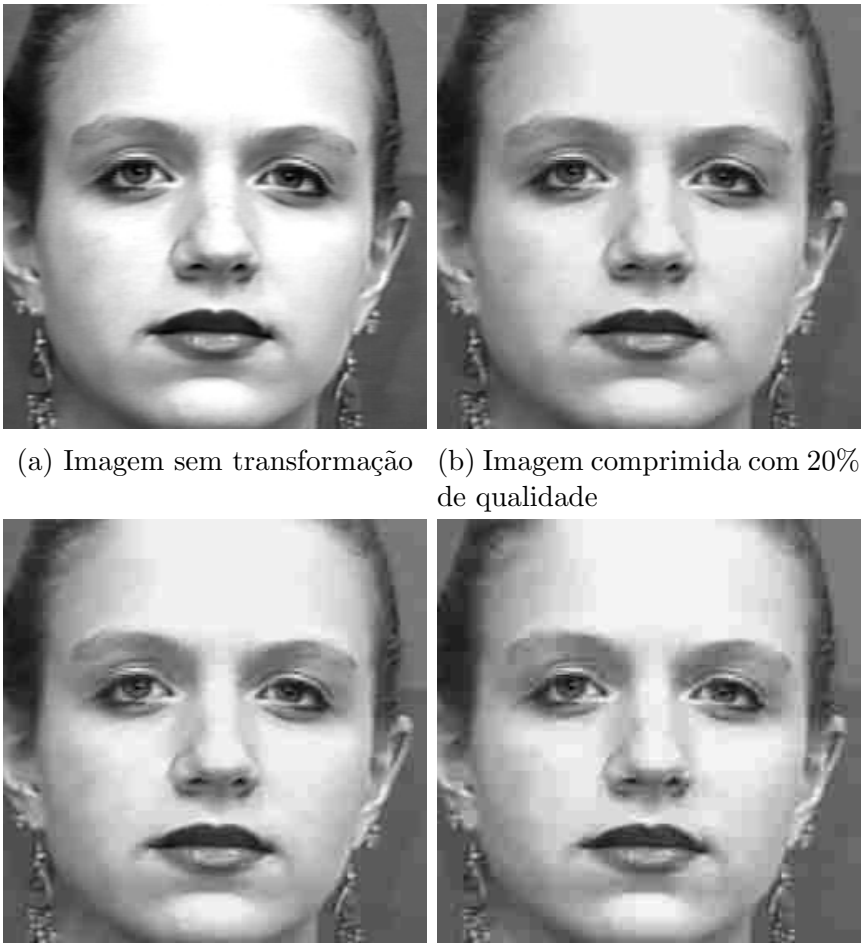
(a) Imagem do banco CK+ (b) Imagem após processo de normalização

Figura 13: Exemplo antes e depois do processo de normalização de imagem
Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

Para que o treinamento da rede neural pudesse atingir melhores resultados de acurácia e não sofresse com *overfitting*, foi empregada uma técnica de aumento de banco de dados chamada de "*Data Augmentation*" ou "*Database Augmentation*". *Database augmentation* consiste em um método capaz de diminuir o *overfitting* no treinamento de um sistema inteligente. Fazendo pequenas transformações nos dados do banco, porém mantendo as informações importantes para o treino, é possível gerar novos dados relevantes de maneira artificial que podem ser utilizados no treinamento. No caso de um banco de dados contendo imagens, são realizadas pequenas transformações para gerar novas imagens que podem ser classificadas na mesma classe que a imagem original (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). As transformações utilizadas para *database augmenta-*

tion foram:

- **Compressão JPG**: Realizando compressão em uma imagem, acrescenta traços característicos de arquivos de imagens de baixa qualidade e com compressão JPG. Nesse projeto, foram utilizados três diferentes configurações de compressão: qualidade de 20%, 15% e 10%. A Figura 14 representa o resultado desse processo.



(a) Imagem sem transformação

(b) Imagem comprimida com 20% de qualidade

(c) Imagem comprimida com 15% de qualidade

(d) Imagem comprimida com 10% de qualidade

Figura 14: Compressão

Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

- **Equalização de Histograma**: Através desse processo automático, a intensidade dos níveis de pixeis da imagem, alterando o contraste e iluminação. A equalização de histograma utilizou a configuração padrão do MATLAB. A Figura 15 representa o resultado desse processo.

(a) Imagem sem transformação (b) Depois da equalização do histograma

Figura 15: Equalização de Histograma

Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

- **Adição de Ruído Gaussiano**: Ruídos podem ocorrer por diversos motivos (qualidade da câmera, ambiente com baixa iluminação, entre outros). Foi adicionado ruído Gaussiano nas imagens, com média 0 e variância 0.01. A Figura 16 representa o resultado desse processo.

(a) Imagem sem transformação (b) Depois da adição de ruído

Figura 16: Adição de Ruído

Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

- **Pirâmide**: Essa técnica consiste em um processo de redução e aumento da imagem que suaviza a imagem. Em cada camada, a imagem resultante da camada anterior (imagem de entrada no caso da primeira camada) é redimensionada pela metade e depois interpolada para o tamanho original. Quanto maior o número de camadas, mais intensificada fica a suavização. Neste projeto, foram criadas apenas duas camadas de suavização. A Figura 17 representa o resultado desse processo.

(a) Imagem sem transformação

(b) Primeira camada da pirâmide (c) Segunda camada da pirâmide

Figura 17: Pirâmide

Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

- ***Grayslicing* ou Múltiplos *Thresholds***: Este processo cria diversos *thresholds* de escala de cinza, alterando os níveis de cinza presentes e modificando o contraste da imagem. Nesse projeto, foram utilizados quatro diferentes configurações equidistantes de níveis de *thresholds*, com 128, 64, 32 e 16 níveis. A Figura 18 representa o resultado desse processo.

(a) Imagem sem transformação

(b) 128 níveis de *thresholds*

(c) 64 níveis de *thresholds*

(d) 32 níveis de *thresholds*

(e) 16 níveis de *thresholds*

Figura 18: *Grayslicing* ou Múltiplos *Thresholds*

Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

- **Espelhamento**: Consiste em espelhar a imagem no seu eixo Y. Não foi considerado espelhar no eixo X porque a maneira mais comum de se encontrar rostos é com orientação vertical para cima. A Figura 19 representa o resultado desse processo.

(a) Imagem sem transformação (b) Imagem expelhada

Figura 19: Espelhamento
Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

Essas transformações aplicadas no banco foram combinadas entre si, aumentando ainda mais o banco, resultando nas seguintes transformações:

- Espelhamento;
- Compressão;
- Compressão-Espelhamento;
- Adição de Ruído;
- Adição de Ruído-Espelhamento;
- Pirâmide;
- Pirâmide-Espelhamento;
- Grayslicing;
- Grayslicing-Espelhamento;
- Equalização de Histograma;
- Equalização de Histograma-Espelhamento;

- Equalização de Histograma-Compressão;
- Equalização de Histograma-Compressão-Espelhamento;
- Equalização de Histograma-Adição de Ruído;
- Equalização de Histograma-Adição de Ruído-Espelhamento;
- Equalização de Histograma-Pirâmide;
- Equalização de Histograma-Pirâmide-Espelhamento;
- Equalização de Histograma-Grayslicing;
- Equalização de Histograma-Grayslicing-Espelhamento.

O tamanho final do banco de dados após a conclusão o processo de *database augmentation* foi de 44 vezes, ou seja, resultando em 149864 imagens no banco de dados final. Infelizmente, o banco de dados está desbalanceado, significando que cada expressão facial não contém o mesmo número de imagens que as represente. A Tabela 1 detalha a quantidade de imagens para cada expressão no banco de dados antes e depois da expansão.

Tabela 1: Detalhamento de número de imagens contidas no banco.

Expressão	Nº de imagens	Nº de imagens após expansão	%
Neutro	636	27984	18,67%
Raiva	418	18392	12,27%
Aversão	356	15664	10,45%
Medo	256	11264	7,52%
Felicidade	882	38808	25,90%
Tristeza	339	14916	9,95%
Surpresa	519	22836	15,24%
Total	3406	149864	100%

5.2 REDE NEURAL CONVOLUCIONAL

O primero passo foi adaptar a arquitetura AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) para funcionar em nosso problema e com o banco de dados disponível. Foram feitos ajustes no *kernel size*, no *padding* e no *stride*. Toda camada de uma Rede Neural deve ser dimensionada de acordo com seu *input*, no caso da primeira camada esse é o *input* da rede inteira e no caso das camadas subsequentes, o *input* é a saída da camada anterior. Sendo assim, mudando o *input* da primeira camada, foi necessário

mudar os paraâmetros citados acima para todas as camadas subsequentes. Ainda sim, foi decidido manter o mesmo número de *kernels* da arquitetura original: 96 para a primeira camada, 256 para a segunda, 384 para terceira e quarta e 256 para a quinta. Para os outros parâmetros da primeira camada foram realizados pequenos ajustes: *kernel size* foi alterado de 11 para 10, o *stride* foi mantido em 4 e o *padding* foi alterado para 1. Isso nos leva a um output de 63 *pixels* para a próxima entrada, calculado a partir da Equação 16.

$$w_{i+1} = \frac{w_i - f + 2 * p}{s} + 1 \tag{16}$$

Em que w é o tamanho do *input* da camada i, começando em 1. f é o tamanho do *kernel*, p o tamanho do *padding* e s o tamanho do *stride*. Também buscamos sempre ficar com os números mais próximos possíveis da arquitetura original, lembrando que a equação acima deve sempre resultar em números inteiros (Pilla Jr et al., 2016). Com isso, chegamos à arquitetura do nosso modelo. A implementação completa dessa arquitetura se encontra no Apêndice B.

Para implementação utilizou-se o *framework Caffe*, um *framework* de aprendizagem computacional, focado principalmente em *Deep Learning*, desenvolvido pelo *Berkeley Vision and Learning Center* e aprimorado pela comunidade que surgiu em torno do mesmo depois de sua publicação no repositório aberto, no site GitHub. Foi criado durante o PhD de Yangqing Jia em Berkeley (JIA et al., 2014).

O *framework Caffe* é modular, podendo ser utilizada nas linguagens C/C++, *Python* e até mesmo interagir com o MATLAB. Além disso, sua natureza de projeto aberto permite uma atualização contínua e rápida, mantendo o código e modelos de redes neurais em alto nível. Outro atrativo para o *framework* é o fato de conter muitos exemplos relacionados diretamente com o reconhecimento de imagens utilizando redes neurais convolucionais. Além disso, a exigência computacional inicial é baixa; o *framework* pode ser utilizado em modo CPU ou em GPU, o que permite uma facilidade nos testes e também uma alta velocidade quando associado a uma GPU (JIA et al., 2014). No caso deste projeto, foi utilizado o o modo GPU.

Iniciando o projeto e tendo escolhido a Rede Neural Convolucional AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) - em uma versão adaptada para o nosso problema e nosso banco de dados - o fato desta rede já ter sido implementada e desenvolvida no Caffe nos motivou a utilizá-lo, garantindo que a Rede Neural Convolucional escolhida teria uma boa resposta dentro deste *framework*. A facilidade de implementação

de uma Rede Neural Convolucional dentro do Caffe, sendo necessário somente uma declaração em um tipo de arquivo *protobuf* (Protocol Buffers - Protocolo de Retentores), que é lido pelo *framework*, foi outro motivador, visto que é possível modificar a modelagem de maneira muito mais ágil. O formato de arquivo *protobuf* foi criado com o conceito de ser neutro em relação à linguagem e à plataforma, com o objetivo de isolar os dados da aplicação e tornar sua leitura rápida. Um exemplo básico retirado da página de documentação pode ser visto a seguir.

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
    enum PhoneType {
        MOBILE = 0;
        HOME = 1;
        WORK = 2;
    }
    message PhoneNumber {
        required string number = 1;
        optional PhoneType type = 2 [default = HOME];
    }
    repeated PhoneNumber phone = 4;
}
```

O formato é similar a XML e possui algumas vantagens sobre este, como: Maior simplicidade, 3 a 10 vezes menor e de 20 a 100 vezes mais rápido, entre outras (GOOGLE, 1999). O formato nos permitiu fazer mudanças rápidas na modelagem da Rede, tornando o processo mais rápido, já que muitas vezes é necessário fazer um ajuste fino na Rede antes de finalizar sua modelagem. Segue um exemplo de uma Rede Neural Convolucional escrita em *.proto*:

```
layer {
  name: "mnist"
  type: "Data"
  transform_param {
    scale: 0.00390625
  }
  data_param {
    source: "mnist_train_lmdb"
    backend: LMDB
    batch_size: 64
  }
  top: "data"
  top: "label"
}
layer {
  name: "conv1"
  type: "Convolution"
  param { lr_mult: 1 }
  param { lr_mult: 2 }
  convolution_param {
```

```
    num_output: 20
    kernel_size: 5
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
  bottom: "data"
  top: "conv1"
}
layer {
  name: "pool1"
  type: "Pooling"
  pooling_param {
    kernel_size: 2
    stride: 2
    pool: MAX
  }
  bottom: "conv1"
  top: "pool1"
}
layer {
  name: "ip1"
  type: "InnerProduct"
  param { lr_mult: 1 }
  param { lr_mult: 2 }
  inner_product_param {
    num_output: 500
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
  bottom: "pool2"
  top: "ip1"
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "ip1"
  top: "ip1"
}
layer {
  name: "ip2"
  type: "InnerProduct"
  param { lr_mult: 1 }
  param { lr_mult: 2 }
  inner_product_param {
    num_output: 10
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
  bottom: "ip1"
  top: "ip2"
}
```

```
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2"
  bottom: "label"
}
```

Para a aplicação de algoritmos de aprendizado de máquina, é necessário separar o banco de dados em pelo menos duas partes, muitas vezes sendo separado em três: treino, teste e validação. Isso se dá para evitar o *overfitting*, visto que não é possível avaliar o desempenho de uma rede em dados na qual ela já foi treinada. Sendo assim, foi feita tal separação em 70% para treinamento, 20% para teste e 10% para validação (GOODFELLOW; BENGIO; COURVILLE, 2016a).

Durante o treinamento, com a porção do banco de dados previamente separada para treino, é realizada uma etapa de teste, utilizando porção do banco de dados previamente separada para teste. Por exemplo, a cada 5 mil ciclos de treinamento são feitos N ciclos de teste, onde é possível avaliar como a rede está progredindo com o seu treinamento. A validação fica separada totalmente da etapa de treinamento, para se ter dados que nunca tiveram contato com a Rede na avaliação da mesma. Os dados separados para validação são utilizados somente após todo o treino da Rede, simulando uma operação com dados do mundo real.

No caso específico de imagens de expressões faciais é necessária uma preocupação adicional com a repetição da mesma pessoa no teste, treino e validação. Como cada indivíduo possui mais de uma imagem para a mesma expressão, a rede poderia facilmente replicar a identificação do mesmo, se comportando de maneira muito boa durante o processo de treinamento, teste e validação e tendo um resultado ruim quando testado em amostras que se diferenciassem desta fisionomia (indivíduo), tornando o processo inteiro irrelevante. Assim, houve uma preocupação de separar as pessoas para que elas não se repetissem entre a separação do banco em treinamento, teste e validação.

Durante o processo de treinamento também é necessário embaralhar as imagens do banco. Se as imagens forem em sequência de uma mesma expressão de uma mesma pessoa, a Rede não irá se adaptar para a expressão, mas sim para a pessoa. Na troca de pessoa, este processo recomeçaria e nunca teríamos uma convergência para a generalização que desejamos. Isso foi percebido no nosso primeiro treinamento, onde a acurácia de Rede começava em 15%, subia até em torno de 80% e depois caia para 15% novamente. Esse embaralhamento de imagens permite que o banco se aproxime mais de uma distribuição natural, uma distribuição que permite que a Rede tenha uma generalização mais efetiva

e não sofra com *overfitting*.

Para tornar o processo de melhoria mais rápido e visto que a equipe não tinha prática com o treino de Redes Neurais Convolucionais, decidiu-se que seria interessante o treino inicial ser feito com apenas três categorias, de maneira que o refinamento da Rede pudesse ser feito de maneira mais rápida e para melhor conhecer todo o processo. Para efeito de comparação, com três emoções o treinamento durou cerca de 4 a 5 horas, sendo que o treinamento posterior, com as 7 emoções durou em torno de 22 horas. O modo utilizado em ambos os treinamentos foi o modo de GPU, mais rápido que o modo CPU, utilizando-se de uma placa de vídeo GeForce 750 TI.

As três expressões escolhidas foram expressões que normalmente não possuem sobreposição na identificação, tornando o processo mais preciso. Com o *dataset* reduzido e com menos categorias, tive-se uma maior facilidade de refinar a arquitetura e os hiper parâmetros da rede. Após um resultado encorajador, decidiu-se prosseguir com o treinamento com todas as expressões. A Tabela 2 apresenta a matriz de confusão para a Rede com somente três emoções.

Tabela 2: Matriz de Confusão da Rede de 3 emoções

		Classificada		
		Aversão	Felicidade	Surpresa
Verdadeira	Aversão	3112	0	1
	Felicidade	143	7527	0
	Surpresa	87	0	4556

Os hiperparâmetros e a arquitetura da Rede foram modificados para a versão com as 7 emoções somente para incluir a possibilidade de mais 4 saídas, visto o sucesso apresentado na Rede inicial. Considerando isso, foram realizados os mesmos processos anteriores de aumento do Banco de Dados nas outras 4 emoções e também foi sentida a necessidade de uma maior análise dos resultados obtidos, visto que nem sempre a acurácia em um processo de treinamento é representativa de como o algoritmo irá se comportar quando apresentado a dados fora do seu universo de treinamento. Por esta razão, foi decidido utilizar o método de validação cruzada, mais específico o tipo *k-fold* (DUAN; KEERTHI; POO, 2003).

O método de validação cruzada de *k-fold* consiste em separar o Banco de Dados em k partes, utilizando uma das separações para teste e $k-1$ para treinamento. Esse processo é repetido de maneira a utilizar todas as separações k como teste. No final

desse processo de treinamento, os resultados são analisados de maneira a perceber se os resultados entre os *k-fold* são muito diferentes e tentar perceber o que pode ter causado esse espalhamento de acurácias. Se é perceptível que as acurácias se mantém próximas em todos ou a maioria dos *k-fold* de treinamento, se tem uma maior possibilidade de que o algoritmo de treinamento seja validado em outros dados fora do universo de treinamento e teste. No nosso caso, foi decidido por utilizar um k igual a 10, ainda nos atentando, como anteriormente, para a não repetição de uma mesma pessoa nos dados de treinamento, teste e validação. Com essa repetição de treinamentos tem-se uma melhor chance de evitar resultados enganosos, diminuindo o número de acertos aleatórios.

6 RESULTADOS

Nesta etapa, estão disponíveis 10 redes treinadas. Cada uma dessas redes possui a mesma estrutura, porém foram treinadas com diferentes imagens utilizando o método de validação cruzada de *10-fold* (*k-fold* utilizando 10 divisões). Isso afeta os filtros e pesos definidos durante o treinamento, o que, por fim, altera a capacidade de classificação da rede.

6.1 ACURÁCIA

Para avaliarmos cada uma das 10 redes treinadas, foram classificadas as imagens de validação em cada uma das respectivas redes e obtidas as respectivas Matrizes de Confusão. A Matriz de Confusão nos permite saber a acurácia da rede neural, quantas vezes a expressão foi classificada corretamente e em quantos casos as expressões são classificadas como uma outra expressão. No Apêndice A estão as matrizes de confusão de cada uma das redes neurais convolucionais treinadas.

Uma das maneiras de interpretar essas tabelas é calculando a acurácia de cada rede neural, ou seja, a porcentagem dos expressões classificadas corretamente para cada uma das CNNs. A Tabela 3 abaixo mostra a acurácia obtida por todas as 10 redes treinadas.

Tabela 3: Acurácia das Redes Treinadas de acordo com a validação

	Acurácia
Rede 1	94,91%
Rede 2	**63,58%**
Rede 3	99,53%
Rede 4	92,23%
Rede 5	**93,24%**
Rede 6	97,19%
Rede 7	94,44%
Rede 8	94,84%
Rede 9	93,93%
Rede 10	98,53%
Média	92,24%
Desvio Padrão	9,80%

Em comparação com os resultados obtidos em (BARTLETT et al., 1996; GU et al., 2012; KENJI, 1991; MISHRA et al., 2015; PADGETT; COTTRELL, 1995; YACOOB; DAVIS, 1996), a acurácia obtida pelas redes é acima do esperado que é de pelo menos 80%. A rede número 2 é uma exceção, pois obteve acurácia de 63,58%.

Porém, lembrando que o banco de dados e as partições criadas pelo *k-fold* são desbalanceadas e não possuem imagens com a mesma expressão igualmente distribuídas. Por esse motivo é preciso analisar os dados das matrizes de confusão com mais rigidez. Para isso, foram calculados os valores de Verdadeiros Positivos (VP), Falsos Positivos (FP), Verdadeiros Negativos (VN) e Falsos Negativos (FN) para cada expressão facial em cada uma das redes neurais. Com esses valores, é possível calcular a Sensibilidade (*Recall*), Especificidade, Valor Preditivo Positivo (*Precision*) e o Valor Preditivo Negativo do sistema para cada expressão (FAWCETT, 2006).

A Sensibilidade (S) é a probabilidade do sistema classificar como pertencendo a uma classe quando realmente é aquela classe (FAWCETT, 2006). A Equação 17 mostra como esse valor é calculado.

$$S = \frac{VP}{VP + FN} \tag{17}$$

A Especificidade (E) indica a probabilidade do sistema classificar como não per-

tencer a uma classe quando realmente não pertence (FAWCETT, 2006). A Equação 18 mostra como esse valor é calculado.

$$E = \frac{VN}{VN + FP} \tag{18}$$

O Valor Preditivo Positivo (VPP) é a proporção de classificações positivas corretas em relação a todas as classificações positivas (FAWCETT, 2006). A Equação 19 mostra como esse valor é calculado.

$$VPP = \frac{VP}{VP + FP} \tag{19}$$

O Valor Preditivo Negativo (VPN) é a proporção de classificações negativas corretas em relação a todas as classificações negativas (FAWCETT, 2006). A Equação 20 mostra como esse valor é calculado.

$$VPN = \frac{VN}{VN + FN} \tag{20}$$

Com esses 4 valores é possível ter uma melhor compreensão sobre a eficiência de classificação do sistema para cada uma das expressões faciais. No entanto, o sistema de Rede Profunda possuí 7 classes para classificação, o que faz com que o valor de VN seja naturalmente alto. Por esse motivo, só serão considerados a Sensibilidade e o Valor Preditivo Positivo. Idealmente, ambos esses valores devem ser os maiores possível. A Tabela 4 abaixo mostra os valores de S e VPP de cada rede neural para cada expressão facial.

Tabela 4: Tabela de Sensibilidade (S) e Valor Preditivo Positivo (VPP) de todas as redes treinadas para cada expressão facial

	Rede 1		Rede 2		**Rede 3**		Rede 4		**Rede 5**	
Expressão	S	VPP	S	VPP	S	VPP	S	VPP	S	VPP
Neutro	76,99%	37,48%	99,72%	9,59%	**100,00%**	**98,62%**	99,85%	94,67%	91,57%	80,58%
Raiva	81,49%	85,61%	65,79%	69,05%	**98,67%**	**100,00%**	89,67%	99,91%	97,82%	89,74%
Aversão	98,95%	100,00%	89,57%	100,00%	**100,00%**	**100,00%**	100,00%	78,09%	100,00%	100,00%
Medo	97.73%	100,00%	23,47%	100,00%	**100,00%**	**98,21%**	99,68%	85,63%	14,39%	100,00%
Felicidade	100,00%	98,28%	80,81%	95,05%	**100,00%**	**100,00%**	96,52%	97,55%	100,00%	100,00%
Tristeza	88,40%	100,00%	75,00%	65,91%	**97,18%**	**100,00%**	70,74%	53,66%	**50,23%**	**29,95%**
Surpresa	98,31%	99,92%	94,12%	99,72%	**100,00%**	**100,00%**	75,94%	100,00%	97,99%	100,00%
	Rede 6		Rede 7		Rede 8		Rede 9		**Rede 10**	
	S	VPP	S	VPP	S	VPP	S	VPP	S	VPP
Neutro	**97,71%**	**96,61%**	97,06%	92,17%	98,09%	95,30%	95,03%	65,53%	**100%**	**96,06%**
Raiva	**80,48%**	**99,87%**	93,43%	81,69%	91,43%	99,86%	84,49%	97,16%	**89,71%**	**91,13%**
Aversão	**100,00%**	**100,00%**	100%	98,25%	91,82%	99,41%	99,68%	89,21%	**100%**	**99,86%**
Medo	**84,09%**	**94,87%**	82,7%3	99,36%	100%	72,85%	100%	100,00%	**98,26%**	**100,00%**
Felicidade	**100%**	**97,93%**	100%	99,70%	100%	95,06%	95,86%	100,00%	**99,90%**	**100,00%**
Tristeza	**100%**	**97,24%**	79,89%	93,36%	97,57%	87,74%	77,57%	99,77%	**95,71%**	**99,78%**
Surpresa	**100%**	**93,08%**	100%	99,68%	87,44%	99,91%	100%	97,93%	**98,94%**	**100,00%**

Com essa tabela, é possível notar que algumas redes neurais, apesar de terem obtido um bom valor de acurácia, têm dificuldade de classificar algumas expressões. Quando a sensibilidade é muito baixa, indica que a rede tende a classificar uma expressão facial específica como outra uma das outras expressões. Por exemplo, a rede neural número 5 obteve acurácia de 93,24% (como visto na Tabela 3), mas apenas 50,23% das expressões de tristeza foram corretamente classificadas como tristeza. Na caso da sensibilidade, quando o valor é muito baixo, indica que a rede neural tende a classificar outras expressões faciais como a expressão específica. No mesmo caso da rede 5, apenas 29,95% das expressões classificadas como tristeza realmente pertenciam a esta classe. Isso indica que essa rede neural não é confiável para classificar essa expressão facial. Se a acurácia indica que a rede neural não é efetiva, no caso da rede 2 com 63,58% de acurácia, S e VPP também apresentarão problemas.

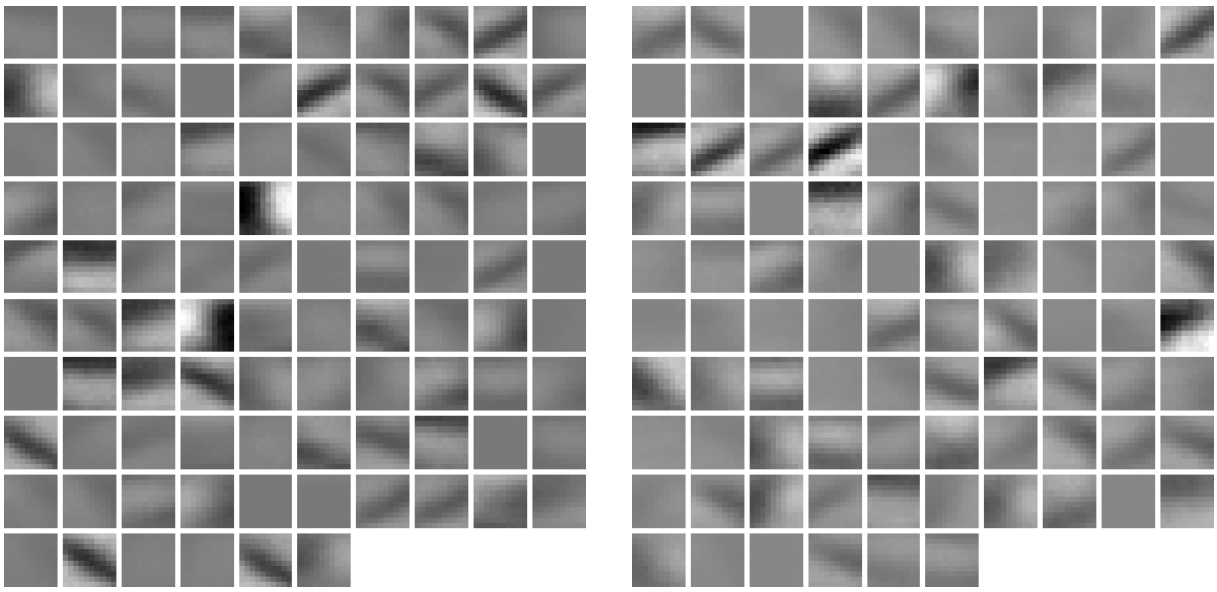
Como parâmetros de escolha das melhores redes, definidos arbitrariamente, espera-se que todos os valore de S e VPP estejam maiores que 80%. As únicas redes que se enquadram nesse parâmetro são as redes 3, 6 e 10. Apenas 3 redes neurais, diferente das 9 redes neurais indicadas pela tabela de acurácia.

6.2 RESULTADOS INTERMEDIÁRIOS DO PROCESSO DE CLASSIFICAÇÃO

Apesar de extremamente complexo, é interessante explorar o que realmente acontece dentro da rede neural durante o processo de classificação. Para que o processo de classificação possa ser observado, serão ilustrados os passos tomados pela CNN e seus resultados intermediários. A Figura 20 serviu como entrada na rede neural para que sua classificação pudesse ter sido observada em seus diversos estágios.

Figura 20: Imagem de entrada da CNN, representando a expressão facial Neutro
Fonte: Alterado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

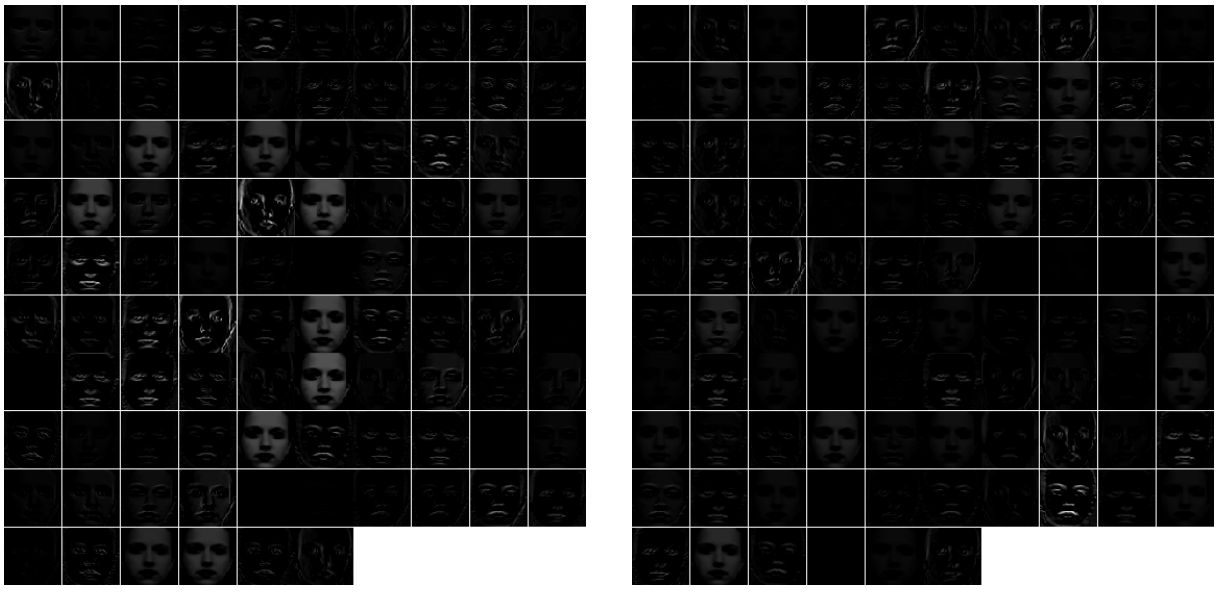
Os filtros de imagem da CNN estão nas 5 primeiras camadas (conv1, conv2, conv3, conv4 e conv5), cada camada com diferentes números de filtros e tamanhos, como visto anteriormente na construção da arquitetura. No total, a arquitetura gerou 368.748 filtros. A Figura 21 apresenta os filtros da primeira camada (conv1) das redes 2 e 3 de tamanho 10x10 *pixels*.



(a) (b)

Figura 21: Filtros da primeira camada da rede neural 2 e 3 (respectivamente)

Durante o processo de classificar a imagem, é feita a convolução da imagem de entrada com os filtros da primeira camada. A Figura 22 ilustra alguns dos resultados da convolução de uma das imagens existentes no banco de dados na rede 2 e rede 3.

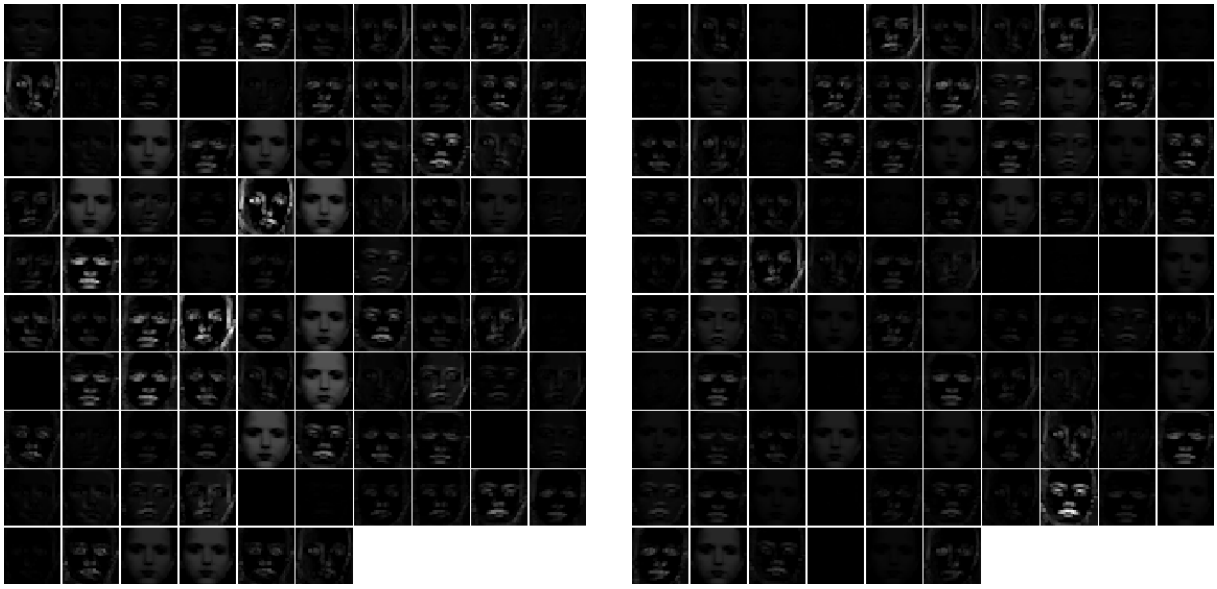


(a) (b)

Figura 22: Imagens resultantes da convolução pela primeira camada da rede neural 2 e 3 (respectivamente)

Logo após a convolução, é realizado o processo de *pooling* na imagem, resultando em uma imagem menor. A Figura 23 ilustra o resultado obtido na figura anterior após

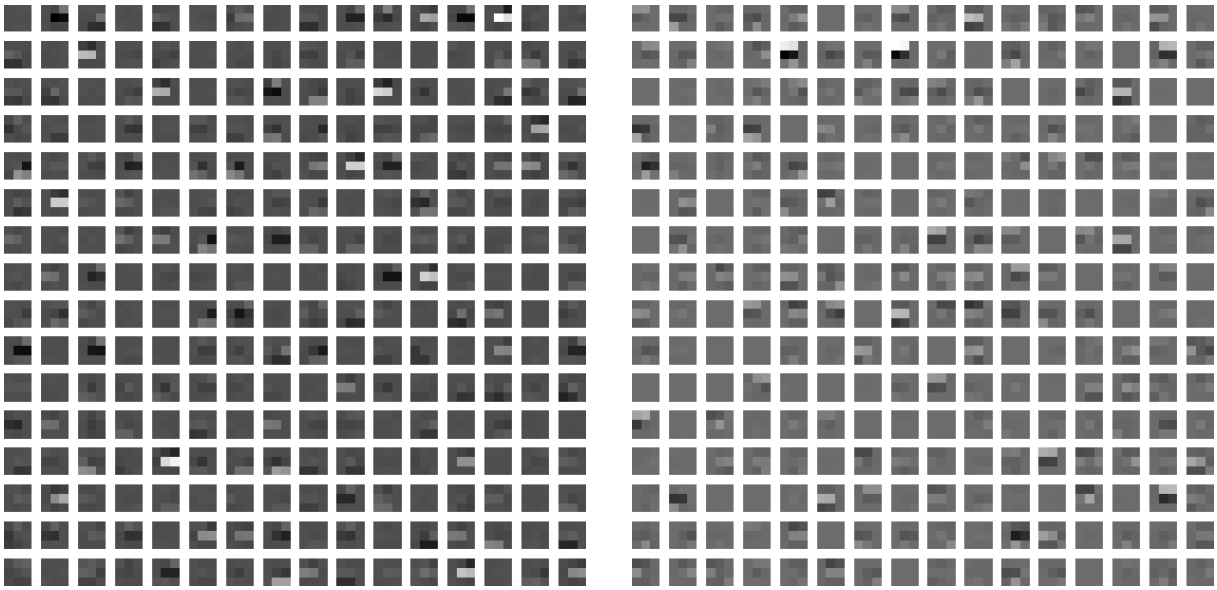
ser feito o processo de *pooling*, que servirão de entrada para a próxima camada da rede neural.



(a) (b)

Figura 23: Imagens resultantes do *pooling* na primeira camada da rede neural 2 e 3 (respectivamente)

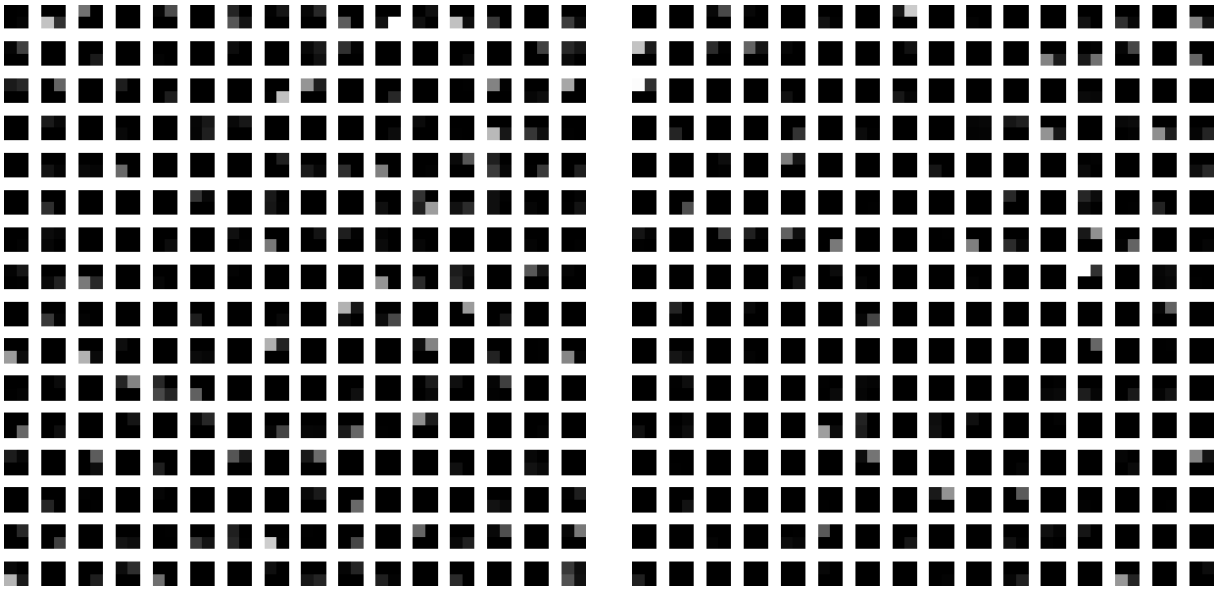
Esse processo de convolução e *pooling* se repete até o fim da última camada convolucional da CNN, antes de entrar nas camadas interconectadas. A Figura 24 mostra apenas alguns dos 98.304 filtros da última camada convolucional (conv5) das redes 2 e 3 de tamanho 3x3 *pixels*.



(a) (b)

Figura 24: Filtros da última camada convolucional da rede neural 2 e 3 (respectivamente)

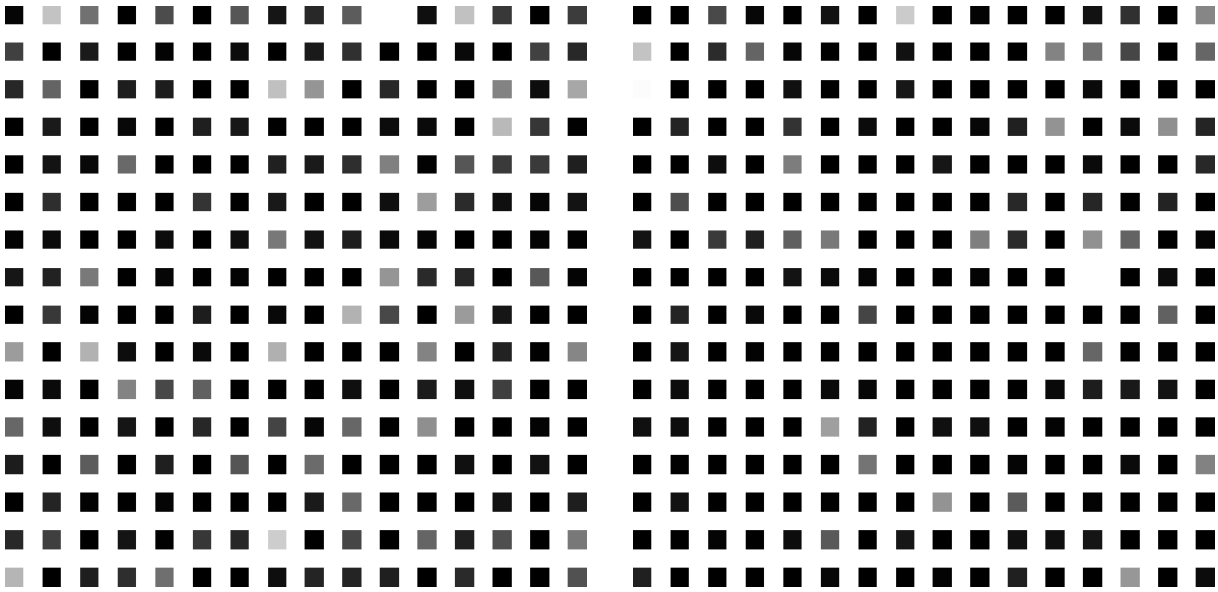
As imagens resultantes da convolução feita pela primeira camada conseguiam ser interpretadas até certa extensão. Por exemplo, a maioria dos filtros da primeira camada são filtros de borda, como pode ser visto na Figura 21 e comprovado na Figura 22. Por causa da complexidade da CNN, não é possível saber exatamente qual a funcionalidade e importância delas no funcionamento da rede como um todo. Nessa última camada da CNN, o resultado de cada convolução é de apenas quatro *pixels* (dimensão de 2x2 *pixels*), o que torna impossível compreendermos seus significados. A Figura 25 ilustra alguns dos resultados da convolução na última camada convolucional.



(a) (b)

Figura 25: Imagens resultantes da convolução pela última camada convolucional da rede neural 2 e 3 (respectivamente)

Após o *pooling*, cada imagem resultante nessa última camada são resumidas à apenas um *pixel*. Cada um desses *pixels* individuais serão usadas como entrada para as camadas interconectadas e sua configuração classificada como uma das sete expressões faciais. A Figura 26 ilustra o resultado do processo de *pooling* obtido na quinta camada convolucional, que servirão de entrada para as camadas interconectadas.



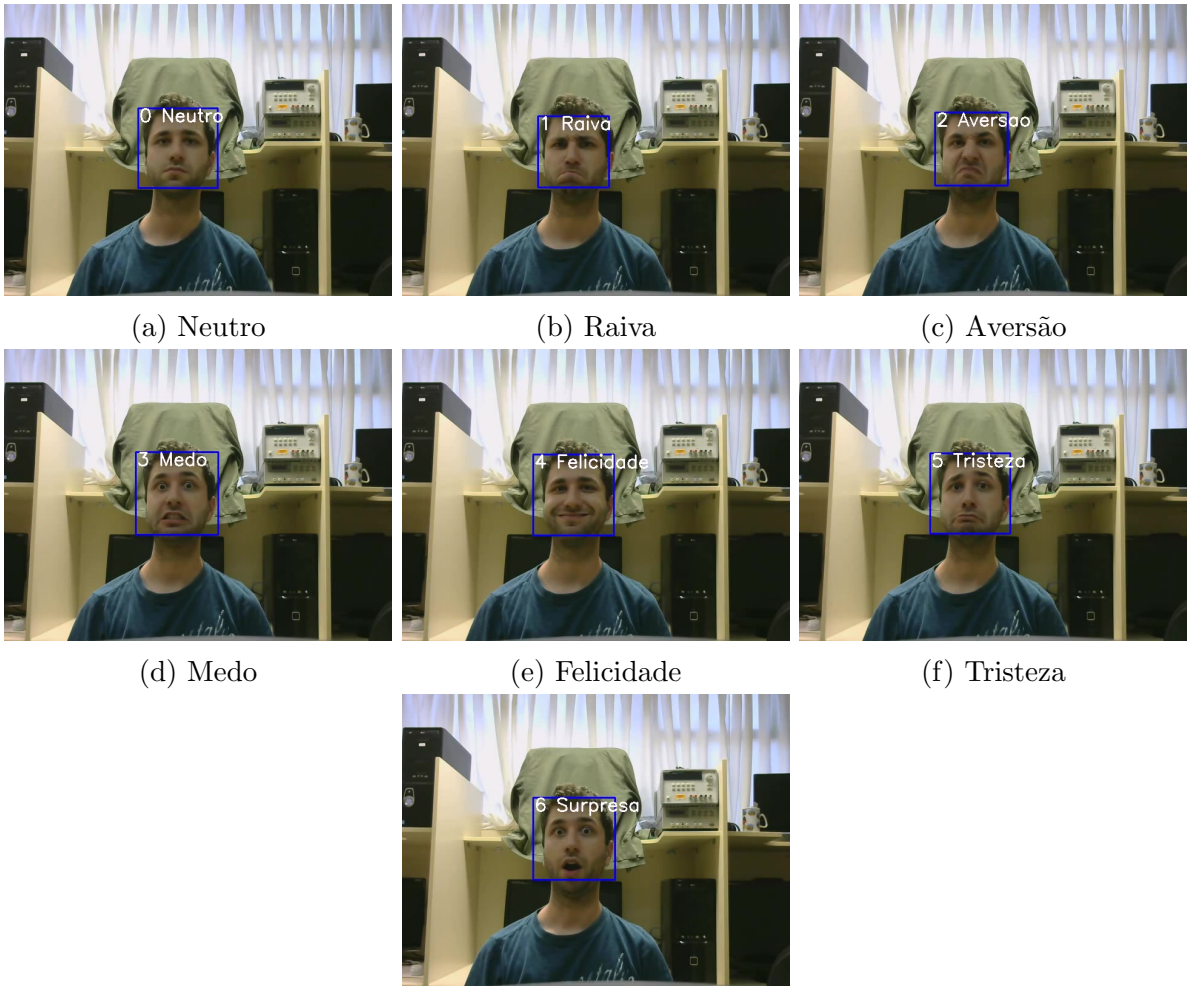
(a) (b)

Figura 26: Imagens resultantes do *pooling* na última camada convolucional da rede neural 2 e 3 (respectivamente)

Os cálculos feitos nas camadas interconectadas da CNN utiliza como entrada os valores da última camada convolucional. Esses cálculos resultam na classificação desses pontos, que são características extraídas da imagem de entrada da CNN. No caso do exemplo dado nessa seção, a classificação realizada em ambas as CNNs foram bem sucedidos, resultando em Neutro.

6.3 TESTE DA CNN EM TEMPO REAL

As redes neurais também foram submetidas a um teste em tempo real. Utilizando uma câmera ligada ao computador, são realizados os mesmos processos utilizados na normalização do banco de dados (detecção facial, recorte quadrado, redimensionamento para 256x256 *pixels* e conversão para níveis de cinza). Por fim, essa imagem do rosto normalizado é classificado pela Rede Neural Profunda. Essa classificação é feita em cada *frame* e é mostrada na tela junto ao rosto. Quando apenas um rosto está sendo classificado por vez, o sistema conseguiu processar a até 25 *frames* por segundo no computador utilizado. Abaixo, na Figura 27, estão alguns *frames* processados e classificados corretamente pela rede neural número 3 em tempo real.



(a) Neutro (b) Raiva (c) Aversão

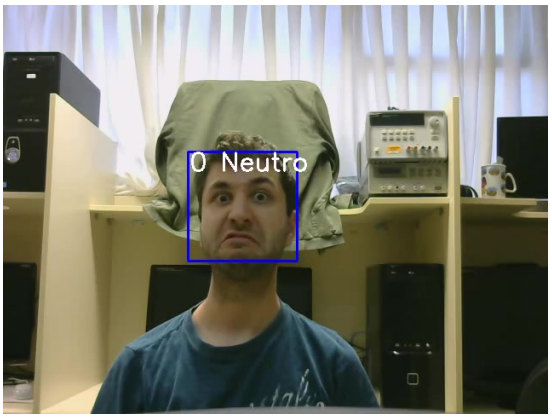
(d) Medo (e) Felicidade (f) Tristeza

(g) Surpresa

Figura 27: *Frames* classificados corretamente pena CNN em tempo real

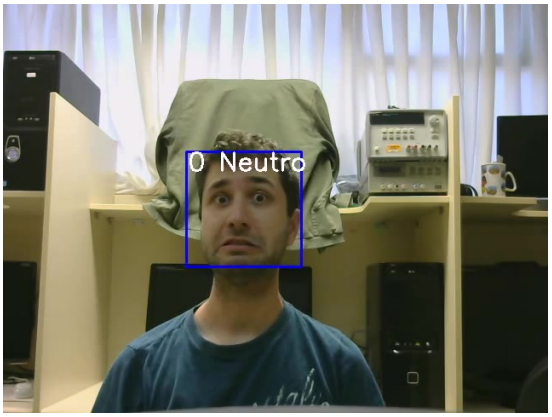
Para que a rede neural consiga classificar corretamente, as imagens de entrada devem estar no mesmo condicionamento que as imagens utilizadas durante o treinamento.

Por esse motivo, espera-se que o desempenho da rede neural profunda não seja o mesmo discutido na seção 6.1. Apesar do resultado desse teste realizado na rede neural seja satisfatório, houve momentos em que a rede neural não conseguiu classificar corretamente. A Figura 28 ilustra alguns desses casos que a rede neural número 3 não conseguiu reconhecer corretamente a expressão facial.



(a) Aversão classificada como neutro

(b) Aversão classificada como tristeza



(c) Medo classificada como neutro

(d) Tristeza classificada como aversão

Figura 28: *Frames* não reconhecidos corretamente pele classificação em tempo real

Para tentar obter um sistema com melhores resultados. Foram escolhidas as 5 melhores redes neurais e implementado um sistema de votação. Todas as redes fazem a classificação de cada *frame* e a expressão facial com mais classificações será a escolhida para classificar a expressão do rosto. Observou-se que os resultados foram melhores, porém houve um decréscimo no *frame rate*, atingindo até 13 *frames* por segundo.

7 CONCLUSÕES E TRABALHOS FUTUROS

Em pesquisas de desenvolvimento de sistemas que fazem a classificação expressões faciais, os valores de acurácia foram acima de 80%. Comparando com esse valor, as Redes Neurais Profundas desenvolvidas obtiveram resultados muito acima do esperado, com acurácia de 92,24% em média. Para melhor análise do comportamento dessas redes neurais, foram calculados os valores de Sensibilidade e Valor Preditivo Positivo para cada uma das expressões a serem classificadas. Três das dez redes obtiveram valores acima da margem de 80% para todas as expressões.

O processo de classificação em tempo real também apresentou resultados satisfatórios. Apesar de parecer não ter a mesma acurácia calculada, a rede aparenta conseguir classificar as expressões faciais com acurácias satisfatórias. Essa diferença pode ser atribuída ao fato do banco de dados ter sido criado em um ambiente muito controlado, diferenciando do ambiente testado pelo sistema desenvolvido neste trabalho.

Existem alguns pontos de partida que projetos futuros podem tomar a partir deste trabalho. Dentre eles sugerimos:

- Testar a rede com um banco de dados diferente e observar o desempenho obtido. Ao testar a rede com diferentes bancos de dados, é possível saber como ela se comporta com imagens em diferentes condições. Isso permite uma melhor validação do funcionamento das Redes Neurais Profundas desenvolvidas.

- Melhorar o desempenho obtido utilizando outros bancos de dados. Os filtros das camadas convolucionais e os pesos das camadas interconectadas podem ser reutilizados como estado inicial de uma rede antes do treinamento. Dessa maneira, é possível submeter a rede neural à novos treinamentos utilizando outro banco de dados. Isso permite que a rede neural se condicione a novas e diferentes informações, como, por exemplo, orientação da cabeça, iluminação e rotação.

- Utilizar a Rede Neural Profunda em aplicações reais. Como citado no inicio do trabalho, existem aplicações para um sistema capaz de classificar expressões faciais. Dentre as áreas de pesquisa podem citadas: Marketing, Interação Humano-Máquina, Entretenimento e Atendimento ao Cliente. Mas qualquer sistema que possa fazer uso das emoções de uma pessoa pode ser um sistema que possa ser desenvolvido. Utilizar este classificador como parte de um sistema maior trás grande validação ao projeto desenvolvido.

REFERÊNCIAS

BARTLETT, M. S.; VIOLA, P. A.; SEJNOWSKI, T. J.; GOLOMB, B. A.; LARSEN, J.; HAGER, J. C.; EKMAN, P. Classifying facial action. **Advances in neural information processing systems**, Morgan Kaufmann Publishers, p. 823–829, 1996.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006.

BONN, U. of. **Computer Science VI - Autonomous Intelligent Systems**. 2010. Disponível em: <http://www.ais.uni-bonn.de/deep_learning/>. Acessado em 23/05/17.

BUSSO, C.; DENG, Z.; YILDIRIM, S.; BULUT, M.; LEE, C. M.; KAZEMZADEH, A.; LEE, S.; NEUMANN, U.; NARAYANAN, S. S. Analysis of emotion recognition using facial expressions, speech and multimodal information. In: **Proceedings of the International Conference on Multimodal Interfaces**. State Park, PA: [s.n.], 2004. p. 205–211.

CALVO, R. A.; D'MELLO, S. Affect detection: An interdisciplinary review of models, methods, and their applications. **IEEE Transactions on affective computing**, IEEE, v. 1, n. 1, p. 18–37, 2010.

DARWIN, C. **The Expression of the Emotions in Man and Animals**. London: John Murray, 1899.

DUAN, K.; KEERTHI, S. S.; POO, A. N. Evaluation of simple performance measures for tuning svm hyperparameters. **Neurocomputing**, Elsevier, v. 51, p. 41–59, 2003.

DUCHENNE, G. B.; CUTHBERTSON, R. A. **The mechanism of human facial expression**. [S.l.]: Cambridge university press, 1990.

EGMONT-PETERSEN, M.; RIDDER, D. de; HANDELS, H. Image processing with neural networks—a review. **Pattern recognition**, Elsevier, v. 35, n. 10, p. 2279–2301, 2002.

FAWCETT, T. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Chapter 5 of deep learning. In: . [S.l.]: MIT Press, 2016. Http://www.deeplearningbook.org.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org>. Acessado em 19/02/17.

GOOGLE. **Google Protocol Buffers overview**. 1999. Disponível em: <https://developers.google.com/protocol-buffers/docs/overview>. Acessado em 21/02/17.

GU, W.; XIANG, C.; VENKATESH, Y.; HUANG, D.; LIN, H. Facial expression recognition using radial encoding of local gabor features and classifier synthesis. **Pattern Recognition**, Elsevier, v. 45, n. 1, p. 80–91, 2012.

HEINE, S. J.; LEHMAN, D. R.; MARKUS, H. R.; KITAYAMA, S. Is there a universal need for positive self-regard? **Psychological review**, American Psychological Association, v. 106, n. 4, p. 766–794, 1999.

JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. **arXiv preprint arXiv:1408.5093**, 2014.

KANADE, T.; COHN, J. F.; TIAN, Y. Comprehensive database for facial expression analysis. In: IEEE. **Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on**. [S.l.], 2000. p. 46–53.

KELTNER, D.; EKMAN, P.; GONZAGA, G. C.; BEER, J. Facial expression of emotion. Oxford University Press, 2003.

KENJI, M. Recognition of facial expression from optical flow. **IEICE Transaction on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, v. 74, n. 10, p. 3474–3483, 1991.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1097–1105.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time series. **The handbook of brain theory and neural networks**, v. 3361, n. 10, p. 1995, 1995.

LEE, H.; PARK, M.; KIM, J. Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning. In: **2016 IEEE International Conference on Image Processing (ICIP)**. [S.l.: s.n.], 2016. p. 3713–3717.

LUCEY, P.; COHN, J. F.; KANADE, T.; SARAGIH, J.; AMBADAR, Z.; MATTHEWS, I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In: IEEE. **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops**. [S.l.], 2010. p. 94–101.

LYONS, M.; AKAMATSU, S.; KAMACHI, M.; GYOBA, J. Coding facial expressions with gabor wavelets. In: IEEE. **Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on**. [S.l.], 1998. p. 200–205.

MARKUS, H. R.; KITAYAMA, S. Culture and the self: Implications for cognition, emotion, and motivation. **Psychological review**, American Psychological Association, v. 98, n. 2, p. 224–253, 1991.

MARSLAND, S. **Machine learning: an algorithmic perspective**. [S.l.]: CRC press, 2009.

MATHWORKS. Detect objects using the Viola-Jones algorithm - MATLAB. Novembro, 2016. Disponível em: <https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html>. Acessado em 19/02/17.

MATLAB. **version 7.10.0 (R2010a) (VERSÃO LICENSIADA PELA UTFPR)**. Natick, Massachusetts: The MathWorks Inc., 2010. Acessado em 17/02/17.

MISHRA, B.; FERNANDES, S. L.; ABHISHEK, K.; ALVA, A.; SHETTY, C.; AJILA, C. V.; SHETTY, D.; RAO, H.; SHETTY, P. Facial expression recognition using feature based techniques and model based techniques: A survey. In: IEEE. **Electronics and Communication Systems (ICECS), 2015 2nd International Conference on**. [S.l.], 2015. p. 589–594.

MITCHELL, T. **Machine Learning**. New York: McGraw-Hill Science/Engineering/Math, 1997.

PADGETT, C.; COTTRELL, G. Identifying emotion in static face images. **Proceedings of the 2nd Joint Symposium on Neural Computation**, v. 5, p. 91–101, 1995.

Pilla Jr, V.; ZANELLATO, A. L.; FERREIRA, C. B.; GAMBA, H. R.; BORBA, G. B.; MEDEIROS, H. Facial expression classification using convolutional neural network and support vector machine. In: **Proceedings do XII Workshop em Visão Computacional**. Campo Grande, Mato Grosso, Brazil: [s.n.], 2016.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, v. 323, p. 533–536, 1986.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. **International Journal of Computer Vision (IJCV)**, v. 115, n. 3, p. 211–252, 2015.

SAMAL, A.; IYENGAR, P. A. Automatic recognition and analysis of human faces and facial expressions: A survey. **Pattern recognition**, Elsevier, v. 25, n. 1, p. 65–77, 1992.

SEBASTIANI, F. Machine learning in automated text categorization. **ACM computing surveys (CSUR)**, ACM, v. 34, n. 1, p. 1–47, 2002.

STANFORD, U. **Convolutional Neural Networks for Visual Recognition**. 2016. Disponível em: <https://http://cs231n.github.io/convolutional-networks/>. Acessado em 19/02/17.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern Recognition**. [S.l.]: Elsevier Academic Press, 2003.

TURK, M.; PENTLAND, A. Eigenfaces for recognition. **Journal of cognitive neuroscience**, MIT Press, v. 3, n. 1, p. 71–86, 1991.

VIOLA, P.; JONES, M. J. Robust real-time face detection. **International journal of computer vision**, Springer, v. 57, n. 2, p. 137–154, 2004.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. **IEEE transactions on evolutionary computation**, IEEE, v. 1, n. 1, p. 67–82, 1997.

WU, Z.; PENG, M.; CHEN, T. Thermal face recognition using convolutional neural network. In: **2016 International Conference on Optoelectronics and Image Processing (ICOIP)**. [S.l.: s.n.], 2016. p. 6–9.

YACOOB, Y.; DAVIS, L. S. Recognizing human facial expressions from long image sequences using optical flow. **IEEE Transactions on pattern analysis and machine intelligence**, IEEE, v. 18, n. 6, p. 636–642, 1996.

ZHU, X.; GHAHRAMANI, Z.; LAFFERTY, J. et al. Semi-supervised learning using gaussian fields and harmonic functions. In: **ICML**. [S.l.: s.n.], 2003. v. 3, p. 912–919.

APÊNDICE A – MATRIZES DE CONFUSÃO DAS 10 CNNS TREINADAS

Tabela 5: Matriz de Confusão da Rede 1

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	271	2	0	0	79	0	0
	Raiva	334	1470	0	0	0	0	0
	Aversão	0	0	1698	0	18	0	0
	Medo	22	1	0	1075	0	0	2
	Felicidade	0	0	0	0	5544	0	0
	Tristeza	52	244	0	0	0	2256	0
	Surpresa	44	0	0	0	0	0	2552

Tabela 6: Matriz de Confusão da Rede 2

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	351	0	0	0	0	1	0
	Raiva	189	2258	0	0	0	985	0
	Aversão	74	4	670	0	0	0	0
	Medo	2816	122	0	919	0	55	4
	Felicidade	46	410	0	0	1920	0	0
	Tristeza	95	476	0	0	100	2013	0
	Surpresa	88	0	0	0	0	0	1408

Tabela 7: Matriz de Confusão da Rede 3

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	4576	0	0	0	0	0	0
	Raiva	40	3039	0	1	0	0	0
	Aversão	0	0	1452	0	0	0	0
	Medo	0	0	0	440	0	0	0
	Felicidade	0	0	0	0	2024	0	0
	Tristeza	24	0	0	7	0	1069	0
	Surpresa	0	0	0	0	0	0	2596

Tabela 8: Matriz de Confusão da Rede 4

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	4086	2	0	0	4	0	0
	Raiva	22	2249	0	103	78	56	0
	Aversão	0	0	1012	0	0	0	0
	Medo	2	0	0	614	0	0	0
	Felicidade	0	0	118	0	3270	0	0
	Tristeza	206	0	0	0	0	498	0
	Surpresa	0	0	166	0	0	374	1704

Tabela 9: Matriz de Confusão da Rede 5

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	1531	123	0	0	0	18	0
	Raiva	24	1076	0	0	0	0	0
	Aversão	0	0	2332	0	0	0	0
	Medo	66	0	0	95	0	499	0
	Felicidade	0	0	0	0	5720	0	0
	Tristeza	219	0	0	0	0	221	0
	Surpresa	60	0	0	0	0	0	2932

Tabela 10: Matriz de Confusão da Rede 6

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	4643	1	0	1	96	11	0
	Raiva	156	779	0	33	0	0	0
	Aversão	0	0	1848	0	0	0	0
	Medo	7	0	0	629	0	4	108
	Felicidade	0	0	0	0	4532	0	0
	Tristeza	0	0	0	0	0	528	0
	Surpresa	0	0	0	0	0	0	1452

Tabela 11: Matriz de Confusão da Rede 7

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	4484	98	18	0	0	20	0
	Raiva	86	1521	0	0	8	13	0
	Aversão	0	0	1012	0	0	0	0
	Medo	107	31	0	1092	0	82	8
	Felicidade	0	0	0	0	2684	0	0
	Tristeza	188	212	0	7	0	1617	0
	Surpresa	0	0	0	0	0	0	2508

Tabela 12: Matriz de Confusão da Rede 8

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	2978	2	0	34	0	20	2
	Raiva	118	1408	12	0	0	2	0
	Aversão	0	0	2020	0	144	36	0
	Medo	0	0	0	660	0	0	0
	Felicidade	0	0	0	0	2772	0	0
	Tristeza	29	0	0	2	0	1245	0
	Surpresa	0	0	0	210	0	116	2270

Tabela 13: Matriz de Confusão da Rede 9

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	1129	14	0	0	0	3	42
	Raiva	232	1264	0	0	0	0	0
	Aversão	6	0	1886	0	0	0	0
	Medo	0	0	0	484	0	0	0
	Felicidade	4	0	228	0	5610	0	10
	Tristeza	352	23	0	0	0	1297	0
	Surpresa	0	0	0	0	0	0	2464

Tabela 14: Matriz de Confusão da Rede 10

		Classificada						
		Neutro	Raiva	Aversão	Medo	Felicidade	Tristeza	Surpresa
Verdadeira	Neutro	3344	0	0	0	0	0	0
	Raiva	82	750	0	0	0	4	0
	Aversão	0	0	1452	0	0	0	0
	Medo	4	19	0	1297	0	0	0
	Felicidade	2	0	2	0	3912	0	0
	Tristeza	29	54	0	0	0	1853	0
	Surpresa	20	0	0	0	0	0	1872

APÊNDICE B – ARQUITETURA E HIPERPARÂMETROS DA REDE NEURAL CONVOLUCIONAL

```
name: "ZanBortCaffe"
layer {
  name: "ZanBort"
  type: "Data"
  top: "data"
  top: "label"
  include{
phase: TRAIN
  }
  transform_param {
    scale: 0.00390625
    mean_file: "/media/andre/3982a960-1be5-41b6-adc0-4ef175fe4ef0/
    BancoImagens/Banco10/treino10mean.binaryproto"
  }
  data_param {
    source: "/media/andre/3982a960-1be5-41b6-adc0-4ef175fe4ef0/
    BancoImagens/Banco10/treino10"
    backend: LMDB
    batch_size: 16
  }
}
layer {
  name: "ZanBort"
  type: "Data"
  top: "data"
  top: "label"
  include{
phase: TEST
  }
  transform_param {
    scale: 0.00390625
    mean_file: "/media/andre/3982a960-1be5-41b6-adc0-4ef175fe4ef0/
    BancoImagens/Banco10/teste10mean.binaryproto"
  }
  data_param {
    source: "/media/andre/3982a960-1be5-41b6-adc0-4ef175fe4ef0/
    BancoImagens/Banco10/teste10"
    backend: LMDB
    batch_size: 16
  }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
  }
  param {
```

```
    lr_mult: 2
  }
  convolution_param {
    num_output: 96
    pad: 1
    kernel_size: 10
    stride: 4
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
    pad: 1
  }
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 256
    pad: 0
    kernel_size: 6
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
```

```
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
    pad: 0
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "pool2"
  top: "conv3"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 384
    pad: 0
    kernel_size: 3
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "pool3"
  type: "Pooling"
  bottom: "conv3"
  top: "pool3"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
    pad: 0
  }
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "pool3"
  top: "conv4"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
```

```
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu4"
  type: "ReLU"
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "pool4"
  type: "Pooling"
  bottom: "conv4"
  top: "pool4"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
    pad: 0
  }
}
layer {
  name: "conv5"
  type: "Convolution"
  bottom: "pool4"
  top: "conv5"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu5"
  type: "ReLU"
  bottom: "conv5"
  top: "conv5"
}
layer {
  name: "pool5"
  type: "Pooling"
  bottom: "conv5"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
```

```
    pad: 0
  }
}
layer {
  name: "ip9"
  type: "InnerProduct"
  bottom: "pool5"
  top: "ip9"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 7
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "drop6"
  type: "Dropout"
  bottom: "ip9"
  top: "ip9"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "accuracy"
  type: "Accuracy"
  bottom: "ip9"
  bottom: "label"
  top: "accuracy"
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip9"
  bottom: "label"
  top: "loss"
}
```

E os hiperparâmetros:

```
test_iter: 10000
test_interval: 2000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 100000
display: 50
max_iter: 200000
momentum: 0.9
weight_decay: 0.0005
snapshot: 50000
# solver mode: CPU or GPU
solver_mode: GPU
```