Universidade Federal
do Rio de Janeiro
Escola Politécnica

DIARIZAÇÃO E IDENTIFICAÇÃO DE LOCUTOR EM CONTEÚDO DE VÍDEO BASEADA EM ANÁLISE DE EXPRESSÃO FACIAL VIA APRENDIZADO DE MÁQUINA SUPERVISIONADO

Renan Fasolato Basilio

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Geraldo Zimbrão da Silva

Rio de Janeiro
Março de 2020

DIARIZAÇÃO E IDENTIFICAÇÃO DE LOCUTOR EM CONTEÚDO DE VÍDEO BASEADA EM ANÁLISE DE EXPRESSÃO FACIAL VIA APRENDIZADO DE MÁQUINA SUPERVISIONADO

Renan Fasolato Basilio

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO.

Examinado por:

_____
Prof. ,

_____
Prof. ,

_____
Prof. ,

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2020

Basilio, Renan Fasolato

Diarização e Identificação de Locutor em Conteúdo de Vídeo Baseada em Análise de Expressão Facial via Aprendizado de Máquina Supervisionado/Renan Fasolato Basilio. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2020.

VII, 29 p. 29, 7cm.

Orientador: Geraldo Zimbrão da Silva

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2020.

Referências Bibliográficas: p. 24 – 28.

1. Aprendizado Supervisionado. 2. Aprendizado de Máquina. 3. Diarização de Locutor. I. da Silva, Geraldo Zimbrão. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Computação e Informação. III. Título.

To-Do: Write Dedication

Sumário

Capítulo 1

Introdução

1.1 Descrição do Problema

A diarização de locutor consiste no processo de identificar os diferentes locutores em um conteúdo multimídia de forma a separa-los temporalmente, definindo quando quem falou, e produzir um tipo de roteiro para o mesmo.

Tradicionalmente, tenta-se resolver esse problema através da análise exclusiva do áudio, por meio da extração de *features* na forma de vetores denominados *I-vectors*, e subsequente clusterização destes. Porém, trata-se de um problema difícil; o timbre, principal característica sonora responsável pela identificação do locutor pelo ser humano, é de caráter neurológico [1], produzido pela decomposição da onda sonora em seus harmônicos pelo trato auditivo. E, ainda, como propriedade intrínseca da etapa de clusterização, a utilização desses algoritmos depende do conhecimento prévio do número de locutores que participam do áudio.

Dadas essas limitações, temos que o desempenho dos algoritmos considerados estado da arte é insuficiente, com taxa de erro de diarização (a partir daqui chamada de DER, do inglês *Diarization Error Rate*) de cerca de 20% [2] nas bases tradicionais. Portanto, a busca de outras técnicas capazes de prover um melhor desempenho nos leva a considerar também outros sinais constituintes do conteúdo multimídia, como o de vídeo do orador.

1.2 Motivação

Em muitas situações, no processo de transcrição de áudio e vídeo, é interessante obter também a informação de quem está falando. Com essa informação seria possível roteirizar a mídia, viabilizando um melhor entendimento e formatação do texto transcrito. Além disso, essa informação adicional permite viabilizar novos critérios de busca sobre o texto transcrito, tornando possível filtrar os resultados por locutor.

1.3 Definição do Problema

O problema de diarização de locutor consiste em particionar automaticamente um sinal de áudio tal que cada uma das partições geradas contenha as falas de um único locutor. Trata-se de um problema muito relevante à área de reconhecimento de voz, já que entender quem falou em cada momento de uma gravação nos permite contextualizar diversos tópicos que dependem desse tipo de informação para sua legibilidade e interpretação, tais como a geração automática de transcrições de depoimentos judiciais e relatórios médicos eletrônicos.

A solução proposta neste trabalho consiste em construir um sistema capaz de diarizar as falas de um único locutor em tempo real a partir de um vídeo frontal do mesmo. Para isso, o sistema de diarização deve ser capaz de, a partir de uma sequência de quadros extraída do vídeo original, determinar se o objeto da gravação está ou não falando. Neste trabalho iremos desconsiderar o áudio associado, já que o processamento deste foge à área de visão computacional.

Com esta finalidade, propomos uma arquitetura em três etapas:

1. Com o uso de um identificador facial, o sistema identificará rostos nas imagens que lhe forem fornecidas.

2. Pontos de referência relevantes serão extraídos dos rostos identificados.

3. Através de um modelo de aprendizado de máquina previamente treinado, o sistema deverá ser capaz de determinar se o sujeito identificado está ou não falando.

1.4 Escopo

De forma geral, gostaríamos de identificar as falas de todos os oradores do texto. No entanto, muitas vezes é suficiente identificar apenas um destes, por exemplo no caso de audiências e depoimentos do Tribunal de Justiça. Nesse caso específico desejamos identificar com maior precisão os trechos falados pelo depoente, independentemente do número de participantes da audiência, de forma a distinguir futuramente o que foi dito pelo mesmo em seu depoimento.

Para esta finalidade, assumimos que esteja disponível um vídeo frontal da face do locutor (depoente), além do áudio combinado de todos os participantes, em canal monoaural. Esta suposição é feita tendo em vista as características dos dados reais, tendo o conhecimento de que caso o áudio estivesse separado em canais correspondentes a cada locutor o problema se tornaria trivial. Assim, este trabalho ficará limitado a tratar de casos nos quais estas informações estejam disponíveis, e com a determinação de que o vídeo deve ser de boa qualidade.

1.5 Estrutura do Trabalho

Este trabalho se encontra organizado da seguinte maneira:

No capítulo 2 apresentamos a fundamentação teórica referente às técnicas e tecnologias utilizadas na implementação de cada uma das etapas propostas nesse trabalho. Em primeiro momento, fazemos uma revisão das abordagens e implementações formuladas para resolução do problema de alinhamento facial até hoje. Em seguida, apresentamos os conceitos e técnicas críticas para nossa implementação, sendo estas as redes neurais convolucionais, detecção de objetos por histograma de gradientes orientados, e alinhamento facial.

No capítulo 3 apresentamos a nossa implementação do sistema proposto. Abordamos tópicos como as ferramentas utilizadas, pré-processamento dos dados de treinamento, e a arquitetura proposta para a rede neural convolucional. Por fim, discutimos a implementação da aplicação final, para lidar com dados não tratados previamente.

No capítulo 4 demonstramos os resultados de nosso método quando aplicado sobre dataset fornecido pela Defensoria Pública do Estado do Rio de Janeiro, ~~além do dataset AMI Meeting Corpus [3],~~ que contém vídeos dos participantes individuais~~,~~ ~~em alta resolução e~~ com as características desejadas.

Por fim, no capítulo 5, resumimos e apresentamos nossas conclusões quanto ao trabalho realizado, discutindo os resultados obtidos e os fatores que contribuíram para estes, e enumeramos trabalhos futuros a serem realizados sobre o mesmo tema.

No apêndice, apresentamos o código fonte desenvolvido para este trabalho.

Capítulo 2

Revisão Bibliográfica

2.1 Trabalhos Correlatos

O problema de diarização de locutor é historicamente abordado de duas maneiras distintas. A abordagem tradicional envolve a extração de vetores de características do áudio que se deseja diarizar, podendo ser estes *I-Vectors* [4], extraídos por meio de transformações matriciais aplicadas sobre o sinal, ou *X-Vectors* [5], obtidos através do uso de redes neurais profundas. A partir de então, estes vetores podem ser clusterizados [6] quando informações referentes ao número de locutores é conhecido, ou, mais recentemente, utilizados como entrada de uma rede neural LSTM [7].

No entanto, a introdução das redes neurais convolucionais para reconhecimento de ações humanas em sinais de vídeos [8, 9] implicou na concepção de uma nova abordagem. Nesta, com o objetivo de melhorar a diarização de sinais heterogêneos, é utilizado um algoritmo de reconhecimento de ações também sobre o sinal de vídeo [10], que se supõe estar sincronizado ao áudio correspondente. O reconhecimento de ações é, nesse caso, utilizado para identificar a fala do locutor e, quando combinado com os sinais de áudio, é capaz de produzir resultados melhores do que o estado da arte no processamento exclusivo de áudio [11].

2.2 Fundamentação Teórica

Nesta seção discutiremos a fundamentação teórica das várias tecnologias utilizadas na execução do trabalho. Primeiramente, na seção 2.2.1, apresentaremos o conceito de redes neurais artificiais, utilizadas para construção do identificador de fala. Depois, na seção 2.2.2, discutiremos a técnica de *histograma de gradientes orientados*, utilizada pelo identificador de rostos frontais. Por fim, na seção 2.2.3, falaremos sobre a técnica de alinhamento facial, utilizada para a detecção dos marcadores faciais.

2.2.1 Redes Neurais Artificiais

Redes Neurais Artificiais são um conjunto de algoritmos inspirados pelo funcionamento do cérebro humano, introduzidos pela primeira vez em 1943 quando Warren McCulloch e Walter Pitts modelaram o funcionamento de neurônios através de circuitos elétricos simples[12]. Esse modelo continuou evoluindo ao longo dos anos, culminando no desenvolvimento do *Perceptron* por Rosenblatt em 1958 [13] (figura 2.3), um algoritmo de classificação binária semelhante a uma rede neural composta por um único neurônio.

No entanto, devido à falta de capacidade computacional e ao grande volume de dados necessários para o treinamento das redes neurais, estes algoritmos permaneceram apenas um conceito acadêmico durante vários anos. Somente em 2006, Geoffrey Hinton proporia formalmente o conceito de rede neural profunda [14], um tipo de rede neural composto por múltiplas camadas de neurônios densamente conectadas.

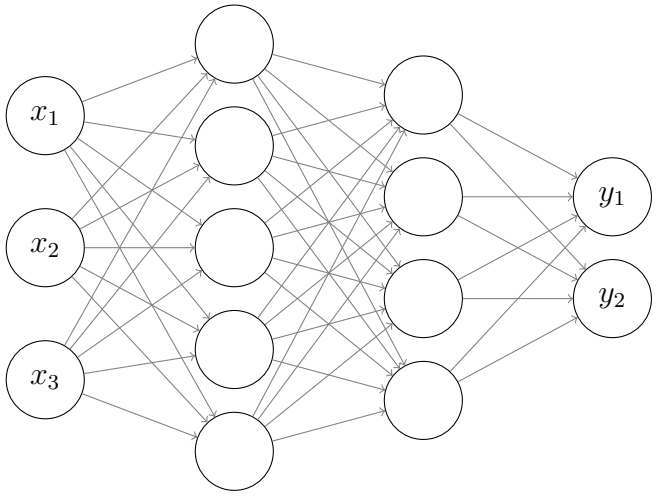

Figura 2.1: Ilustração de uma rede neural profunda com 3 camadas.

Recentemente, com o desenvolvimento de algoritmos para o treinamento de redes neurais em GPU (do Inglês *Graphics Processing Unit*) que possibilitaram o treinamento mais rápido e eficiente de redes neurais de múltiplas camadas, assim como o crescente volume de dados disponível para treinamento, o tópico de redes neurais vem rapidamente ganhando popularidade. Algoritmos desse tipo já são utilizados em diversas áreas da ciência, incluindo aplicações em medicina [15–17], previsão do tempo [18], veículos autônomos [19, 20], entre outras [21].

Tratam-se de algoritmos próprios para problemas que envolvem o reconhecimento de padrões, e a capacidade de treina-los a partir de dados existentes os torna eficazes para aplicação em tópicos nos quais as relações entre os dados de entrada não sejam totalmente conhecidas.

2.2.1.1 Neurônios Artificiais

Biologicamente, quando o potencial elétrico na base do axônio de um neurônio atinge um limiar pré-determinado através do acúmulo de sinais de entrada recebidos pelos dendritos, um pulso elétrico é gerado e propagado até as sinapses, terminais nos quais o neurônio se conecta com os demais. Trata-se de um sinal binário; o neurônio pode ou não estar ativado a cada instante. No entanto, a frequência relativa das ativações pode conter informação [22].

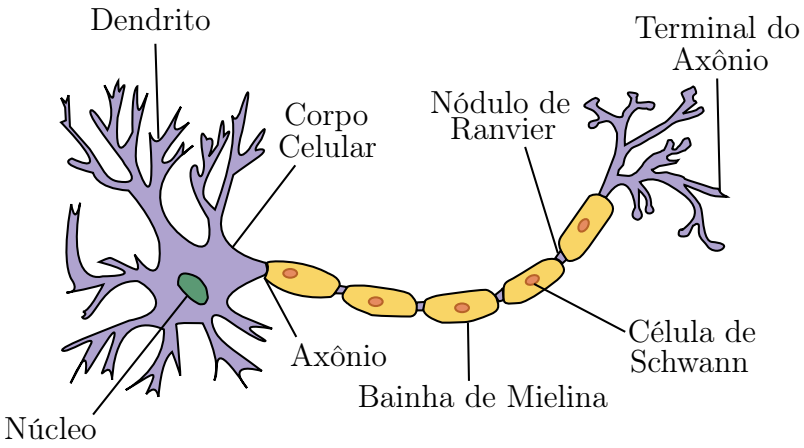

Figura 2.2: A estrutura de um neurônio biológico. Imagem adaptada de [23].

Funções de Ativação, também conhecidas como funções de propagação ou neurônios artificiais quando aplicadas no contexto de redes neurais, buscam modelar de forma simplificada esses aspectos biológicos. Elas recebem como entrada os valores resultantes das camadas anteriores, ponderados por pesos específicos a cada conexão, e, quando estes valores atingem um limiar de excitação pré-determinado, os transformam em uma única saída.

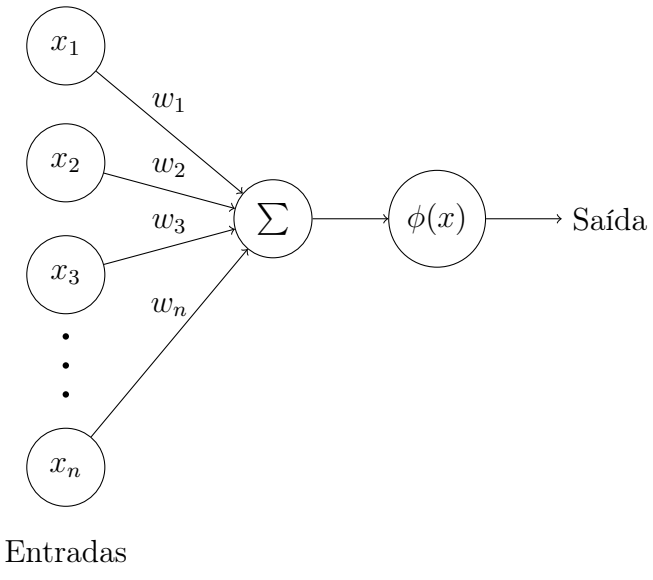

Figura 2.3: O *Perceptron*, uma rede neural com um único neurônio.

Geralmente, a função de ativação é constante e independente do processo de treinamento da rede neural; o ajuste é feito sobre os pesos das conexões, através do algoritmo de retropropagação [24]. O algoritmo é capaz de, a partir do resultado de uma camada, calcular o efeito de cada peso sobre o gradiente da função. Assim, ele possibilita a utilização de algoritmos tradicionais de otimização para encontrar os pesos locais de cada conexão em função do efeito global destes, mesmo em redes com múltiplas camadas de centenas de neurônios.

Comumente, são utilizadas como função de ativação as funções ReLU (*Rectified Linear Unit*), tangente hiperbólica, sigmoide, e Heaviside, em função do intervalo de saída e comportamento desejado.

$$\phi(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{2.1}$$

A função ReLU (equação 2.1), utilizada neste trabalho, tem comportamento tal que sua ativação ocorre sempre que $x > 0$, com o valor resultante sendo o próprio valor de x. Essa função é ideal para aplicações em redes neurais profundas pois o gradiente é preservado após cada camada da rede, o que não ocorre com as funções sigmoide e tangente hiperbólica. Nessas, o gradiente tende a valores infinitesimalmente pequenos após camadas sucessivas, dificultando o uso do algoritmo de retropropagação. Além disso, o treinamento de redes neurais profundas que utilizam a função ReLU é mais rápido do que o das outras funções de ativação, devido à sua simplicidade [25].

2.2.1.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais são redes neurais compostas por uma ou mais camadas de convolução. São ideais para o tratamento de dados matriciais, como por exemplo imagens, que constituem em uma matriz bi-dimensional de pixels, ou vídeos, matrizes tridimensionais compostas por múltiplas imagens dispostas segundo sua relação temporal. Essas redes são capazes de identificar relações entre os elementos vizinhos de uma matriz e, através da técnica de Max Pooling, focar somente nas regiões mais relevantes da entrada, reduzindo a complexidade do problema quando aplicadas a conjuntos de dados muito grandes.

Convolução e Filtros

A convolução é uma técnica matemática que consiste na aplicação de uma matriz, denominada matriz de convolução ou *filtro* quando utilizada no contexto de processamento de imagens, sobre cada elemento de uma outra matriz. Esse processo produz uma nova matriz na qual o valor de cada elemento corresponde ao seu valor

original combinado com o de seus vizinhos, e ponderados pelos valores da matriz de convolução (denominados *kernel* da convolução).

Para dados bi-dimensionais, a operação de convolução pode ser descrita da seguinte forma:

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \tag{2.2}$$

Essa técnica é frequentemente utilizada em áreas que trabalham com imagens, como na área de veículos autônomos, onde é capaz de identificar as faixas pintadas sobre o asfalto [20], assim como outros veículos e pedestres na rua. Através do uso dessa técnica, é possível remover características da imagem de entrada irrelevantes e enfatizar características mais relevantes para o problema que se deseja resolver, como mostra a figura 2.4.

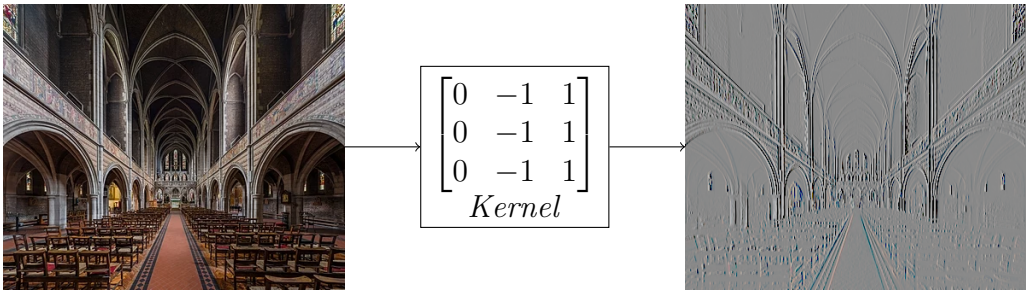

Figura 2.4: O resultado da aplicação de um *kernel* simples para detecção de bordas verticais a uma foto. O *kernel* é aplicado a cada pixel da imagem, substituindo o valor deste pela soma de sua vizinhança ponderada pelos valores da matriz. Foto original obtida de [26].

No contexto de redes neurais convolucionais, isso consiste em neurônios que realizam uma operação de convolução sobre os dados de entrada. O *kernel* da matriz de convolução é treinado de forma a identificar os aspectos da entrada mais relevantes para o problema, através do algoritmo de retropropagação.

Pooling

Um outro tipo de camada tradicional nas redes neurais convolucionais é a camada de *pooling*, ou "agrupamento". Esse tipo de camada tem como função reduzir o número e complexidade dos dados de entrada e promover uma melhor generalização espacial dos dados. Isso é necessários pois as camadas convolucionais tendem a manter a localização espacial das características detectadas na matriz de entrada, informação essa que na maioria dos casos não é relevante para o problema que se deseja resolver.

A operação consiste em subdividir a matriz de entrada em conjuntos de valores e, segundo um critério especificado, escolher entre estes um único valor resultante.

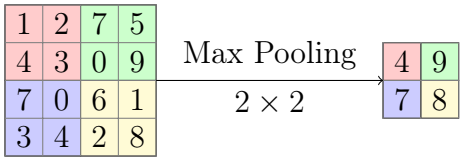

Figura 2.5: O resultado da operação de *Max Pooling* com dimensão 2×2 sobre uma matriz de entrada.

No contexto de redes neurais convolucionais, no entanto, tendo em mente que os valores resultantes da camada convolucional teriam sido filtrados por uma função de ativação não-linear, como a ReLU (equação 2.1), temos que os valores obtidos correspondem à intensidade da ativação dos neurônios. Assim, devido ao fato de que valores maiores devem corresponder a características de maior importância na matriz de entrada, temos dois critérios mais usados para a operação:

Max Pooling: Para cada grupo, calcula o valor máximo (figura 2.5).

Average Pooling: Para cada grupo, calcula a média dos valores.

A escolha entre esses critérios é feita em função do propósito da camada anterior; se esta for, por exemplo, um detector de bordas em imagens, temos que a técnica de *Max Pooling* será preferível, visto que as bordas serão sempre os elementos de maior valor do grupo, enquanto os demais terão valor igual a zero. Já no caso de se tratar de um reconhecedor de variação temporal, a técnica de *Average Pooling* pode ser preferível, já que será capaz de manter informações sobre o valor médio dos elementos no instante que está sendo comprimido.

2.2.2 Histograma de Gradientes Orientados

Histograma de Gradientes Orientados, ou HOG, do inglês *Histogram of Oriented Gradients*, é uma técnica de visão computacional utilizada para detecção de objetos em imagens, introduzida pela primeira vez em 1982 por Robert McConnel [27], e formalizada em 1994 por Freeman e Roth [28]. Trata-se do treinamento de um histograma de blocos descritivos dos gradientes que se espera observar em cada uma das regiões que se deseja reconhecer, e posterior comparação deste com os gradientes de áreas ou células da imagem de entrada.

Neste trabalho utilizaremos a F-HOG[1], uma técnica de HOG que consiste na composição de histograma com 36 descritores por célula utilizado para o treinamento de um classificador SVM (*Support Vector Machine*). Essa técnica consiste em, primeiramente, calcular os gradientes de cada ponto da imagem, normalizada

[1]F-HOG é uma implementação de detector HOG proposta por Felzenszwalb em seu artigo [29]. O identificador facial frontal da biblioteca Dlib (seção 3.1.1), utilizado neste trabalho, adota essa implementação.

quanto à cor através de sua conversão a escala de cinza. Esse cálculo é feito a partir da aplicação de um vetor de diferenciação $[-1, 0, +1]$ e seu transposto a cada pixel da imagem. O processo calcula um ângulo θ e intensidade r representativos do gradiente local em cada um dos pontos aos quais é aplicado.

A magnitude de cada gradiente é, então, discretizada conforme a direção entre p direções pré-definidas e adicionada às 4 células adjacentes do histograma, através de sua interpolação bilinear. A orientação final de cada bloco é então calculada a partir dos gradientes da região espacial que o compõem, por meio da soma de suas componentes. Isso garante a invariância do reconhecedor a pequenas alterações nas bordas, além de diminuir o tamanho deste, já que em geral o valor resultante das células tenderá a permanecer constante quando agrupadas.

$$\begin{aligned} N_{\delta,\gamma}(i,j) = \big(&\|C(i,j)\|^2 + \|C(i+\delta,j)\|^2 + \\ &+ \|C(i,j+\gamma)\|^2 + \|C(i+\delta,j+\gamma)\|^2\big)^{1/2}, \qquad \delta,\gamma \in \{-1,1\} \end{aligned} \tag{2.3}$$

$$H(i,j) = \begin{pmatrix} T_\alpha(C(i,j)/N_{-1,-1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,-1}(i,j)) \\ T_\alpha(C(i,j)/N_{+1,+1}(i,j)) \\ T_\alpha(C(i,j)/N_{-1,+1}(i,j)) \end{pmatrix} \tag{2.4}$$

É feita, então, a normalização dos blocos descritores. As componentes de cada célula são normalizadas em relação à energia de sua vizinhança (calculada através da equação 2.3), e truncadas em um valor $\alpha = 0.2$, conforme a equação 2.4. Isso produz uma matriz $H(i,j)$ de dimensão $p \times 4$ correspondente a cada célula do histograma, na qual cada elemento representa a normalização da intensidade do gradiente em uma das direções pré-definidas segundo os 4 critérios discutidos.

Os histogramas obtidos a partir dessa técnica são utilizados para treinamento de um classificador SVM.

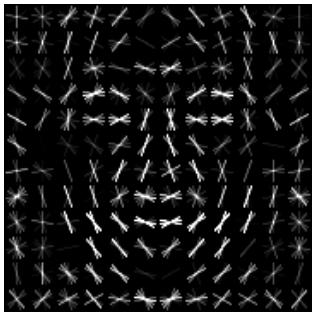

Figura 2.6: Um exemplo de HOG gerado a partir da imagem de um rosto. Imagem retirada de [30].

Para a detecção de objetos por meio desta técnica o processo de extração de

descritores é repetido, produzindo um histograma para a imagem alvo como um todo. O classificador visita cada região do histograma em diferentes resoluções por meio de uma *sliding window* piramidal [31], detectando as regiões desta que contém as mesmas características dos histogramas com os quais foi treinado. A implementação em forma de pirâmide da janela de reconhecimento permite que os objetos desejados sejam detectados de maneira independente de suas dimensões na imagem a partir de um único histograma treinado.

2.2.3 Alinhamento Facial

Alinhamento facial é uma técnica que vêm sendo bastante estudada nos últimos anos, com diversos métodos propostos para sua execução [32–35]. Originalmente introduzida em [36] por Yuille et al, a técnica busca a identificação de pontos representantes do contorno geométrico de diversas características de rostos em imagens, tais como os olhos, lábios, nariz, e sobrancelhas.

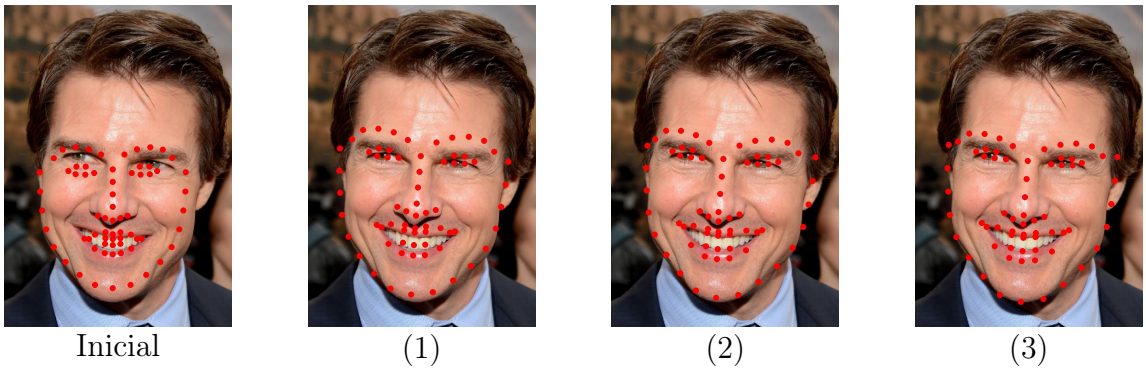

Figura 2.7: Estado do vetor de marcadores faciais em cada iteração do algoritmo de alinhamento facial.

Neste trabalho discutiremos o método proposto por Kazemi e Sullivan [37], que consiste em ajustar iterativamente um conjunto de pontos iniciais obtidos a partir dos dados de treinamento. Esse ajuste, modelado pela equação 2.5, é feito através da aplicação de uma cascata de florestas de árvores regressoras sobre a imagem de entrada, como mostra a figura 2.7.

$$\hat{S}^{(t+1)} = \hat{S}^{(t)} + r_t(I, \hat{S}^{(t)}) \tag{2.5}$$

Cada regressor consiste em uma árvore de decisão que, a partir da diferença de intensidade entre pares de pixels da imagem, é capaz de escolher um vetor de atualização a ser aplicado sobre o ponto resultante da iteração anterior. Esses vetores são pré-definidos e constantes, e são atribuídos às folhas da árvore. Nas primeiras iterações, estes são de maior magnitude, permitindo atualizações maiores nos pontos. Com cada iteração subsequente, são utilizados vetores de atualização menores, resultando em ajustes finos nos pontos.

As árvores são treinadas utilizando da técnica de *gradient boosting*. Os pixels a serem considerados como entrada de cada nível da cascata são escolhidos aleatoriamente durante o treinamento, com o único critério sendo a função exponencial de custo definida na equação 2.6. A aleatoriedade da seleção dessas características de entrada serve para tornar o algoritmo menos sensível a alterações na iluminação.

$$P(u, v) \propto e^{-\lambda \|u-v\|} \tag{2.6}$$

Essa implementação permite a localização de marcadores faciais em tempo real mesmo em imagens de baixa resolução, visto que estes não são propriamente identificados mas sim meramente posicionados sobre a imagem de entrada. A principal limitação deste algoritmo é que o mesmo só se aplica a imagens com composição semelhante às utilizadas durante o treinamento da cascata, devido à necessidade de escolha dos pontos a serem utilizados na decisão durante essa etapa.

Capítulo 3

Implementação

Neste capítulo discutiremos tópicos relacionados à implementação do sistema de diarização de locutor proposto.

Na seção 3.1 apresentamos as ferramentas principais utilizadas no desenvolvimento deste trabalho. Em seguida, na seção 3.2, apresentamos o trabalho realizado para preparação e pré-processamento dos dados. Na seção 3.3 discutimos a arquitetura do sistema desenvolvido, discussão esta que aprofundamos na seção 3.4 através da demonstração da topologia da rede neural utilizada.

3.1 Ferramentas

Nesta seção apresentamos as principais ferramentas utilizadas durante a implementação deste trabalho. Definiremos suas principais características, assim como as funcionalidades das mesmas que foram utilizadas, e as motivações por trás de sua escolha.

Na seção 3.1.1 apresentaremos a biblioteca Dlib, utilizada para propósito de identificação e demarcação facial no projeto. Em seguida, na seção 3.1.2 apresentaremos a biblioteca Tensorflow, utilizada por sua robusta implementação do modelo de rede neural convolucional. Na seção 3.1.3 apresentaremos o ambiente utilizado para treinamento do modelo, assim como seus recursos computacionais. Por fim, na seção 3.1.4, mencionaremos brevemente as demais ferramentas utilizadas em caráter pontual no trabalho, e que, por essa razão, não receberam seções dedicadas.

3.1.1 Dlib

A Dlib[38] é uma biblioteca de código aberto desenvolvida em C++ e com interface em Python. Trata-se de uma biblioteca generalista, contendo implementações de algoritmos para processamento paralelo, grafos, entre outros. Porém, seu foco principal se encontra nas áreas de aprendizado de máquina, processamento de imagem

e reconhecimento facial.

A biblioteca possui um identificador facial baseado em *Histogram of Oriented Gradients* (HOG) capaz de detectar rostos frontais mesmo em imagens de baixa resolução. As características específicas deste reconhecedor foram discutidas na seção 2.2.3.

Figura 3.1: Detecção de Marcadores Faciais pela biblioteca Dlib. Imagem publicada sob licença Creative Commons[39].

Além disso, a mesma implementa um detector de marcadores faciais baseado em uma floresta de árvores de regressão, capaz de identificar (ou estimar, caso não estejam visíveis) as posições de um conjunto de pontos em um rosto, como mostra a figura 3.1. A biblioteca fornece também um modelo pré-treinado para este detector para identificação de 68 destes marcadores, sob licença que permite uso acadêmico. Esta funcionalidade foi chave para a escolha da biblioteca para a implementação do trabalho, visto que permitiu o treinamento do classificador utilizando um conjunto menor de dados, sem preocupações quanto a vieses relacionados às características físicas do locutor.

A biblioteca Dlib foi utilizada em sua versão 19.19.0, compilada a partir do código fonte com suporte para GPU e otimizações referentes à arquitetura da CPU.

3.1.2 Tensorflow

Tensorflow[40] é um framework de código aberto para aplicações de aprendizado de máquina. A biblioteca, construída pela empresa Google, apresenta implementações robustas de diversos algoritmos da área, além de uma API que permite a declaração de uma rede neural em função de suas camadas. A biblioteca suporta, ainda, o uso de uma ou mais GPUs para treinamento da rede neural, através da biblioteca cuDNN. No trabalho, essa foi utilizada para assistir na modelagem da rede neural, assim como seu treinamento e posterior execução como parte do sistema completo.

A escolha desta biblioteca se deu devido à sua implementação de algoritmos chave para o desenvolvimento do trabalho, tais como a rede neural convolucional tridimensional, discutida na seção **??**. Além disso, suas interfaces nas linguagens de programação C++ e Python, assim como a facilidade de utilização de interface em Python para prototipagem rápida de redes neurais convolucionais com topologias diferentes foram decisivas para sua escolha para a realização do trabalho.

Nesse trabalho, utilizamos a versão do Tensorflow 2.1.0, compilada a partir do código fonte com suporte para GPU e otimizações do referentes à arquitetura da CPU.

3.1.3 Ambiente de Desenvolvimento

No desenvolvimento deste trabalho foi utilizada em caráter primário a linguagem de programação Python. Originalmente, o trabalho seria desenvolvido em C++ devido ao mais alto desempenho desta linguagem; no entanto, a utilização de diversas bibliotecas com interface em Python assim como o mais rápido desenvolvimento e prototipagem nesta linguagem levou à decisão final de utiliza-la para a implementação do projeto.

A IDE utilizada no desenvolvimento foi o Visual Studio Code, ferramenta de código aberto criada pela Microsoft, e o Jupyter Notebook[41], por sua capacidade de subdividir e visualizar o estado do programa em execução. ~~O treinamento da rede neural foi realizado em máquina com sistema operacional Windows 10, equipada com uma CPU Intel Core i7 de quarta geração, 12 GB de memória RAM e uma GPU Nvidia GTX 980 com 4 GB de memória dedicada.~~ O treinamento da rede neural foi realizado em máquina com sistema operacional Ubuntu 16.04, equipada com uma CPU Intel Core i7 de primeira geração, 12 GB de memória RAM, e uma GPU Nvidia GTX 1080 Ti com 12 GB de memória dedicada.

3.1.4 Outras Ferramentas

Nesta seção apresentamos as demais bibliotecas utilizadas no desenvolvimento do projeto. Em cada subseção descrevemos brevemente a biblioteca, definindo seu papel no projeto.

As bibliotecas se encontram ordenadas por sua função na pipeline do classificador, discutida de forma mais aprofundada na seção 3.3.

3.1.4.1 OpenCV

OpenCV[42] é uma biblioteca de código aberto para aplicações de Visão Computacional. Ela foi utilizada para realizar a leitura quadro a quadro dos arquivos de vídeo

a serem processados pelo sistema, e para codificar em vídeo a saída do classificador. Neste trabalho foi utilizada a biblioteca opencv-python em sua versão 4.2.0.32.

3.1.4.2 Matplotlib

A Matplotlib[43] é uma biblioteca para produção de gráficos e imagens em Python. Ela foi utilizada para produzir as imagens intermediárias, através do desenho de polígonos a partir dos vértices produzidos pelo processo de detecção de marcadores facial. Neste trabalho utilizamos a Matplotlib na versão 3.1.3 da biblioteca.

3.1.4.3 Pandas

Pandas[44] é uma biblioteca de processamento de dados em Python. Ela foi utilizada no pré-processamento dos dados com a finalidade de manipular os arquivos csv contendo a diarização manual dos vídeos do dataset de depoimentos. A biblioteca foi utilizada em sua versão 1.0.0.

3.2 Preparação dos Dados

Originalmente, foi fornecido pela Defensoria Pública do Estado do Rio de Janeiro um dataset contendo 29 vídeos, totalizando cerca de 5 horas de vídeo, referente a depoimentos prestados por diferentes participantes. Os vídeos fornecidos apresentam resolução de 320×240 pixels, a uma taxa de 30 quadros por segundo. O conjunto de dados não apresentava nenhuma anotação referente à fala dos depoentes.

Primeiramente, segmentamos os vídeos em fragmentos de 15 quadros, ou meio segundo. A motivação para a segmentação do vídeo de entrada em trechos deste comprimento foi a estipulação de que o tempo necessário para a fala da primeira sílaba em uma frase, por uma pessoa normal, no português do Brasil, é de 252 milissegundos[45]. Sendo assim, ao dobrar este tempo, adquirimos a capacidade de detectar por completo o movimento do locutor quando referente a frases curtas, tais como interjeições.

Estes trechos foram validados para determinar se a face do depoente poderia ser reconhecida, com finalidade de descartar fragmentos nos quais este não olhava em direção à câmera, ou nas quais o detector HOG não poderia identifica-los. Cada trecho foi então manualmente classificado quanto à ocorrência de fala pelo depoente, produzindo uma tabela que associava o identificador de cada arquivo à classe correspondente ao mesmo.

Feita essa diarização, os vídeos foram separados em conjuntos de teste e treinamento, tais que vídeos diferentes seriam utilizados para cada fase do treinamento da rede neural. Fizemos essa separação pois, como os fragmentos possuem relação

temporal tanto entre si quanto com outros vídeos do mesmo depoente, seria possível que o treinamento da rede neural utilizando segmentos semelhantes aos fragmentos de teste pudesse inflar artificialmente o desempenho da mesma.

Por fim, as diarizações produzidas manualmente foram utilizadas para organizar os fragmentos em uma estrutura de diretórios, tal que primeiramente estivessem separados por conjunto de teste ou treinamento, depois por vídeo de origem, e por fim por classe. Essa estrutura foi construída de tal forma que as informações relevantes ao gerador de amostras da rede neural pudessem ser obtidas todas a partir do caminho para o mesmo no sistema de arquivos.

Para a execução de todas essas tarefas foram desenvolvidos scripts em Python e Bash, capazes de segmentar os vídeos do dataset e de validar, mover, e organizar os fragmentos extraídos destes. O código referente a estes pode ser encontrado no apêndice A.

3.3 Arquitetura do Sistema

Foram desenvolvidos dois conjuntos scripts distintos para este trabalho; um sistema para treinamento da rede neural convolucional, e outro para a diarização de locutor em um vídeo com as características definidas anteriormente. Sendo assim, discutiremos a arquitetura utilizada para treinamento do modelo na seção 3.3.1, e a utilizada no processamento de uma mídia de entrada na seção 3.3.2.

3.3.1 Treinamento

A arquitetura do sistema de treinamento consiste em um gerador de dados e um modelo de rede neural convolucional a ser treinado.

A função primária do gerador de dados é carregar dinamicamente os vídeos a serem alimentados ao modelo em cada etapa de seu treinamento. Para isto, ele é inicializado com uma lista dos arquivos a serem carregados, assim como uma série de parâmetros que definem aspectos de sua operação, tais como o tamanho da batelada ou o pré-processamento a ser aplicado aos quadros.

O gerador de dados possui 3 modos distintos de operação. Estes são:

- RGB: Nesse modo, o quadro é codificado em padrão RGB, ou seja, com cores representadas por suas componentes vermelha, verde, e azul, nessa ordem.
- Grayscale: Nesse modo, o quadro RGB é convertido para escala de cinza, segundo a função $Y = 0.299R + 0.587G + 0.114B$.
- Landmarks: Nesse modo, é gerada uma imagem em preto e branco, rasterizada a partir dos pontos identificados pela detecção de marcadores faciais em

cada quadro carregado. Este foi o modo utilizado na implementação final do trabalho.

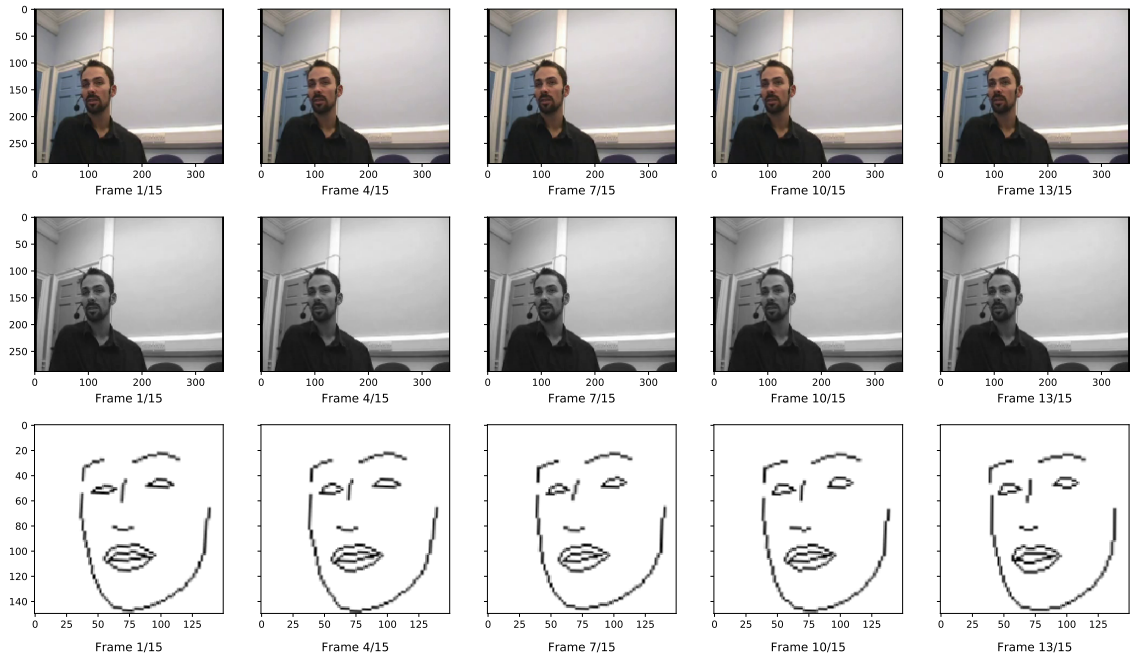

Figura 3.2: A saída do gerador de dados em cada um de seus três modos de operação.

O gerador é capaz, ainda, de manter um *cache* com os resultados de cálculos realizados no carregamento dos quadros, com a finalidade de possibilitar treinamento mais rápido do modelo em épocas ou execuções consecutivas.

Em nossa implementação final, utilizamos o modo de *landmarks*. Esse modo de operação implementa o seguinte algoritmo:

1. O gerador verifica se esse fragmento já foi processado anteriormente. Se a resposta for positiva, o resultado do processamento anterior é carregado do cache. Caso contrário, prossegue para a próxima etapa.

2. O segmento de vídeo é carregado do disco quadro a quadro.

3. Os quadros passam por um identificador facial, que retorna coordenadas de retângulos que contêm rostos no quadro original.

4. Os rostos identificados passam por um detector de marcadores faciais, que retorna o conjunto de 68 pontos correspondentes a seus marcadores.

5. Os pontos são, então, utilizados para o desenho de polígonos representativos do rosto sobre um canvas branco de mesmo tamanho da imagem original, na etapa à qual nos referimos como rasterização. Essa etapa garante a preservação

das relações entre os marcadores faciais, que seriam perdidas caso somente os pontos fossem utilizados.

6. A imagem rasterizada é recortada segundo o retângulo delimitado pelo identificador facial e alinhada horizontalmente.

7. O resultado da etapa anterior é escrito no cache, a ser utilizado em processamento futuro.

Em seguida, o modelo da rede neural é configurado. Os detalhes e decisões referentes à configuração deste podem ser encontrados na seção 3.4.

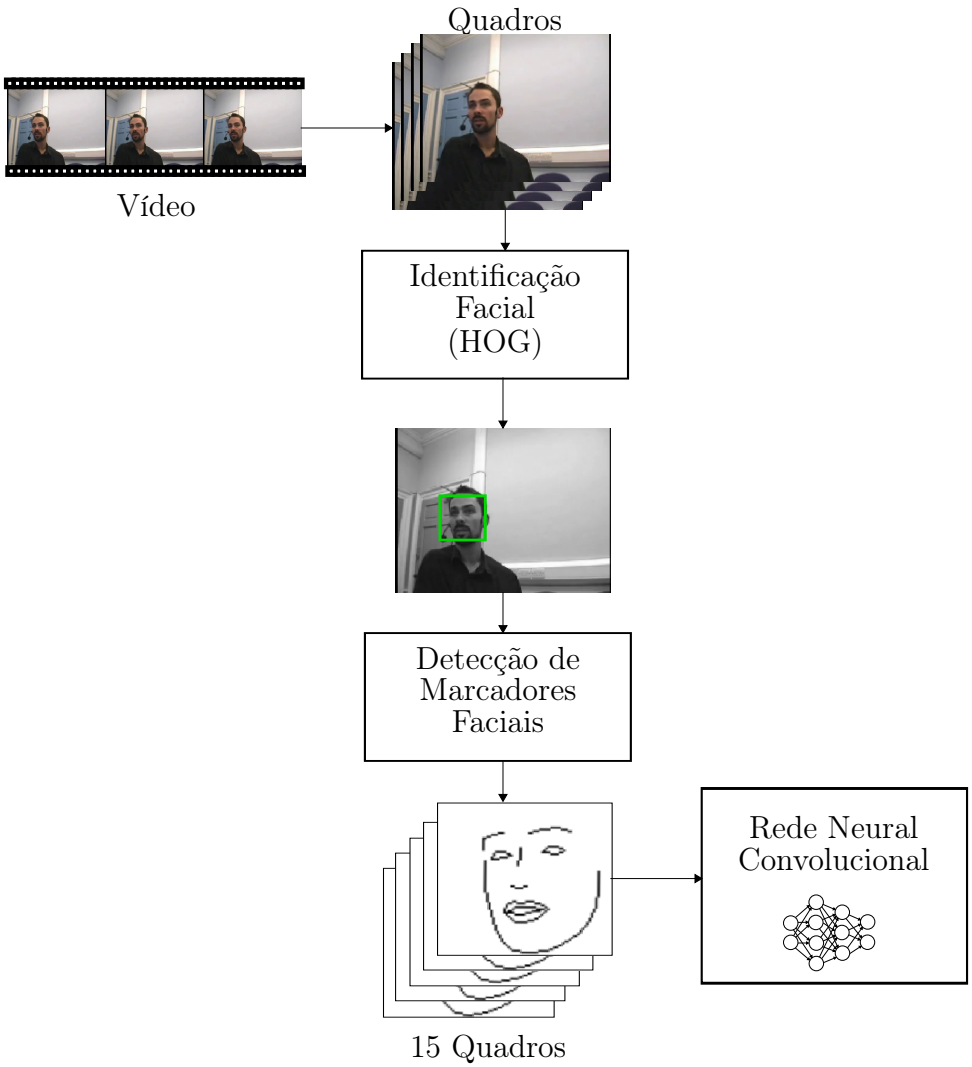

Figura 3.3: A arquitetura do sistema de treinamento.

O treinamento é então realizado em um loop. Para prevenção de *overfitting* na rede neural quanto ao conjunto de treinamento, paramos o mesmo quando o valor da função custo no conjunto de validação não fosse reduzido em até de 10 épocas e restauramos os pesos com melhor desempenho no conjunto de validação.

Como função de custo, dado que estamos lidando com um problema de classificação, utilizamos a função de entropia categórica cruzada, ou *custo softmax* (equação 3.1). Nessa equação, $\hat{y}$ é o vetor das probabilidades das categorias, e y é a categoria real, codificada *one-hot*.

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij})) \tag{3.1}$$

Visto que trata-se um problema de classificação binária, seria possível utilizar a função de entropia cruzada binária. Porém, considerando a possibilidade de expansão futura do modelo para detecção de outras ações conversacionais, tais como gestos com a cabeça, decidimos utilizar uma função custo mais escalável para adição de novas classes.

Em cada época, o modelo chama o gerador de dados de treinamento, que lhe retorna um número fixo de amostras (batelada). Cada uma dessas amostras é, como discutido anteriormente, um segmento de 15 quadros consecutivos, extraído de um vídeo e pré-processado pelo gerador. Essas amostras são alimentadas através do modelo, que faz predições quanto às mesmas e, através do algoritmo de retro propagação, ajusta seus pesos em função da corretude destas.

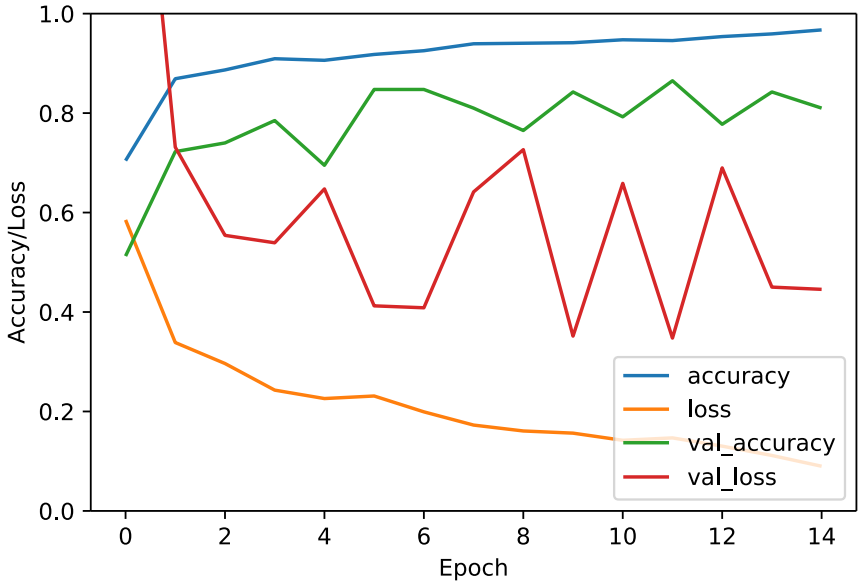

Figura 3.4: Evolução das métricas do treinamento ao longo do tempo. A métrica val_loss, valor da função custo sobre os resultados do conjunto de validação, foi critério de parada do treinamento.

Ao esgotarem as amostras de teste, o modelo chama, então, o gerador de dados de validação, realizando predições sem retro propagação sobre as entradas retornadas, e calculando o valor da função custo para os resultados. Por fim, ao se esgotarem também essas amostras, dá-se o final de uma época do treinamento, e os geradores

tem suas listas de arquivos embaralhadas de forma a garantir aleatoriedade na ordem da próxima época. Caso o valor da função custo não tenha sido reduzido nas últimas três épocas, o treinamento chega ao seu fim, e os pesos correspondentes ao melhor desempenho obtido no conjunto de validação são restaurados.

Como mostra a figura 3.4, que representa a evolução das métricas de desempenho durante o treinamento do modelo, o processo teve duração de 15 épocas.

3.3.2 Aplicação

3.4 Topologia da Rede Neural

Capítulo 4

Resultados

4.1 Dataset AMI Meeting Corpus

4.2 Métricas de Desempenho

4.3 Resultados

Capítulo 5

Conclusão

5.1 Trabalhos Futuros

Referências Bibliográficas

[1] A. J. Oxenham, "Pitch Perception," *Journal of Neuroscience*, vol. 32, no. 39, pp. 13 335–13 338, Sep. 2012.

[2] A. W. Zewoudie, J. Luque, and J. Hernando, "The use of long-term features for GMM- and i-vector-based speaker diarization systems," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, p. 14, Sep. 2018.

[3] I. Mccowan, G. Lathoud, M. Lincoln, A. Lisowska, W. Post, D. Reidsma, and P. Wellner, "The AMI Meeting Corpus," in *In: Proceedings Measuring Behavior 2005, 5th International Conference on Methods and Techniques in Behavioral Research. L.P.J.J. Noldus, F. Grieco, L.W.S. Loijens and P.H. Zimmerman (Eds.), Wageningen: Noldus Information Technology*, 2005.

[4] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, May 2011, conference Name: IEEE Transactions on Audio, Speech, and Language Processing.

[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Calgary, AB: IEEE, Apr. 2018, pp. 5329–5333.

[6] G. Sell and D. Garcia-Romero, "Speaker diarization with plda i-vector scoring and unsupervised calibration," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, Dec. 2014, pp. 413–417.

[7] Q. Wang, C. Downey, L. Wan, P. A. Mansfield, and I. L. Moreno, "Speaker Diarization with LSTM," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018, pp. 5239–5243.

[8] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan. 2013, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[9] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, Jun. 2014, pp. 1725–1732.

[10] J. Hershey, H. Attias, N. Jojic, and T. Kristjansson, "Audio-visual graphical models for speech processing," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference On*, vol. 5, Jun. 2004, pp. V–649.

[11] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–11, Jul. 2018.

[12] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[13] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," *Psychological Review*, pp. 65–386, 1958.

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[15] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, "Cardiologist-Level Arrhythmia Detection and Classification in Ambulatory Electrocardiograms Using a Deep Neural Network," *Nature medicine*, vol. 25, no. 1, pp. 65–69, Jan. 2019.

[16] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Predicting healthcare trajectories from medical records: A deep learning approach," *Journal of Biomedical Informatics*, vol. 69, pp. 218–229, May 2017.

[17] J. Hou, B. Adhikari, and J. Cheng, "DeepSF: Deep convolutional neural network for mapping protein sequences to folds," *Bioinformatics*, vol. 34, no. 8, pp. 1295–1303, Apr. 2018.

[18] M. Akram and C. El, "Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks," *International Journal of Computer Applications*, vol. 143, no. 11, pp. 7–11, Jun. 2016.

[19] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to End Learning for Self-Driving Cars," *arXiv:1604.07316 [cs]*, Apr. 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[20] B. Y. Peddagolla, S. S. Kathiresan, N. C. Doshi, N. S. Prabhu, and M. H. Zadeh, "On the Lane Detection for Autonomous Driving: A Computational Experimental Study on Performance of Edge Detectors," p. 10.

[21] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, Nov. 2018.

[22] S. Behnke, *Hierarchical Neural Networks for Image Interpretation*, Jan. 2003, vol. 2766, journal Abbreviation: Lecture Notes in Computer Science Publication Title: Lecture Notes in Computer Science.

[23] Dhp1080, "Ido: Skemo pri la kompozanti di un neurono." Oct. 2016. [Online]. Available: https://commons.wikimedia.org/wiki/File:Neurono-ido.svg#/media/File:Neurono-ido.svg

[24] S. E. Dreyfus, "Artificial neural networks, back propagation, and the Kelley-Bryson gradient procedure," *Journal of Guidance, Control, and Dynamics*, vol. 13, no. 5, pp. 926–928, Sep. 1990.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[26] Diliff, "English: Looking east from the rear of the nave of St Augustine's Church in Kilburn, a northern suburb of London, England." Feb. 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:St_Augustine%27s_Church,_Kilburn_Interior_1,_London,_UK_-_Diliff.jpg

[27] R. K. McConnell, "Method of and apparatus for pattern recognition," US Patent US4 567 610A, Jan., 1986, library Catalog: Google Patents. [Online]. Available: https://patents.google.com/patent/US4567610/en

[28] W. T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition," p. 9.

[29] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," p. 20.

[30] D. King, "Dlib 18.6 released: Make your own object detector!" library Catalog: blog.dlib.net. [Online]. Available: http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html

[31] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 886–893.

[32] X. Cao, Y. Wei, F. Wen, and J. Sun, "Face Alignment by Explicit Shape Regression," *International Journal of Computer Vision*, vol. 107, no. 2, pp. 177–190, Apr. 2014.

[33] L. Liang, R. Xiao, F. Wen, and J. Sun, "Face Alignment Via Component-Based Discriminative Search," in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5303, pp. 72–85, series Title: Lecture Notes in Computer Science.

[34] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool, "Real-time facial feature detection using conditional regression forests," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 2578–2585.

[35] Xiangxin Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Providence, RI: IEEE, Jun. 2012, pp. 2879–2886.

[36] A. L. Yuille, P. W. Hallinan, and D. S. Cohen, "Feature extraction from faces using deformable templates," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 99–111, Aug. 1992, company: Springer Distributor: Springer Institution: Springer Label: Springer.

[37] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH: IEEE, Jun. 2014, pp. 1867–1874.

[38] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[39] MTheiler, "Detecção de Marcadores Faciais pela biblioteca Dlib," Jan. 2019. [Online]. Available: https://commons.wikimedia.org/wiki/File:Dlib-face_landmark_detection.jpg

[40] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: https://www.tensorflow.org/

[41] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter Notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87–90.

[42] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[43] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, May 2007, conference Name: Computing in Science Engineering.

[44] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and Jarrod Millman, Eds., 2010, pp. 51–56.

[45] P. A. Barbosa, ""Syllable-timing in Brazilian Portuguese": Uma crítica a Roy Major," *DELTA: Documentação de Estudos em Lingüística Teórica e Aplicada*, vol. 16, no. 2, pp. 369–402, 2000.

Apêndice A

Código Fonte