Universidade Federal
do Rio de Janeiro
Escola Politécnica

DIARIZAÇÃO 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
Agosto de 2020

DIARIZAÇÃO 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. Geraldo Zimbrão da Silva, D.Sc.

Prof. Jano Moreira de Souza, Ph.D.

Prof. Heraldo Luís Silveira de Almeida, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2020

Basilio, Renan Fasolato

Diarização 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.

XII, 43 p.: il.; 29, 7cm.

Orientador: Geraldo Zimbrão da Silva

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

Referências Bibliográficas: p. 38 – 43.

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.

Aos meus pais, que sempre me apoiaram, e minha noiva Allison, que esteve sempre ao meu lado e esperou por mim durante toda a minha jornada universitária.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação.

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

Renan Fasolato Basilio

Agosto/2020

Orientador: Geraldo Zimbrão da Silva

Curso: Engenharia de Computação e Informação

Este trabalho apresenta uma prova de conceito para um sistema diarizador baseado em uma rede neural convolucional capaz de identificar o estado de fala de um locutor a partir de um vídeo do mesmo, sem fazer uso da onda de áudio relacionada, para aplicação em casos onde esta se encontre em baixa qualidade, ruidosa, ou mesmo ausente. Para isso, é realizado um pré-processamento sobre a imagem de entrada de forma a identificar a posição da face do locutor e extrair desta suas feições principais, que servem de entrada para a rede neural. Uma arquitetura para a rede neural baseada em uma VGG, modificada para lidar com dados tridimensionais, foi construída, cuja implementação levou a um modelo com acurácia preditiva de 86.56%, resultando em uma taxa de erro de diarização de 32.5 sobre os dados de teste no melhor caso.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

FACIAL EXPRESSION ANALYSIS FOR SPEAKER DIARIZATION AND IDENTIFICATION IN VIDEO CONTENT VIA SUPERVISED MACHINE LEARNING

Renan Fasolato Basilio

August/2020

Advisor: Geraldo Zimbrão da Silva

Course: Computer Engineering

This work introduces a proof of concept for a speaker diarization system based on a convolutional neural network capable of identifying speech in a frontal video of a speaker without making use of the associatied audio wave, for use cases in which the latter is either low quality, noisy, or outright missing. For this purpose we extract facial landmarks from each frame of the input video, and feed them into the neural network. We also propose an architecture for the prediction model based on a VGG deep neural network modified to handle three-dimensional data, with which we obtained a predictive accuracy of 86.56% on the test dataset, which resulted in a diarization error rate of 32.5.

Sumário

Lista de Figuras

Lista de Tabelas

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*[5], 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 [6], 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% 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, intuição da qual se encontra demonstrada na figura 1.1. 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.

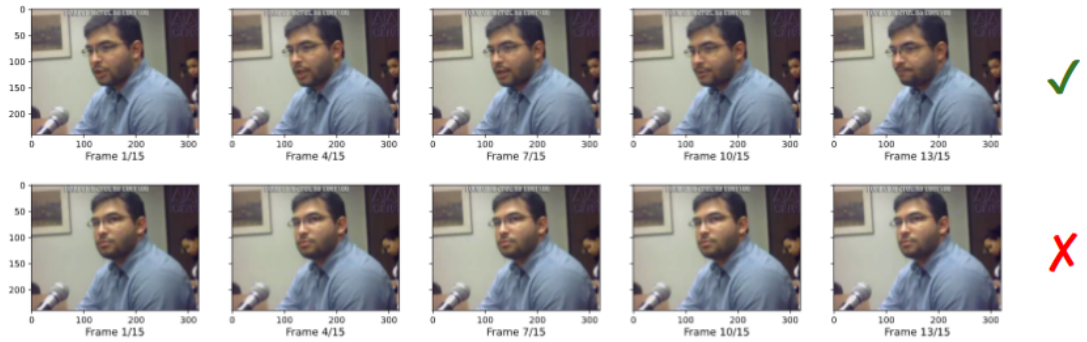


Figura 1.1: Intuição por trás desta proposta de solução. Note a facilidade de se identificar corretamente a fala do locutor mesmo quando apresentados apenas poucos quadros de um segmento contínuo.

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 um 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, que contém vídeos dos participantes individuais 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.

1.6 Código Fonte

Todo o código fonte desenvolvido para este trabalho pode ser obtido online no repositório do GitHub em `https://github.com/RenanBasilio/SpeechActionClassifier`.

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* [7], extraídos por meio de transformações matriciais aplicadas sobre o sinal, ou *X-Vectors* [8], obtidos através do uso de redes neurais profundas. A partir de então, estes vetores podem ser clusterizados [9] quando informações referentes ao número de locutores é conhecido, ou, mais recentemente, utilizados como entrada de uma rede neural LSTM [10].

No entanto, a introdução das redes neurais convolucionais para reconhecimento de ações humanas em sinais de vídeos [11, 12] 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 [13], 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 [14].

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[15]. Esse modelo continuou evoluindo ao longo dos anos, culminando no desenvolvimento do *Perceptron* por Rosenblatt em 1958 [16] (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 [17], um tipo de rede neural composto por múltiplas camadas de neurônios densamente conectadas.

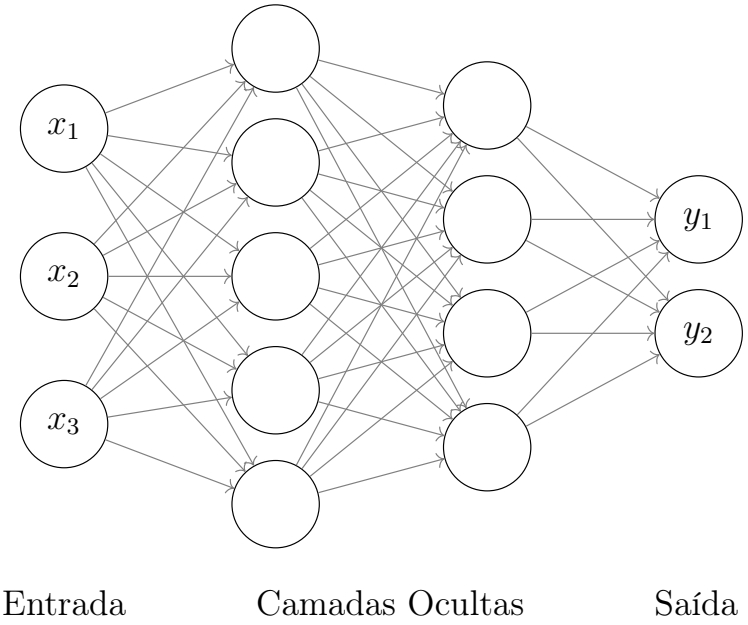


Figura 2.1: Ilustração de uma rede neural profunda com 3 camadas.

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 [18–20], previsão do tempo [21], veículos autônomos [22, 23], entre outras [24].

Tratam-se de algoritmos próprios para problemas que envolvem o reconhecimento de padrões, e a capacidade de treinalos 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 prédeterminado 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 [25].

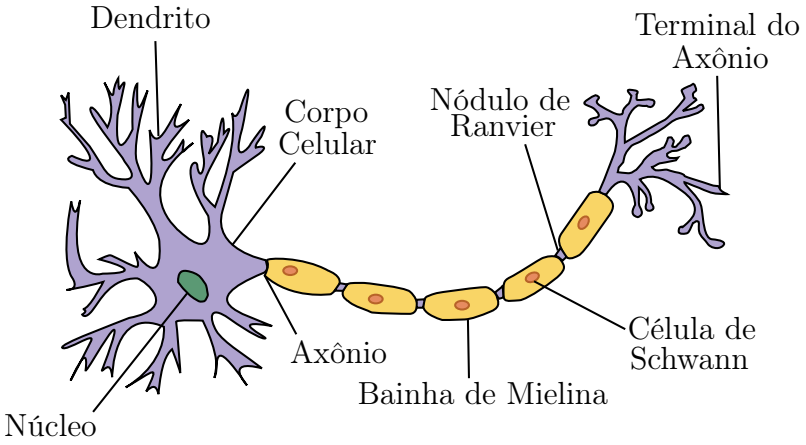


Figura 2.2: A estrutura de um neurônio biológico. Imagem adaptada de [1].

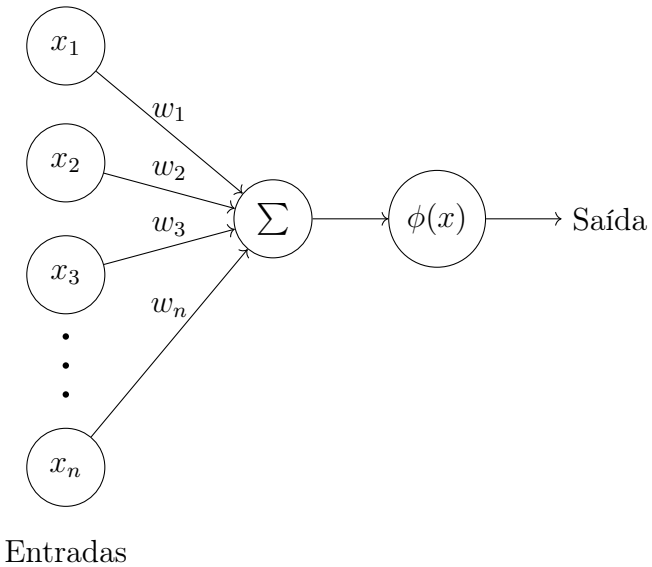


Figura 2.3: O *Perceptron*, uma rede neural com um único neurônio.

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.

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 [26]. 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 [27].

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 [23], 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.

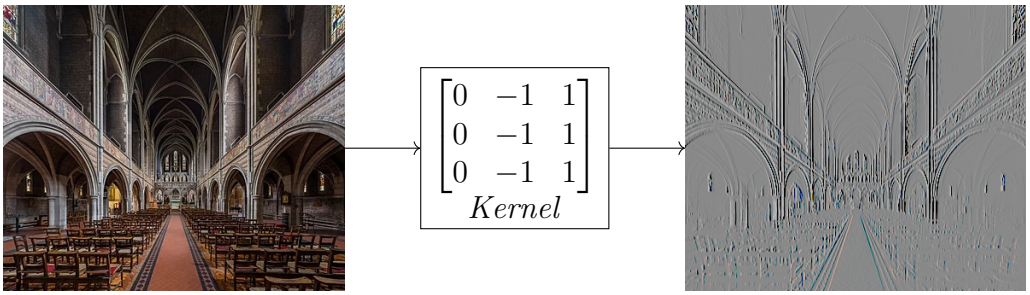


Figura 2.4: O resultado da aplicação de um *kernel* simples para detecção de bordas verticais a uma foto. O *kernel* é aplicado a cada pixel da imagem, substituindo o valor deste pela soma de sua vizinhança ponderada pelos valores da matriz. Foto original obtida de [2].

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.

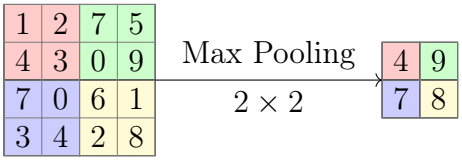


Figura 2.5: O resultado da operação de *Max Pooling* com dimensão 2×2 sobre uma matriz de entrada.

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 [28], e formalizada em 1994 por Freeman e Roth [29]. 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 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.

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

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

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.

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

[1]F-HOG é uma implementação de detector HOG proposta por Felzenszwalb em seu artigo [30]. O identificador facial frontal da biblioteca Dlib (seção 3.1.1), utilizado neste trabalho, adota essa implementação.

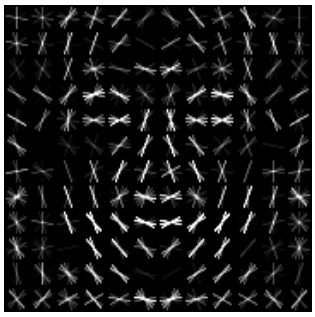


Figura 2.6: Um exemplo de HOG gerado a partir da imagem de um rosto. Imagem retirada de [3].

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.

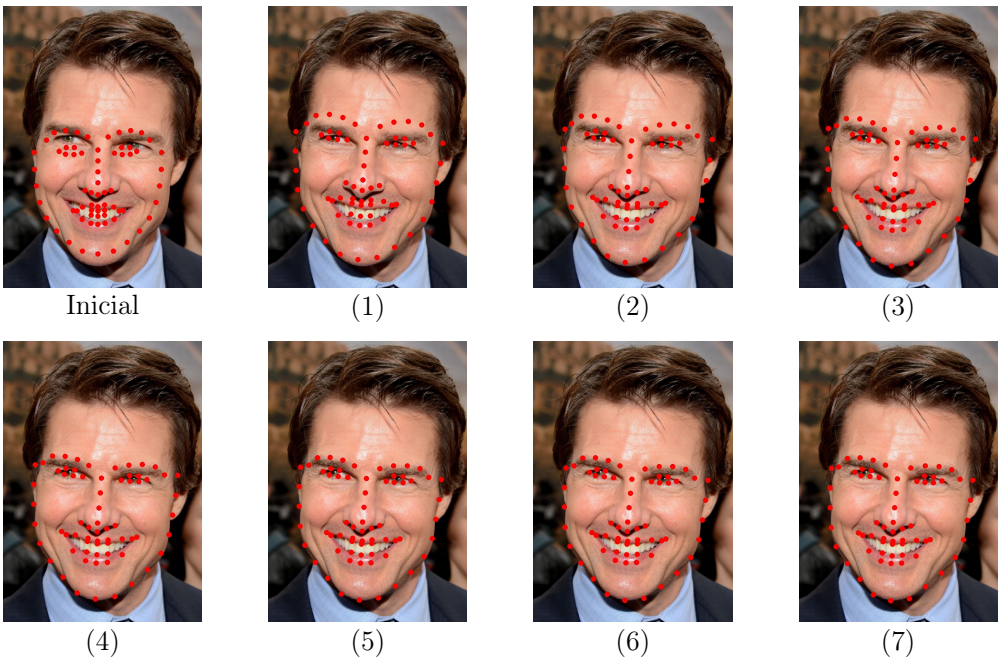


Figura 2.7: Estado do vetor de marcadores faciais ao final das primeiras 7 iterações do algoritmo de alinhamento facial.

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.

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, e a utilizada no processamento de uma mídia de entrada na seção 3.4.

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[4].

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[39] 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 2.2.1. 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[40], 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 Ubuntu Linux 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[41] 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[42] 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[43] 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.

3.1.4.4 Scikit Learn e dscore

As bibliotecas Scikit-Learn[44], biblioteca de aprendizado de máquina tradicional em python, e dscore[45], biblioteca para cálculo de métricas de diarização, foram utilizadas para o cálculo de diversas métricas relacionadas ao desempenho do sistema. Essas foram utilizadas em suas versões 0.22.2.post1 e 1.1.0, respectivamente.

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[46]. 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 manual, os vídeos foram separados em conjuntos de treinamento e teste, tal que fragmentos provenientes de um mesmo vídeo nunca seriam utilizados em ambas as fases do processo. 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.

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.

3.3 Treinamento

Para o treinamento do modelo foi desenvolvido um sistema composto por dois componentes principais.

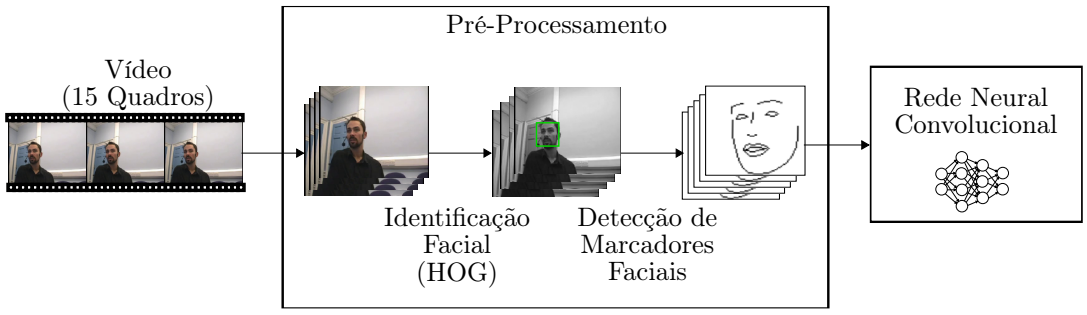


Figura 3.2: A sequência de carregamento dos dados e treinamento do modelo.

Primeiramente, foi construído um gerador de dados com a finalidade de alimentar os dados da entrada ao modelo dinamicamente. As características e funcionalidades se encontram descritas na seção 3.3.1. Depois, foi definida a topologia do modelo de rede neural, discutida na seção 3.3.3, e os critérios e parâmetros relevantes ao fluxo de treinamento, discutidos na seção 3.3.4.

3.3.1 Carregador de Dados

A função primária do carregador de dados é ler dinamicamente os vídeos a serem alimentados ao modelo em cada etapa de seu treinamento. Um módulo capaz de realizar tal carregamento dinâmico foi necessário devido ao grande volume de dados sendo processados, por virtude de se tratarem de dados de vídeo. 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.

Cada vídeo fornecido ao carregador é, ainda, invertido horizontalmente. Isso é feito para garantir que o modelo fosse treinado para funcionar independentemente do ângulo do rosto do sujeito, desde que reconhecível pelo HOG, visto que na grande maioria dos vídeos do dataset original o falante se encontrava olhando para o lado esquerdo da câmera ou diretamente para esta.

Para acelerar o processo de identificação facial e detecção de marcadores faciais, o carregador é capaz de processar múltiplos fragmentos em paralelo na CPU, além de manter um cache dos fragmentos processados em iterações anteriores do treinamento. Esse paralelismo possibilitou a redução do tempo de execução da primeira iteração do treinamento e teste de cerca de 9 horas para aproximadamente uma hora e trinta minutos, uma redução de 83% no tempo de execução com 6 threads, enquanto a implementação do cache reduziu o tempo necessário para etapas subsequentes do processo de para cerca de 3 minutos.

3.3.2 Pré-Processamento

O carregador de dados possui 3 modos distintos de operação, demonstrados na figura 3.3, que correspondem aos filtros de pré-processamento que podem ser aplicados aos quadros carregados para compor a entrada do modelo de predição. Estes são:

(1) RGB: Nesse modo, o quadro é codificado em padrão RGB, ou seja, com cores representadas por suas componentes vermelha, verde, e azul, nessa ordem.

(2) Grayscale: Nesse modo, o quadro RGB é convertido para escala de cinza, segundo a função $Y = 0.299R + 0.587G + 0.114B$.

(3) 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 3.3: A saída do carregador de dados em cada um de seus três modos de operação.

Algoritmo 1: Pré-processamento de um fragmento de vídeo (landmarks).

Resultado: Sequência de 15 quadros pré-processados

enquanto *Quadros Carregados* < 15 **faça**

- Carrega quadro do fragmento;
- Detecta faces no quadro carregado usando o detector facial (HOG);
- Calcula os 68 marcadores faciais utilizando o algoritmo de alinhamento facial;
- Rasteriza os marcadores obtidos sobre um canvas branco;
- Recorta o canvas segundo a região identificada pelo HOG;
- Alinha horizontalmente a face detectada e amplia o desenho;

fim

Em nossa implementação final, utilizamos o modo de *landmarks*. Esse modo implementa o algoritmo 1.

3.3.3 Topologia do Modelo

Para o modelo de predição, definimos uma topologia baseada na arquitetura VGG[1], definido na figura 3.4, que consiste em um número de pares de camadas de convolução e *max pooling*, seguidas de camadas densamente conectadas. A arquitetura

[1]VGG é um estilo de arquitetura de rede neural convolucional para reconhecimento de objetos em fotos, desenvolvida pelo Grupo de Geometria Visual (do inglês *Visual Geometry Group*) da universidade de Oxford [47].

foi adaptada para lidar com dados de caráter tri-dimensional. Nesta seção iremos discutir as decisões tomadas neste processo de adaptação da arquitetura, e os fatores que levaram a estas decisões.

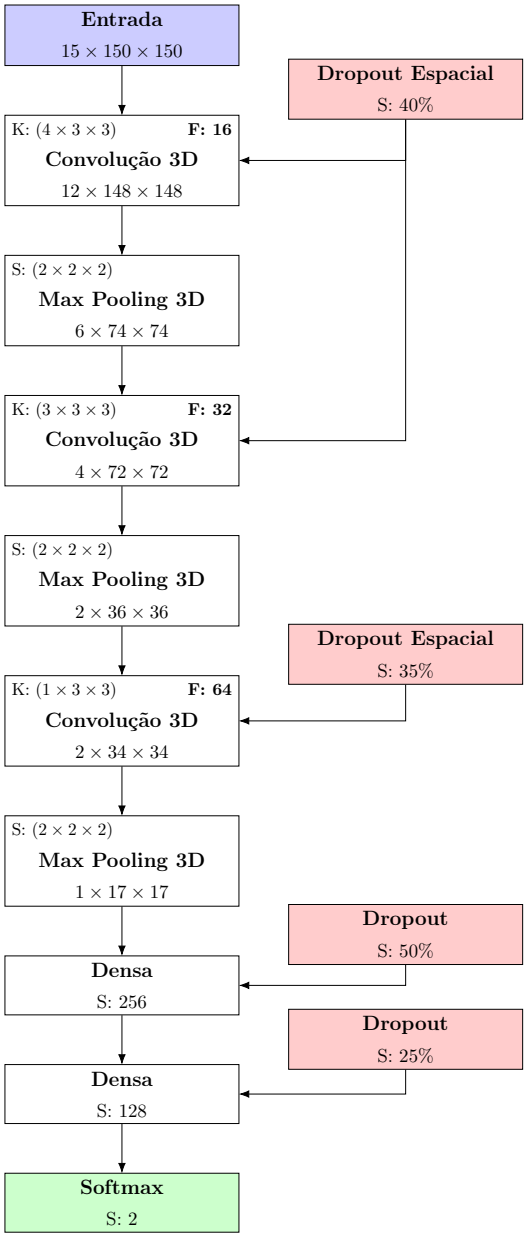


Figura 3.4: Topologia do Modelo

Primeiramente, na primeira camada convolucional da rede, que frequentemente tem a função de detectar bordas na imagem de entrada, adicionamos uma componente temporal ao *kernel* com dimensão 4. Essa dimensão par não é usual para uma rede neural convolucional, que tradicionalmente busca identificar padrões nas vizinhanças de cada elemento, devido ao fato de que não atua sobre um elemento da imagem original alterando-o em relação a seus vizinhos. Ao invés disso, a di-

mensão par produz um novo elemento, localizado espacialmente em seu centro, e cujo valor é obtido a partir da média ponderada pelos pesos do *kernel* de células da entrada. Em nosso modelo isso corresponde a construir um conjunto de filtros capazes de identificar variações temporais relevantes em cada sequência de 4 quadros da entrada.

Na segunda camada de convolução, adicionamos uma componente temporal com dimensão 3, igual às demais. Isto consiste em avaliar o valor de cada ponto da matriz de entrada em função dos valores de seus vizinhos, o que possibilita a identificação das áreas da matriz relevantes ao problema. A baixa dimensão utilizada em relação ao tamanho da entrada faz com que as características observadas sejam características locais, tais como bordas e cantos nas quais ocorram as variações desejadas já previamente identificadas.

Na última camada de convolução, mantivemos caráter bi-dimensional do *kernel* através da dimensão temporal igual a 1, apesar de constituir uma convolução tri-dimensional. Isso corresponde a operar individualmente sobre os quadros da entrada, agora compostos pela combinação temporal dos quadros originais, buscando características destes que sejam relevantes para a classificação, a ser realizada em seguida pelas camadas densas.

Finalmente, o número de neurônios das camadas totalmente conectadas foi reduzido a partir dos valores originais propostos na VGG16. Essa redução foi realizada de forma iterativa, buscando manter a acurácia do modelo constante e reduzir o *overfitting* através da eliminação de parâmetros redundantes.

3.3.4 Treinamento do Modelo

Em cada época, o sistema de treinamento do modelo chama o carregador de dados de treinamento, que lhe retorna um número fixo de amostras (batelada) compostas por segmentos de 15 quadros consecutivos. O modelo então faz suas predições quanto a estas, e tem seus pesos ajustados através do algoritmo de retro-propagação de forma a minimizar uma função de custo.

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

Como se trata de um problema de classificação, escolhemos como função custo do treinamento a função de entropia categórica cruzada, definida na equação 3.1. Nessa equação, $\hat{y}$ é o vetor das probabilidades das categorias, e y é a categoria real, codificada *one-hot*[2].

[2]Codificação *one-hot* é uma forma de representar a classe dentre um conjunto de classes à qual um item pertence, na forma de uma sequência binária na qual um único bit, correspondente a esta, esteja no nível lógico 1.

Visto que trata-se um problema de classificação binária, seria possível também 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 escalável para adição de novas classes.

Como forma de reduzir o *overfitting* da rede neural aos dados do treinamento, aplicamos *Dropout*[3] às camadas totalmente conectadas desta, e *Dropout Espacial* à saída de suas camadas convolucionais. A figura 3.5 mostra o resultado da utilização dessas técnicas no treinamento do modelo.

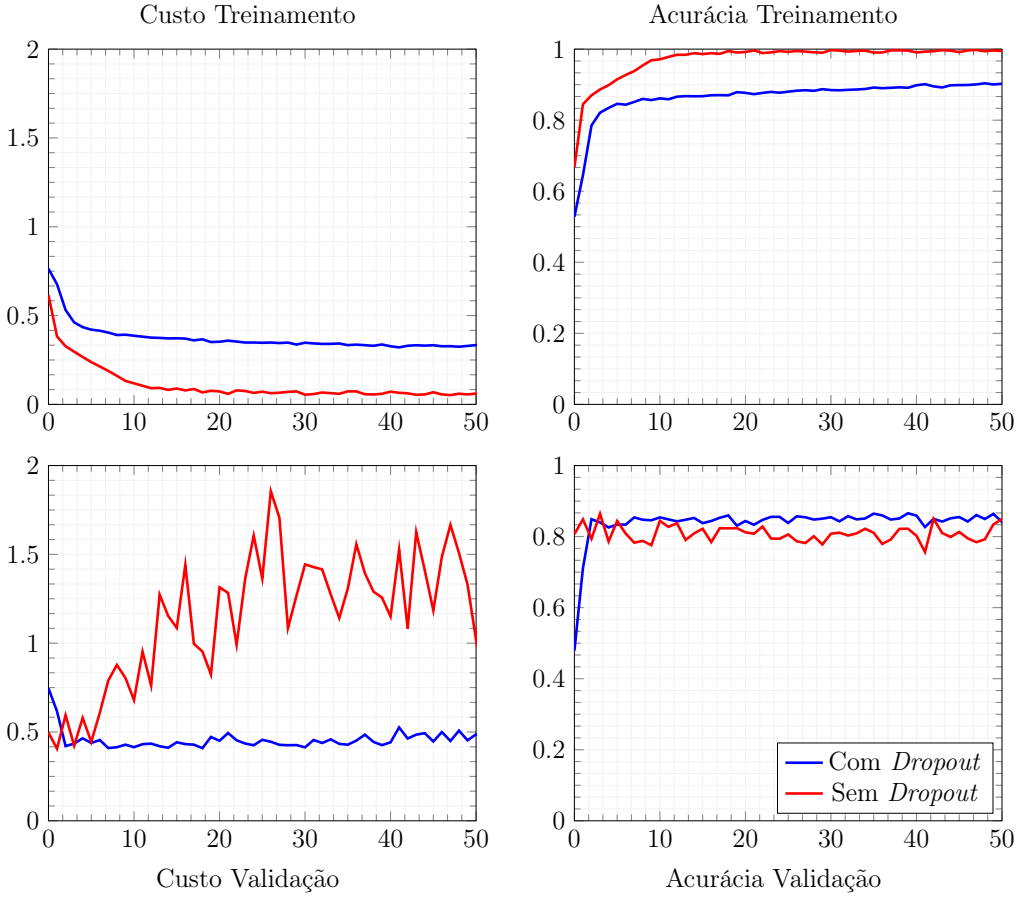


Figura 3.5: Evolução das métricas de desempenho ao longo dos processos de treinamento e validação do modelo, com e sem *dropout*, representado pelas linhas azul e vermelha respectivamente. Note que, sem *dropout*, a acurácia no conjunto de treinamento tende a 1 e sua função custo tende a 0, enquanto a função custo no conjunto de validação não converge.

Além disso, aplicamos regularização L2 sobre o *kernel* e *bias* de todas as camadas da rede. Essa regularização consiste em acrescentar uma penalidade sobre o ajuste do valor de ativação de cada neurônio x_i, definida na equação 3.2, ao valor da função

[3] *Dropout* é uma técnica que consiste na remoção aleatória de neurônios temporariamente de uma camada da rede neural durante o treinamento. *Dropout Espacial* é uma variação desta técnica, que consiste em remover neurônios agrupados por região espacial.

custo definida anteriormente.

$$p_{l_2} = l_2 \sum_{i=0}^{n} (x_i)^2 \tag{3.2}$$

Uma vez esgotadas as amostras de treinamento, o modelo chama, então, o carregador de dados de validação, realizando predições sem retro-propagação sobre as entradas retornadas, e calculando o valor da função custo para os resultados. Esta etapa é fundamental para garantirmos que o modelo treinado seja aplicável a dados do mundo real, e não exclusivamente aos dados do conjunto de treinamento.

Por fim, ao se esgotarem também essas amostras, dá-se o final de uma época do treinamento, e os carregadores tem suas listas de arquivos embaralhadas de forma a garantir aleatoriedade na ordem da próxima época. Ainda para prevenção de *overfitting* na rede neural foi definido como critério de parada do treinamento a condição de que não haja redução da função custo durante cinquenta épocas do treinamento.

3.4 Aplicação

A aplicação de diarização resultante deste trabalho é capaz de, quando executada sobre um arquivo de vídeo, diarizar o mesmo produzindo um arquivo em formato RTTM[4] identificando os períodos correspondentes à fala do orador. Essa aplicação implementa o algoritmo 2, que define uma janela deslizante que percorre o vídeo de entrada, avançando a cada iteração um número fixo de quadros determinado pelo parâmetro *step*. A figura 3.6 demonstra a execuçã das duas primeiras iterações do algoritmo para a diarização de uma mídia completa.

O algoritmo utilizado para pré-processamento dos quadros já foi discutido na seção 3.3.2, e as características do modelo de classificação foram discutidas na seção 3.3.3. Sendo assim, discutiremos nesta seção, primariamente, as considerações feitas durante o processo de desenvolvimento da aplicação quanto ao algoritmo utilizado para consolidação das predições realizadas para os quadros individuais, e o fluxo de erro adotado quando não é possível obter uma classificação para um determinado quadro.

3.4.1 Consolidação das Classificações

Com a finalidade de obter maior precisão temporal do que seria possível apenas com a classificação de sequências de 15 quadros, a diarização é feita com tamanho

[4]O formato RTTM, do inglês *Rich Transcription Time Marked*, é um formato de arquivo texto definido pela NIST em 2009 para representação de transcrições. Sua especificação formal pode ser encontrada em [48], publicado pela organização.

Algoritmo 2: Algoritmo de diarização de mídia.

Resultado: Diarização completa da mídia de entrada.
enquanto *Mídia de entrada possui quadros* **faça**
  enquanto *Quadros Carregados < 15* **faça**
    Carrega quadro da mídia de entrada;
    Aplica pré-processamento sobre o quadro carregado;
    Adiciona quadro carregado à janela;
  fim
  Realiza predição sobre os quadros da janela utilizando o modelo classificador;
  Armazena resultado em listas correspondentes aos quadros;
  Consolida os *step* quadros mais antigos;
  Descarta os *step* quadros mais antigos;
fim

de passo $step \leq 15$ tal que cada quadro passe pelo classificador um número $n = \lfloor 15/step \rfloor$ de vezes, em diferentes posições no fragmento. Dessa forma, torna-se necessário consolidar as diversas predições realizadas sobre cada quadro ao adicionalos à transcrição final.

Para esse processo, implementamos 5 algoritmos distintos. Estes consistem em funções que, quando aplicadas a um conjunto de predições de tamanho variável, retornam um único conjunto de confianças correspondentes a cada classe alvo. Com esse propósito, implementamos as funções *valor central*, *média*, *frequência de predição*, *média gaussiana*, e *frequência gaussiana*. A classe de maior confiança retornada por estes algoritmos é considerada a classe real do quadro, e é adicionada à diarização.

3.4.1.1 Valor Central

O algoritmo de valor central é o algoritmo de consolidação mais simples entre os algoritmos implementados. Este consiste em selecionar as confianças de cada classe referentes à predição realizada com o quadro no centro do fragmento.

3.4.1.2 Média

Este algoritmo consiste em calcular a média aritmética das confianças retornadas pela camada *softmax* do modelo preditor para cada classe. Essa média é utilizada como nova confiança do quadro.

3.4.1.3 Frequência

O algoritmo de frequência é bastante semelhante ao algoritmo de média, no entanto ao invés de fazer uso das confianças retornadas pelo modelo este considera as

predições deste absolutas e seleciona a predição mais frequente para cada quadro individual.

3.4.1.4 Média e Frequência Gaussiana

Estes algoritmos consistem em implementações dos algoritmos de média e frequência já analisados anteriormente nas quais os valores das predições individuais são ponderados a partir de pesos obtidos a partir de valores de uma distribuição normal (equação 3.3) em $\lfloor 15/step \rfloor$ pontos distribuídos linearmente em $[-2, 2]$.

$$\varphi = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \tag{3.3}$$

A intuição por trás da inclusão destes algoritmos foi de que predições realizadas com o quadro em questão próximo ao centro do fragmento seriam mais representativas da real classe desse do que predições realizadas com o mesmo nas extremidades do fragmento. O uso de uma distribuição normal, então, enfatizaria as confianças ou frequências dessas classificações.

3.4.2 Fluxo de Erro e Dados Ausentes

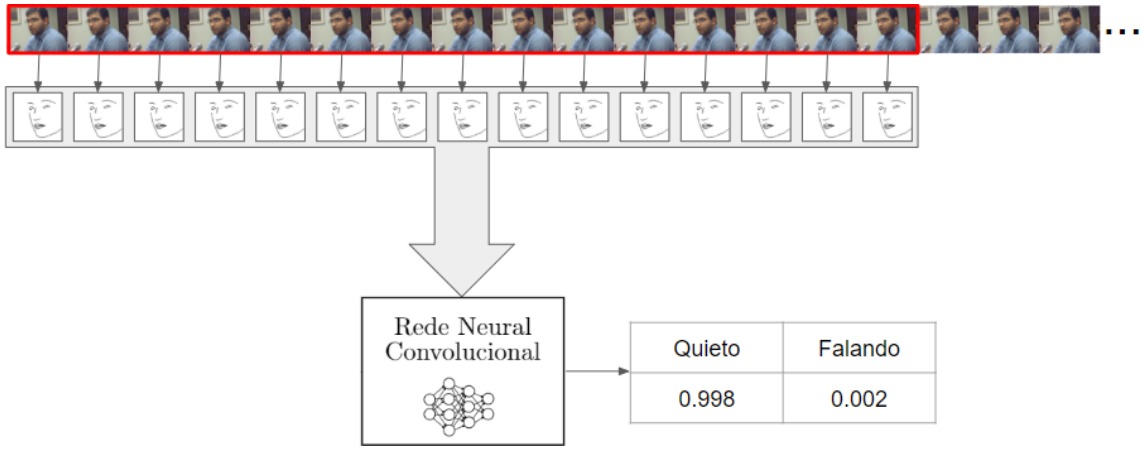
Caracterizamos como dado ausente todo quadros no qual o algoritmo de detecção facial não é capaz de detectar um rosto na imagem. Isso pode ocorrer em diversas situações, desde um movimento por parte do orador que esconda sua face até sua real ausência na gravação. Note que neste momento não consideramos detecções incorretas como um dado ausente.

Uma vez que o diarizador encontra um dado ausente, torna-se impossível completar uma sequência de 15 quadros para a realização de novas classificações. É necessário, então, consolidar as predições quanto a todos os quadros já feitas anteriormente. Isso é feito utilizando os mesmos algoritmos já discutidos na seção 3.4.1.

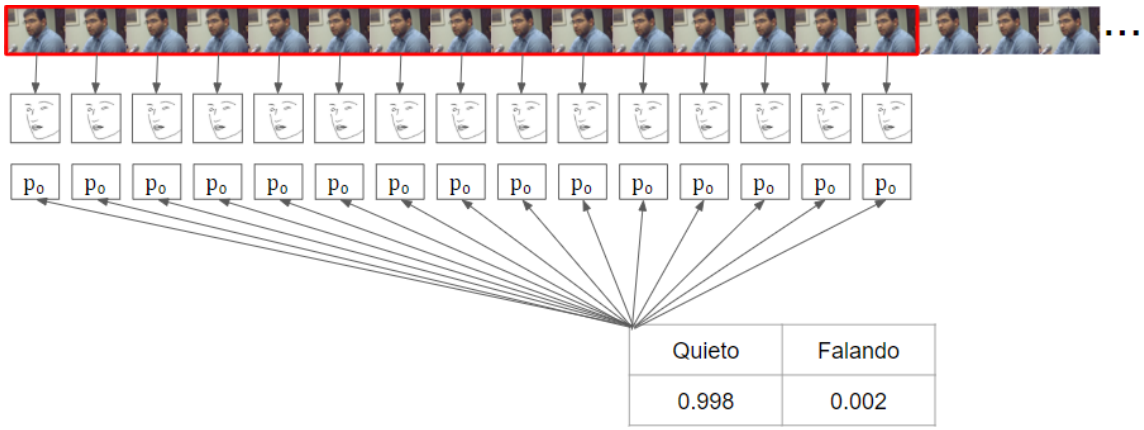
O quadro que iniciou o erro é, então adicionado à diarização com classe desconhecida. Quadros subsequentes que também não possuam faces identificáveis incrementam a duração deste trecho de diarização desconhecida. Uma vez carregados consecutivamente 15 novos quadros válidos, essa é encerrada e o diarizador retorna ao fluxo normal.

Adicionalmente, é também considerado com erro qualquer quadro que, ao ter suas classificações consolidadas, apresente confianças iguais para duas ou mais classes.

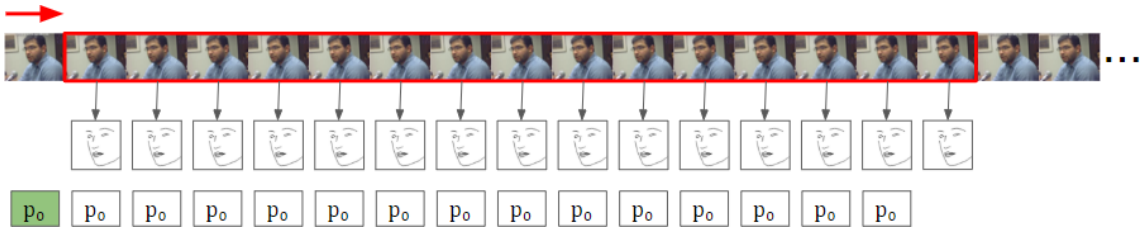
(a) Os 15 primeiros quadros são carregados e pré-processados, produzindo bitmaps traçados a partir dos marcadores faciais detectados.



(b) A janela é classificada pela rede neural, que retorna as probabilidades de cada classe para essa.

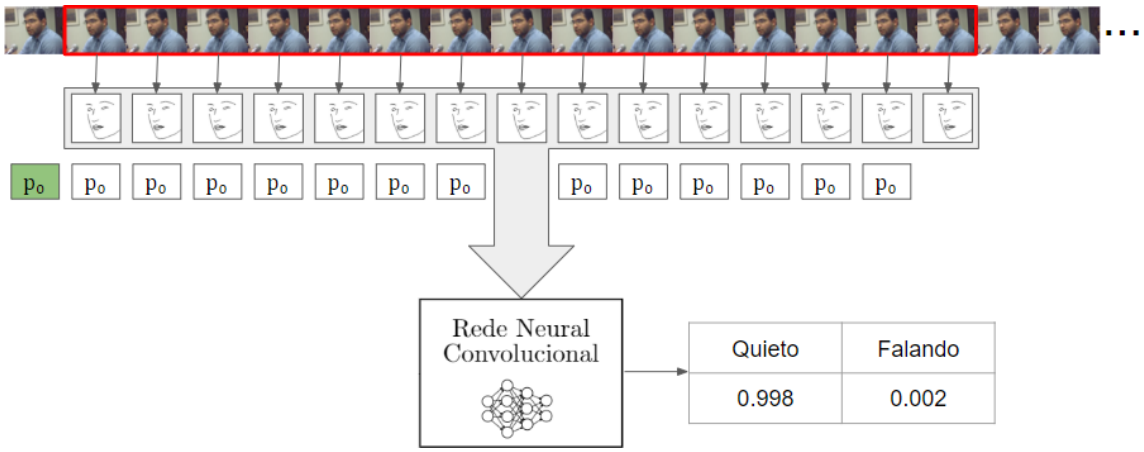


(c) As probabilidades retornadas pelo classificador são associadas a cada um dos quadros.

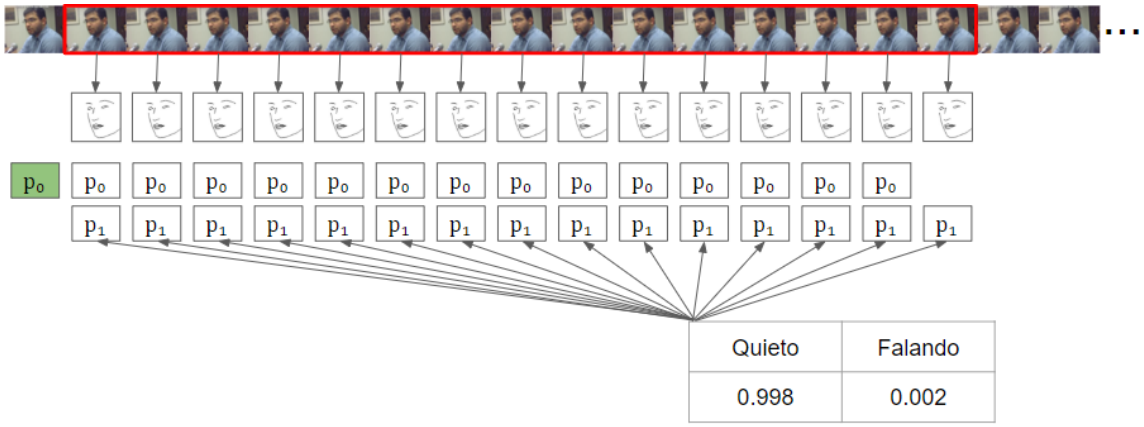


(d) A janela desliza um quadro para a direita, e o próximo quadro é pré-processado. Como o quadro que saiu da janela possui apenas uma classificação, a classe com maior probabilidade é utilizada.

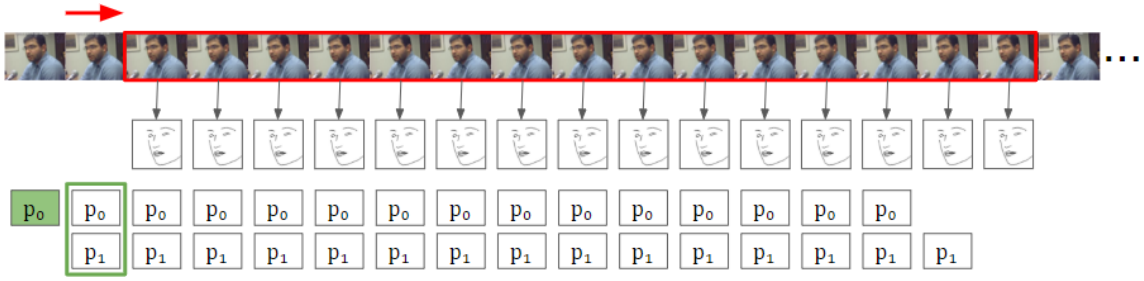
Figura 3.6: Demonstração do funcionamento do algoritmo para as duas primeiras iterações do processo de diarização de uma mídia completa com $step = 1$.



(e) A nova janela é classificada pela rede neural, produzindo um novo conjunto de probabilidades.



(f) As probabilidades retornadas pelo classificador são associadas a cada um dos quadros.



(g) A janela desliza outro quadro para a direita, e o próximo quadro é pré-processado. Como o quadro que saiu da janela possui mais de uma classificação, suas probabilidades são consolidadas segundo o algoritmo selecionado.

Figura 3.6: Demonstração do funcionamento do algoritmo para as duas primeiras iterações do processo de diarização de uma mídia completa com $step = 1$.

Capítulo 4

Resultados

Neste capítulo discutiremos os resultados obtidos nos testes realizados com o sistema diarizador. Primeiramente, na seção 4.1, discutiremos os resultados referentes ao modelo classificador treinado como componente preditora do sistema. Depois, na seção 4.2, apresentaremos os resultados referentes à aplicação diarizadora como um todo.

4.1 Modelo Classificador

Nesta seção avaliaremos o desempenho do modelo treinado em relação à sua capacidade de classificar corretamente segmentos de vídeo de 15 quadros. Para essa finalidade, utilizamos os segmentos de 2 vídeos distintos do dataset que não foram utilizados anteriormente durante o treinamento do modelo, totalizando 990 segmentos válidos com duração de meio segundo e classificados manualmente. Além disso, utilizamos também versões destes espelhadas horizontalmente, totalizando 1980 amostras totais para teste do modelo.

Na seção 4.1.1 apresentamos as métricas gerais de desempenho do modelo, e na seção 4.1.2 apresentamos as métricas de confusão, referentes ao desempenho deste em relação a cada uma das classes alvo. Todas as métricas serão apresentadas para 3 conjuntos de pesos distintos obtidos durante o processo treinamento do modelo. Estes são:

- Os pesos que minimizam o valor da custo utilizada;
- Os pesos que maximizam o valor da acurácia do modelo;
- Os pesos obtidos ao final de todas as épocas do treinamento.

4.1.1 Métricas Gerais

As métricas gerais calculadas para o modelo incluem a sua acurácia, que representa a capacidade do modelo de classificar um dado segmento de video corretamente, a entropia categórica (equação 3.1), que corresponde à certeza do modelo quanto a suas predições, e a área abaixo da curva ROC. A figura 4.1 mostra a evolução dessas métricas ao longo do processo de treinamento do modelo, enquanto a tabela 4.1 destaca os valores destas métricas para os conjuntos de pesos obtidos ao final do treinamento.

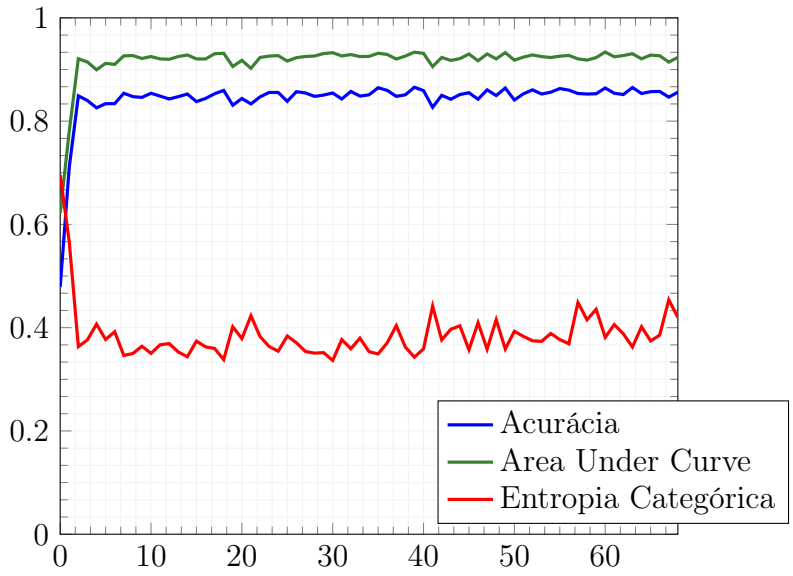


Figura 4.1: Evolução das métricas gerais sobre o conjunto de validação durante o treinamento do modelo. A linha azul representa a evolução da acurácia, enquanto a linha verde corresponde à evolução da área abaixo da curva ROC, e a linha vermelha à da entropia categórica cruzada.

Pesos	Época	Acurácia	AUC	Entropia Categórica
Custo Mínimo	18	0.8596	0.9314	0.3382
Acurácia Máxima	39	0.8656	0.9336	0.3429
Final	68	0.8561	0.9236	0.4198

Tabela 4.1: Métricas referentes aos conjuntos de pesos obtidos ao final do processo de treinamento.

4.1.2 Métricas de Confusão

As métricas de confusão demonstram a capacidade do modelo de classificar corretamente os itens pertencentes a cada uma das classes. Essas incluem a matriz de confusão, que visualiza as classificações feitas pelo diarizador em relação à classe

real de cada item, revelando as tendências deste, e os valores de precisão, *recall*, e F_1 *Score* calculados a partir desta.

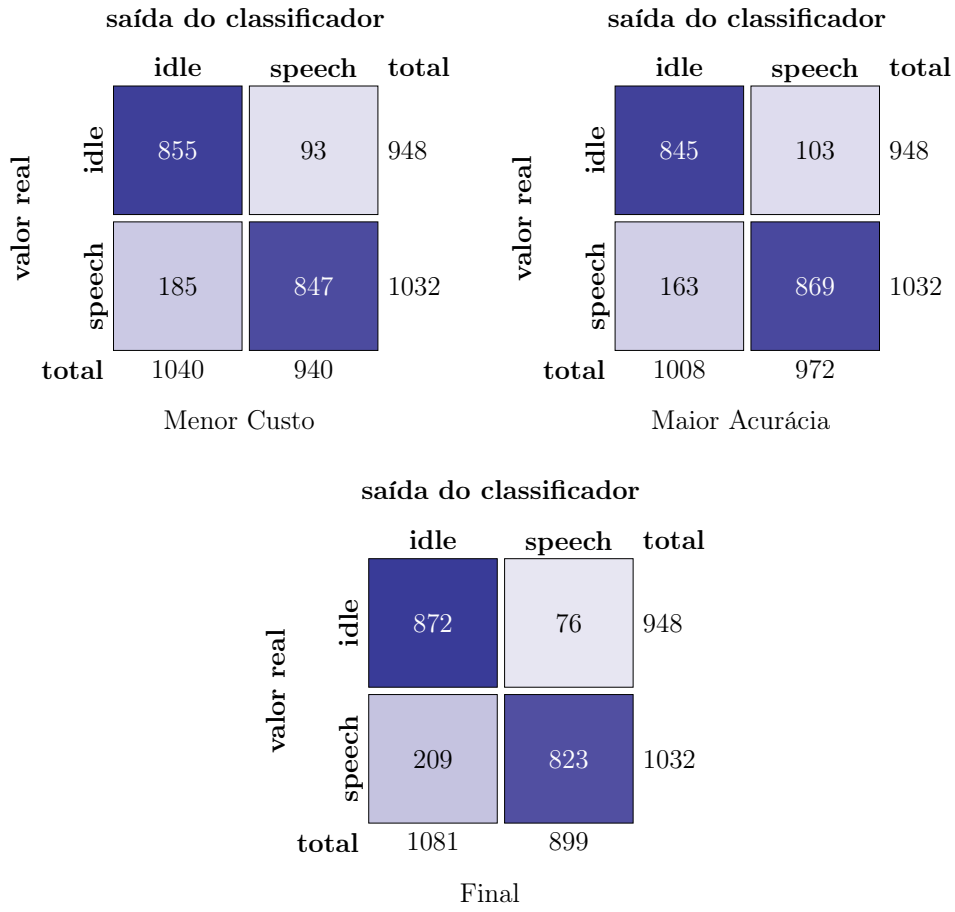


Figura 4.2: Matrizes de Confusão para os conjuntos de pesos obtidos ao final do treinamento do modelo.

A figura 4.2 mostra as matrizes de confusão dos conjuntos de pesos treinados que obtiveram melhor desempenho sobre o conjunto de treinamento. Observa-se a acurácia do modelo no percentual de valores que se encontram da diagonal principal desta; Esses valores correspondem às classificações corretas dos elementos de cada classe. Note, ainda, que os conjuntos de pesos de menor custo e finais do treinamento tendem à esquerda, indicando uma tendência do modelo a classificar negativamente os segmentos observados.

Além disso, para melhor entendimento das propriedades do modelo treinado, calculamos a precisão e o *recall* correspondente a cada classe, assim como o F_1 *Score* para os conjuntos de pesos obtidos ao final do processo de treinamento do modelo. A precisão, calculada a partir da equação 4.1, representa a probabilidade de que um elemento classificado como pertencente a uma classe pertença realmente à mesma, enquanto a taxa de *recall*, calculada a partir da equação 4.2, representa a capacidade do modelo de classificar corretamente os elementos pertencentes a uma determinada classe. Por fim, o F_1 *Score*, calculado através da equação 4.3, corresponde à média

harmônica dessas duas métricas, representando a acurácia do modelo na classificação de uma determinada classe.

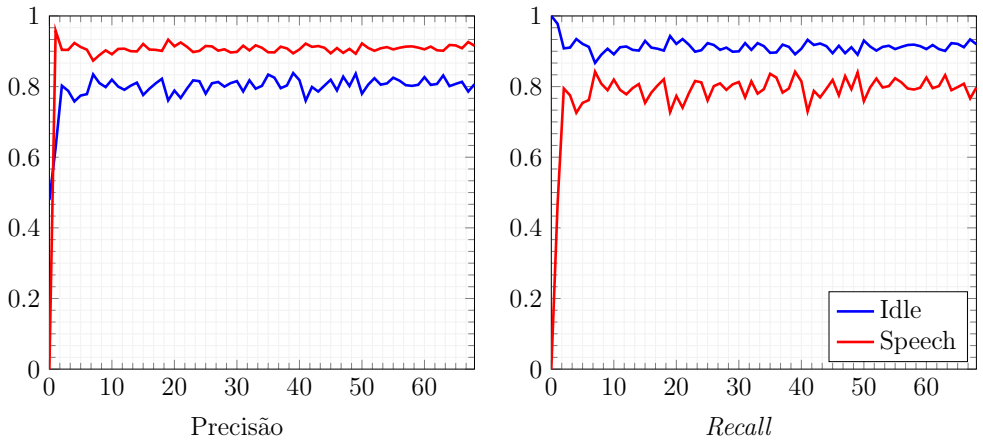


Figura 4.3: Evolução da Precisão e *Recall* de cada classe sobre o conjunto de validação durante o processo de treinamento do modelo. A linha azul representa a classe *Idle*, que corresponde à ausência de fala no fragmento, enquanto a linha vermelha representa a classe *Speech*, que corresponde à detecção de fala no fragmento.

$$Precisão = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$F_1 = 2 \times \frac{precisão \times recall}{precisão + recall} \tag{4.3}$$

	Idle			Speech		
Pesos	Precisão	*Recall*	F_1	Precisão	*Recall*	F_1
Custo Mínimo	0.8221	0.9019	0.8601	0.9011	0.8207	0.8590
Acurácia Máxima	0.8383	0.8940	0.8652	0.8914	0.8421	0.8660
Final	0.8067	0.9155	0.8577	0.9198	0.7975	0.8543

Tabela 4.2: Métricas de confusão para os conjuntos de pesos obtidos ao final de processo de treinamento do modelo.

A figura 4.3 demonstra a evolução das métricas precisão e *recall* calculadas a partir da matriz de confusão ao longo do processo de treinamento para cada classe, enquanto a tabela 4.2 explicita o seu valor para os conjuntos de pesos obtidos ao final do treinamento.

4.2 Aplicação Diarizadora

Nesta seção avaliaremos o desempenho da aplicação diarizadora. Para isso, utilizaremos como métricas de comparação a Taxa de Erro de Diarização (DER, do inglês *Diarization Error Rate*) e a Taxa de Erro de Jaccard (JER, do inglês *Jaccard Error Rate*).

$$DER = \frac{False\ Alarm + Miss + Overlap + Confusion}{Time} \tag{4.4}$$

$$JER_{Speaker} = \frac{False\ Alarm + Miss}{Speech_{Ref} + Speech_{Sys}} \tag{4.5}$$

A Taxa de Erro de Diarização, definida na equação 4.4, consiste na fração do tempo total da diarização que não é atribuída corretamente (*False Alarm*), não é identificada (*Miss*), possui sobreposição de locutores (*Overlap*), ou é atribuída ao locutor errado (*Confusion*). A métrica foi estabelecida pelo *National Institute of Standards and Technology* norte-americano, NIST, e se encontra definida formalmente em [48] e [49], e é desde então a mais utilizada para medição de desempenho em sistemas de diarização.

Já a Taxa de Erro de Jaccard, definida na equação 4.5, consiste no número de segmentos atribuídos incorretamente na diarização produzida pelo sistema, em relação à diarização de referência (*False Alarm*), e vice-versa (*Miss*). Sua definição formal foi feita no artigo introdutório do desafio DIHARD II, e pode ser encontrada em [50].

Para o ajuste do tamanho do passo e algoritmo de consolidação da aplicação, discutidos na seção 3.4.1, assim como a avaliação da aplicação diarizadora como um todo, executamos a mesma sobre um dos vídeos completos para diversas combinações de parâmetros e para cada um dos modelos treinados. Além disso, para o cálculo da DER, consideramos um tamanho de passo de 0.033 segundo, correspondente a um quadro da mídia. Por fim, devido à dificuldade de diarizar as bordas de um segmento de fala de maneira consistente, é utilizado um colarinho de 0.1 segundo, que corresponde a uma tolerância aplicada às bordas dentro da qual um segmento detectado pelo sistema seja considerado condizente com a diarização de referência.

Os resultados da DER e JER para o vídeo diarizado em função dos parâmetros do diarizador podem observados, respectivamente, nas tabelas 4.3 e 4.4. Observe que o melhor algoritmo de consolidação das classificações foi, em todos os casos, o de média das confianças. Já no tamanho do passo, obtivemos melhores resultados quando este era de 3 quadros para modelos com maior confiança (representados pelo valor mínimo da função custo) ou 1 quadro para o modelo com acurácia máxima. Por fim, ambos os modelos obtido ao final do treinamento e de valor mínimo da função

custo obtiveram o mesmo desempenho quando quantificado pela DER, enquanto o modelo final obteve desempenho ligeiramente melhor (aprox. 1.2%) quando avaliado pela JER.

Modelo	Passo	Valor Central	Média	Frequência	Média Gauss.	Freq. Gauss.
Custo Mínimo	1	36.08	32.80	32.90	34.78	34.65
	3	36.31	**32.50**	32.90	35.20	34.99
	5	37.09	35.07	35.23	37.09	37.09
	15	40.47	40.47	40.47	40.47	40.47
Acurácia Máxima	1	34.86	**32.52**	32.56	33.78	33.64
	3	35.77	34.15	34.35	34.96	34.96
	5	36.39	36.56	36.56	36.72	36.39
	15	39.69	39.69	39.69	39.69	39.69
Final	1	35.56	32.76	32.79	33.64	33.53
	3	35.67	**32.50**	32.60	34.02	33.81
	5	36.08	35.00	35.00	36.05	36.08
	15	40.61	40.61	40.61	40.61	40.61

Tabela 4.3: Taxa de Erro de Diarização (DER) para a diarização em função dos parâmetros do diarizador.

Modelo	Passo	Valor Central	Média	Frequência	Média Gauss.	Freq. Gauss.
Custo Mínimo	1	37.08	34.90	34.98	36.35	36.23
	3	37.18	**34.63**	34.97	36.51	36.36
	5	37.91	36.78	36.91	37.91	37.91
	15	40.90	40.90	40.90	40.90	40.90
Acurácia Máxima	1	36.50	**34.93**	34.96	35.67	35.56
	3	37.09	36.30	36.46	36.77	36.77
	5	37.43	38.29	38.29	37.73	37.43
	15	40.22	40.22	40.22	40.22	40.22
Final	1	36.37	34.56	34.59	34.84	34.75
	3	36.46	**34.21**	34.29	35.25	35.09
	5	36.58	35.99	35.99	36.49	36.58
	15	39.46	39.46	39.46	39.46	39.46

Tabela 4.4: Taxa de Erro de Jaccard (JER) para a diarização em função dos parâmetros do diarizador.

A tabela 4.5 demonstra os resultados obtidos por nosso sistema em relação a outros algoritmos publicados na literatura.

[1]Dataset de Depoimentos da Defensoria Pública do Estado do Rio de Janeiro.

[2]O algoritmo é capaz de inferir o número de locutores dinamicamente, mas foi fixado em 2 para essa avaliação.

[3]O número de locutores de cada gravação precisa ser fornecido previamente ao algoritmo.

Algoritmo	Dataset	Tipo da Mídia	Locutores	Taxa de Erro%
Sistema Proposto	D3PERJ[1]	Vídeo	1	32.50 (DER)
LSTM [10] (Naive i-Vectors)	CALLHOME	Áudio	Dinâmico[2]	32.36 (DER)
Naive i-Vector Clustering [51]	DIHARD	Áudio	Predefinido[3]	30.38 (DER)
State-of-the-Art i-Vector Clustering [51]	DIHARD	Áudio	Dinâmico	23.99 (DER)
LSTM [10] (Naive X-Vectors)	CALLHOME	Áudio	Dinâmico[2]	18.87 (DER)
LSTM [10] (Spectral X-Vectors)	CALLHOME	Áudio	Dinâmico[2]	12.48 (DER)
Multi-stream LSTM [14]	AVSpeech	Misto	1	16.00 (SDR*)

Tabela 4.5: Comparação da taxa de erro obtida com as apresentadas em outros trabalhos da literatura, em ordem decrescente. Note que a o algoritmo *Multi-stream LSTM* se encontra fora de ordem devido à incompatibilidade entre as métricas utilizadas.

O sistema apresentou desempenho semelhante ao de implementações tradicionais de diarizadores baseados em i-Vectors, mas consideravelmente pior do que aquele obtido através de técnicas mais modernas, tais como o uso de X-Vectors, extraídos por meio de uma rede neural, ou a clusterização espectral. Porém, a não uniformidade dos datasets utilizados para teste dos sitemas nos impede de fazer comparações mais relevantes.

Gostaríamos, ainda, de chamar atenção ao algoritmo de diarização de mídia mista proposto por proposto por Ephrat et al. em [14]. Este sistema foi avaliado utilizando uma métrica não usual, a SDR, do Inglês *Signal-to-Distortion Ratio*, e utilizando apenas com framentos de 3 segundos gerados a partir do dataset AVSpeech, enquanto os demais consideram diarização total de suas mídias. Dadas estas particularidades deste, embora incluso neste trabalho, hesitamos na utilização de suas métricas para efeito de comparação.

Capítulo 5

Conclusão

Neste trabalho, produzimos um sistema capaz de diarizar um vídeo com um único locutor, com resultado comparável aos obtidos através do uso de i-Vectors, tradicional para este propósito quando apenas o áudio da gravação é considerado. O sistema apresentou desempenho consideravelmente pior do que o de técnicas de processamento de áudio mais modernas. No entanto, devido à sua independência do áudio, esse pode ser utilizado quando o mesmo estiver ausente ou em baixa qualidade para a finalidade proposta.

Ainda, a arquitetura desenvolvida para o modelo de rede neural se demonstrou sólida, frequentemente tendendo ao *overfitting*, o que demonstra sua aptidão para o problema em questão. Dada a introdução de um maior volume de dados para o treinamento da rede, esta poderia vir a apresentar maior acurácia.

5.1 Trabalhos Futuros

Um tópico recorrente no desenvolvimento deste trabalho foi a desconsideração do áudio associado ao vídeo sendo processado. Julgamos que seria possível melhorar consideravelmente o desempenho da rede considerando também os níveis de áudio associados às ações que estão sendo classificadas, tal que um movimento do locutor possa ser classificado também em função do efeito que produz sobre a onda de áudio.

Alternativamente, com algumas adaptações do modelo para obter maior precisão sobre a classe positiva, em detrimento do *recall* sobre esta classe e da acurácia geral deste, o sistema poderia ser utilizado para isolar regiões da mídia nas quais o locutor definitivamente falou, dado que poderia ser utilizado para obter um tipo de perfil de sua voz, que possa agir como um centro definitivo para a clusterização tradicional.

Adicionalmente, para obter melhor acurácia, seria possível adaptar o modelo para considerar métricas calculadas sobre os quadros, tais como o fluxo ótico, métrica representativa da variação, e, consequentemente, do movimento, entre dois quadros consecutivos. Além disso, a identificação facial pode ser melhorada para

permitir reconhecimento de locutores em perfil, visto que a técnica utilizada permite somente a identificação facial frontal, limitação que não se aplica à detecção de marcadores faciais. Ainda, o sistema poderia ser adaptado para utilizar uma rede neural recorrente no processamento de cada quadro individualmente, potencialmente melhorando ainda mais seu desempenho.

Finalmente, o sistema pode ser adaptado para lidar com múltiplos locutores no vídeo, utilizando reconhecimento facial e a localização espacial de cada locutor para identificação destes quadro-a-quadro.

Referências Bibliográficas

[1] 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

[2] 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

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

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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

[23] 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.

[24] 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.

[25] S. Behnke, *Hierarchical Neural Networks for Image Interpretation*, Jan. 2003, vol. 2766.

[26] 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.

[27] 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.

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

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

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

[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.

[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.

[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] 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/

[40] 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.

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

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

[43] 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.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[45] N. Ryant, "Nryant/dscore," May 2020. [Online]. Available: https://github.com/nryant/dscore

[46] 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.

[47] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Apr. 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[48] NIST, "The 2009 (RT-09) Rich Transcription Meeting Recognition Evaluation Plan," 2009. [Online]. Available: https://web.archive.org/web/20170119114252if_/http://www.itl.nist.gov/iad/mig/tests/rt/2009/docs/rt09-meeting-eval-plan-v2.pdf

[49] J. G. Fiscus, J. Ajot, M. Michel, and J. S. Garofolo, "The Rich Transcription 2006 Spring Meeting Recognition Evaluation," in *Machine Learning for Multimodal Interaction*. Springer, Berlin, Heidelberg, May 2006, pp. 309–322.

[50] N. Ryant, K. Church, C. Cieri, A. Cristia, J. Du, S. Ganapathy, and M. Liberman, "The Second DIHARD Diarization Challenge: Dataset, task, and baselines," *arXiv:1906.07839 [cs, eess]*, Jun. 2019. [Online]. Available: http://arxiv.org/abs/1906.07839

[51] D. Dimitriadis, "Enhancements for Audio-only Diarization Systems," *arXiv:1909.00082 [cs, eess]*, Aug. 2019. [Online]. Available: http://arxiv.org/abs/1909.00082