

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO**

DANILO NUNES SOARES

**IDENTIFICADOR DE SENTIMENTO EM VOZ USANDO UM
ARRANJO DE MÚLTIPLOS CLASSIFICADORES**

VITÓRIA
2020

DANILO NUNES SOARES

**IDENTIFICADOR DE SENTIMENTO EM VOZ USANDO UM
ARRANJO DE MÚLTIPLOS CLASSIFICADORES**

Parte manuscrita do Projeto de Graduação do aluno **Daniilo Nunes Soares**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Patrick Marques Ciarelli

Coorientador: MSc. Guilherme Butzke Schreiber Gering

VITÓRIA
2020

DANILO NUNES SOARES

**IDENTIFICADOR DE SENTIMENTO EM VOZ USANDO UM
ARRANJO DE MÚLTIPLOS CLASSIFICADORES**

Parte manuscrita do Projeto de Graduação do aluno **Daniilo Nunes Soares**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 15 de dezembro de 2020.

COMISSÃO EXAMINADORA:

Prof. Dr. Patrick Marques Ciarelli
Universidade Federal do Espírito Santo
Orientador

MSc. Guilherme Butzke Schreiber Gering
Coorientador

Prof. Dr. Evandro Ottoni Teatini Salles
Universidade Federal do Espírito Santo
Examinador

Prof. Dr. Klaus Fabian Côco
Universidade Federal do Espírito Santo
Examinador

AGRADECIMENTOS

Agradeço primeiramente a Deus pela oportunidade de realizar uma graduação de qualidade e por ter me dado forças para percorrer todo o trajeto. Aos meus pais e minha irmã, agradeço por todo o apoio, todo o investimento que fizeram nos meus estudos para que eu pudesse me capacitar e todo o incentivo que recebi. Agradeço à minha noiva por sempre confiar em mim e me incentivar a alcançar meus objetivos, me motivando quando o caminho parecia difícil e incerto.

Aos professores, sou muito grato pelos preciosos ensinamentos e pela paciência para me ensinar e corrigir quando necessário. Ao meu orientador e coorientador, meus sinceros agradecimentos pela disposição a me guiar pelo bom caminho, pelas correções, sugestões e por todo o aprendizado que pude obter. Quanto à UFES, não tenho palavras para agradecer todos os recursos utilizados para meu aprendizado e o empenho de todos os profissionais que constituem a instituição. Agradeço também a IMT Atlantique por me receber de braços abertos para um intercâmbio acadêmico, por me oferecer um ensino de qualidade e contribuir tão ricamente para a minha formação profissional.

Quanto aos meus amigos e colegas de classe, me sinto honrado pelos bons momentos vividos juntos, por compartilharem comigo as vitórias e as dificuldades, bem como por me ajudarem a crescer, tanto como profissional quanto como pessoa.

Reconheço, afinal, que não chegaria nesse momento sem a participação de todos esses em cada passo dessa longa jornada. Cada dia, um novo aprendizado e a cada período uma nova vitória. Por fim, faço menção da célebre frase de Isaac Newton para expressar meu presente sentimento: “Se eu vi mais longe, foi por estar sobre o ombro de gigantes”.

RESUMO

Com a evolução da tecnologia, as pessoas buscam maneiras humanamente naturais de trocar informações com as máquinas. Nesse contexto, deseja-se que a máquina compreenda não só a linguagem humana como também suas emoções. O presente estudo utiliza um arranjo de classificadores para identificação de sentimento em voz, de forma a obter resultados melhores do que o uso de um único classificador individual. Propõe-se uma máquina de vetores de suporte para processar características espectrais, de energia e de frequência extraídas dos áudios analisados, de forma a obter uma saída com a probabilidade de pertinência a cada classe. Em paralelo, uma rede convolucional bidimensional é usada em conjunto com uma rede recorrente para a análise do Mel espectrograma dos sinais de voz, obtendo também uma saída probabilística. Ambos os resultados são agregados por uma máquina de vetores de suporte que compõe o último nível de classificação. Para se aproximar de uma configuração de hiperparâmetros eficaz para cada classificador, emprega-se um algoritmo de otimização bayesiana. As análises são realizadas com os áudios da base de dados Berlin, base essa que possui um significativo desbalanceamento de classes. Para amenizar essa característica, implementa-se um aumento de dados baseado na adição de ruído gaussiano branco e alteração da velocidade do sinal de voz. Obteve-se ao fim dos experimentos um valor médio de 0,86 na métrica de desempenho F1, valor maior do que o obtido no artigo utilizado como referência. Observou-se que a otimização bayesiana é efetiva para se estimar os valores de hiperparâmetros de algoritmos de aprendizado de máquina, enquanto o aumento de dados pouco contribuiu para a melhoria de desempenho do arranjo de classificadores.

Palavras-chave: Identificação de sentimento. Processamento de sinais. Classificação. Redes neurais profundas.

ABSTRACT

Along with the technological development, people look for natural ways to exchange information with machines. In this context, it is desirable that the machine understands not only human language but also their emotions. This research uses an arrange of classifiers for speech emotion recognition aiming to obtain better results than the use of a single classifier. The use of a support vector machine is proposed to process spectral, energy and frequency features from analyzed audios, in order to obtain the belonging probability of the audio with respect to each class. In parallel, a two-dimensional convolutional neural network is used along with a recurrent neural network to analyze the Mel spectrogram of audio signals, resulting in a probabilistic output too. Both results are joined by a support vector machine that composes the last classification level. To estimate an effective hyperparameter configuration of each classifier, a Bayesian optimization algorithm is applied. The analysis are made using audios from the Berlin database, which has a meaningful class unbalance. To soften this characteristic, a data augmentation approach is implemented using white gaussian noise addition and audio time-stretch. In the end of the experiments, a mean value of 0,86 for the F1 performance metric is obtained, a bigger value than the observed in the article used as reference. It was noticed that the Bayesian optimization is effective to estimate the machine learning algorithms hyperparameter values, while the data augmentation techniques brought a small contribution to the performance improvement of the classifier arrangement.

Keywords: Speech emotion recognition. Signal processing. Classification. Deep neural networks.

LISTA DE FIGURAS

Figura 1 – Mapa de afeto de Russell	16
Figura 2 – Modelo bidimensional de emoções para a base Berlin	17
Figura 3 – Taxas de reconhecimento e diferenças significantes entre emoções	21
Figura 4 – Separação de classes com SVM de margens rígidas.....	24
Figura 5 – Uma operação de convolução	28
Figura 6 – Esquema de uma operação de <i>max pooling</i> 2D	28
Figura 7 – Esquema simplificado de uma célula da LSTM	29
Figura 8 – Diagrama interno de uma célula LSTM.....	30
Figura 9 – Esquema explicativo do classificador proposto	37
Figura 10 – Procedimento do <i>k-fold cross validation</i> para $k = 5$	38
Figura 11 – Histograma de uma característica com distribuição assimétrica.....	41
Figura 12 – Histograma da característica normalizada com desvio padrão	41
Figura 13 – Normalização da característica com o desvio padrão da mediana.....	42
Figura 14 – Mel espectrograma de uma janela de áudio	45
Figura 15 – Bloco de aprendizado de características locais (BACL).....	46
Figura 16 – Diagrama de blocos da CNN-LSTM 2D.....	46
Figura 17 – Classificação no segundo nível para um áudio segmentado em 2 janelas	50
Figura 18 – Ajuste global do hiperparâmetro C da SVM linear.....	52
Figura 19 – Ajuste local do hiperparâmetro C da SVM linear	53
Figura 20 – Ajuste global do parâmetro γ da SVM Gaussiana	53
Figura 21 – Ajuste local do parâmetro γ da SVM Gaussiana.....	54
Figura 22 – Ajuste global do parâmetro C da SVM Gaussiana.....	54
Figura 23 – Ajuste local do hiperparâmetro C da SVM Gaussiana.....	55
Figura 24 – Ajuste das variáveis a serem utilizadas na SVM Linear	56
Figura 25 – Ajuste das variáveis a serem utilizadas na SVM Gaussiana	56
Figura 26 – Ajuste do tamanho da janela de áudio.....	57
Figura 27 – Primeiro ajuste do tamanho do passo da janela	57
Figura 28 – Segundo ajuste do tamanho do passo da janela	58
Figura 29 – Ajuste da taxa de aprendizado da CNN-LSTM-2D	59
Figura 30 – Ajuste do parâmetro C da SVM de segundo nível.....	61
Figura 31 – Ajuste do parâmetro γ da SVM de segundo nível.....	62
Figura 32 – Mel espectrograma original de uma janela de áudio.....	64

Figura 33 – Mel espectrograma de uma janela de áudio com adição de ruído.....	64
Figura 34 – Mel espectrograma de uma janela de áudio com diminuição de velocidade	65
Figura 35 – Mel espectrograma de uma janela de áudio com aumento de velocidade	65

LISTA DE TABELAS

Tabela 1 – Distribuição dos áudios por classe.....	22
--	----

LISTA DE QUADROS

Quadro 1 – Exemplo de matriz de confusão.....	35
Quadro 2 – Porcentagem de amostras por classe em cada conjunto da execução 0.....	38
Quadro 3 – Faixa de valores do ajuste de hiperparâmetros da SVM de primeiro nível.....	43
Quadro 4 – Faixa de valores do ajuste do tamanho da janela de áudio.....	44
Quadro 5 – Descrição das camadas da CNN-LSTM 2D.....	47
Quadro 6 – Faixa de valores do ajuste de hiperparâmetros da SVM de segundo nível.....	49
Quadro 7 – Valores escolhidos para os hiperparâmetros.....	55
Quadro 8 – Resultado do ajuste de hiperparâmetros da SVM de primeiro nível.....	58
Quadro 9 – Desempenho da SVM ajustada sobre o conjunto de validação.....	59
Quadro 10 – Resultado da otimização de hiperparâmetros da CNN-LSTM-2D.....	60
Quadro 11 – Desempenho da CNN-LSTM 2D ajustada no conjunto de validação.....	60
Quadro 12 – Resultado do ajuste da SVM do segundo nível.....	62
Quadro 13 – Desempenho do algoritmo completo ajustado nos conjuntos de teste.....	62
Quadro 14 – Desempenho por classe do algoritmo completo.....	63
Quadro 15 – Resultado da aplicação do aumento de dados na SVM de primeiro nível analisada nos folds de teste.....	66
Quadro 16 – Resultado da aplicação do aumento de dados na CNN-LSTM 2D analisada nos folds de teste.....	66
Quadro 17 – Resultado da aplicação do aumento de dados no arranjo de classificadores.....	66
Quadro 18 – Comparação do F1 obtido no modelo proposto com resultados de outros trabalhos.....	67
Quadro 19 – Média e desvio padrão da duração dos áudios de cada classe.....	68

LISTA DE ABREVIATURAS E SIGLAS

1D	Unidimensional
2D	Bidimensional
BACL	Bloco de Aprendizizado de Características Locais
CNN	<i>Convolutional Neural Network</i>
EI	<i>Expected Improvement</i>
LPCMCC	<i>Linear Prediction Coefficients Mel Cepstrum Coefficients</i>
LSTM	<i>Long Short-Term Memory</i>
PLN	Processamento de Linguagem Natural
SER	<i>Speech Emotion Recognition</i>
SNR	<i>Signal-to-Noise Ratio</i>
SVM	<i>Support Vector Machine</i>
TPE	<i>Tree-structured Parzen Estimator</i>
ELU	<i>Exponential Linear Unit</i>

LISTA DE SÍMBOLOS

s	Segundo
T	Conjunto de treinamento
x	Vetor de dados
X	Espaço de dados
y	Rótulo de dado
Y	Espaço de rótulos
H	Fronteira de separação
w	Vetor normal à fronteira de separação
\cdot	Produto escalar
b	Escalar real
d	Projeção na direção de w
ξ	Folga
ϕ	Função de mapeamento
K	<i>Kernel</i>
k	Escalar inteiro
γ	Hiperparâmetro do kernel
$x(t)$	Sinal de entrada
$*$	Operação de convolução
$s(t)$	Sinal de saída
I	Sinal de entrada 2D
m	Tamanho da primeira dimensão de I
n	Tamanho da segunda dimensão de I
i	Índice
j	Índice
E	Vetor de entrada da célula da LSTM
S	Vetor com agrupamento de todas as saídas das células da LSTM
O	Vetor com agrupamento de todos os estados ocultos das células da LSTM
t	Instante de tempo
f	Mapeamento do algoritmo de aprendizado de máquina
θ	Vetor de parâmetros
$g^{(t)}$	Valor da porta de entrada da LSTM

$q^{(t)}$	Valor da porta de saída da LSTM
$s^{(t)}$	Vetor estado da célula
$h^{(t)}$	Vetor de saída de todas as células
$U^{(t)}$	Peso para os dados de entrada anteriores
$V^{(t)}$	Peso para os dados de saída anteriores
$\sigma(t)$	Função sigmoide
α	Vetor de hiperparâmetros
u	Medida do desempenho do algoritmo de aprendizado de máquina
u^*	Menor valor de custo já encontrado para o algoritmo de aprendizado de máquina
$l(\alpha)$	Densidade de probabilidade formada usando as observações que resultam em um valor de u maior que u^*
$g(\alpha)$	Densidade de probabilidade formada com as observações complementares a $l(\alpha)$
q	Quantil
VP	Verdadeiros positivos
FP	Falsos positivos
FN	Falsos negativos
VN	Verdadeiros negativos
C	Parâmetro de regularização da SVM
F_{mel}	Frequência na escala Mel
F_{Hz}	Frequência em Hertz
$\sigma(x)$	Função <i>Softmax</i>
C	Conjunto de classes
M	Classe majoritária
j_M	Quantidade de janelas da classe majoritária
j_i	Quantidade de janelas da classe i
B	Valor base de janelas a serem geradas
n_i	Valor intermediário da classe i
N	Quantidade de janelas a serem geradas
A_r	Amplitude do ruído a ser inserido
A_m	Amplitude média dos sinais de uma classe
S	Conjunto de amostras de áudio base escolhidas
k	Número de divisões da validação cruzada

SUMÁRIO

1	INTRODUÇÃO	15
1.1	Apresentação e Objeto de Pesquisa	15
1.2	Justificativa	18
1.3	Objetivos	19
1.3.1	Objetivo Geral.....	19
1.3.2	Objetivos Específicos.....	19
1.4	Estrutura do Trabalho	20
2	BASE DE DADOS	21
3	REFENCIAL TEÓRICO	23
3.1	Algoritmos de Aprendizado de Máquina	23
3.1.1	Máquinas de Vetores de Suporte (SVM)	23
3.1.2	Redes Neurais Convolucionais	26
3.1.3	Redes Neurais Recorrentes	28
3.2	Algoritmo de Otimização de Hiperparâmetros	31
3.3	Algoritmos de Aumento de Dados	34
3.4	Métricas de Desempenho	34
4	METODOLOGIA	36
4.1	Pré-processamento dos Áudios	38
4.2	Extração de Características do Sinal de Voz	39
4.3	Implementação e Ajuste da SVM do Primeiro Nível	42
4.4	Extração do Mel Espectrograma dos Sinais de Voz (Entrada da SVM)	44
4.5	Implementação e Ajuste da CNN-LSTM 2D	45
4.6	Aumento de Dados (<i>Data Augmentation</i>)	47
4.7	Implementação e Ajuste da SVM de Segundo Nível e Classificação do Áudio	49
5	RESULTADOS E DISCUSSÃO	51
5.1	Recursos Computacionais Utilizados	51
5.2	Ajuste de Hiperparâmetros da SVM do Primeiro Nível	51
5.3	Ajuste dos Hiperparâmetros da CNN-LSTM-2D	59
5.4	Ajuste da SVM do Segundo Nível	61

5.5	Aumento de Dados	63
5.6	Comparação de Resultados	66
6	CONCLUSÕES E PROJETOS FUTUROS	69
6.1	Conclusões.....	69
6.2	Sugestões para Projetos Futuros	69
	REFERÊNCIAS BIBLIOGRÁFICAS.....	71

1 INTRODUÇÃO

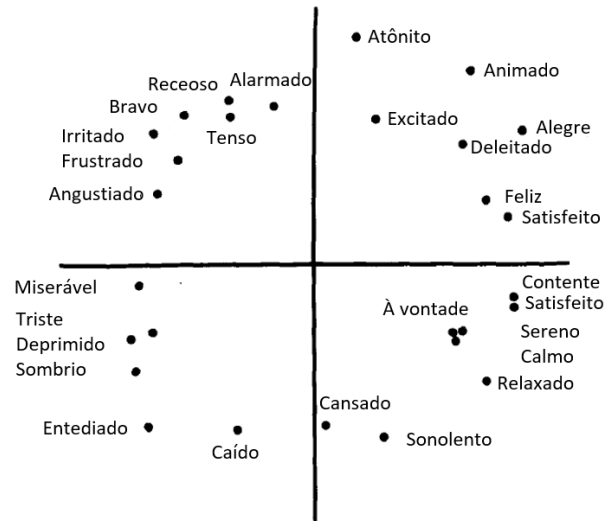
1.1 Apresentação e Objeto de Pesquisa

A voz é um meio através do qual as pessoas se relacionam e trocam informações. As mensagens são transmitidas de duas principais maneiras: a linguagem verbal e a não verbal (PATHAK; KOLHE, 2016). Para compreender as interações com seus semelhantes, as pessoas aprendem desde crianças a interpretar ambas as linguagens e a conectá-las para identificar com clareza a mensagem transmitida. Com os avanços da tecnologia e a popularização da relação homem-máquina, torna-se conveniente que máquinas também sejam capazes de interpretar a comunicação humana e suas nuances, área de estudo da ciência da computação conhecida como processamento de linguagem natural (PLN). Sob essa ótica, Hirschberg e Manning (2015) destacam que ferramentas de PLN para analisar a parte verbal da comunicação já existem na forma de produtos bem difundidos no mercado, como a Siri da *Apple* e o *Google Tradutor*. Por outro lado, a identificação do sentimento do locutor ganha destaque ao se mostrar uma parte complexa e relevante da análise não-verbal da mensagem, dado que uma mesma palavra dita com entonações diferentes pode trazer diferentes significados (PATHAK; KOLHE, 2016).

É importante notar, todavia, que o sentimento de uma pessoa não é naturalmente bem definido, visto que as emoções são normalmente expressas em conjunto. Heider (1920, p. 175) afirma que cada sentimento tende a incluir em seu sistema todas as emoções, pensamentos, processos volitivos e qualidades de caráter que são vantajosos para o locutor realizar seus fins. Além disso, a forma como uma dada emoção é expressa depende, de forma geral, da cultura e do ambiente do locutor (AYADI; KAMEL; KARRAY, 2011). Ainda assim, torna-se necessária a distinção de sentimentos através de um modelo simplificado para viabilizar o aprendizado de máquina. Uma modelagem popular no ramo da PLN é a de Russell (1980), que utiliza um modelo bidimensional para caracterizar as emoções, conforme a Figura 1. Nessa, o eixo horizontal caracteriza a valência, que mede o nível de prazer do sentimento, de forma que a valência negativa indica descontentamento e a valência positiva indica prazer. O eixo vertical indica a excitação (ou intensidade) do sentimento, numa escala que varia entre baixa e alta excitação. Nesse contexto, nota-se que os sentimentos possuem caráter contínuo, podendo assumir qualquer valor em ambos os eixos. Ainda na Figura 1, Russell (1980) propõe diversas classes

discretas que representam uma gama de emoções e seus respectivos termos utilizados pelos humanos.

Figura 1 – Mapa de afeto de Russell



Fonte: Russell (1980).

Nota: Adaptado pelo autor.

Em posse de um conjunto de classes de sentimentos bem definido, procuram-se alternativas para ensinar a máquina como realizar essa tarefa de classificação. Na área da PLN, diversos métodos de aprendizado de máquina adquiriram grande popularidade devido ao aumento do poder computacional observado nas últimas décadas (HIRSCHBERG; MANNING, 2015). Dentre eles, os mais comuns são: máquinas de vetores de suporte (SVM, do inglês *support vector machines*), como se observa nos trabalhos de Eray, Tokat e Iplikci (2018) e Shen, Changjun e Chen (2011); e redes neurais convolucionais (CNN, do inglês *convolutional neural network*) de uma e duas dimensões (1D e 2D), como descrito nos artigos de Zhao, Mao e Chen (2019) e Badshah e outros (2017). Essas metodologias possuem acurácias comparáveis e, dependendo do caso específico, uma pode se mostrar mais eficiente do que as outras.

Nesse contexto, Gering, Ciarelli e Salles (2019) propuseram a combinação de diversos algoritmos de aprendizado de máquina (incluindo os supracitados) para tratar o reconhecimento de sentimento em voz (SER, do inglês *speech emotion recognition*), com o entendimento de que cada classificador pudesse contribuir com alguma informação relevante para a decisão final. Dada a pluralidade de métodos utilizados, a complexidade e a atualidade da tarefa, optou-se no presente projeto por dar continuidade a esse trabalho, principalmente no que diz respeito

às sugestões feitas para projetos futuros. Convém destacar que, doravante, esse trabalho será chamado de artigo base. Em sua dissertação, Gering (2019) traz mais detalhes a respeito desse estudo. Os autores utilizaram a base de dados Berlin (BURKHARDT et al., 2005) em suas avaliações. Ela é de domínio público e composta por áudios que representam 7 classes de emoções (alegria, desgosto, medo, raiva, tédio e tristeza), cujas posições foram registradas no mapa de afeto de Russell da Figura 2, juntamente com os nomes dados aos quadrantes.

Figura 2 – Modelo bidimensional de emoções para a base Berlin



Fonte: Gering, Ciarelli e Salles (2019).

Zhao, Mao e Chen (2019) também realizaram um relevante trabalho utilizando a base de dados Berlin para seus estudos. Duas redes neurais foram implementadas: uma contendo camadas convolucionais 1D para extrair informações do sinal de voz e outra que utilizava camadas convolucionais 2D para extrair características dos log-mel espectrogramas dos sinais de voz. Ambas possuíam uma camada recorrente no final da rede neural para realizar a classificação dos sentimentos. Nos resultados apresentados pelos autores, a maior acurácia de classificação foi obtida com a utilização dos log-mel espectrogramas nas redes neurais convolucionais 2D.

Apesar dos bons resultados apresentados em ambos os trabalhos supracitados, sabe-se que uma característica marcante da Berlin é o número não uniforme de gravações de voz por classe, o que é chamado de base de dados desbalanceada. Mountassir, Benbrahim e Berrada (2012) afirmam que essa característica é muito comum em situações práticas, de forma que a equalização da quantidade de amostras por classe se torna imprescindível para obter melhores

desempenhos no aprendizado de máquina. Para esse fim, o aumento de dados (mais popular na forma em inglês, *data augmentation*) tem sido muito utilizado por pesquisadores da área (ZHU et al., 2018). No contexto da análise de fala, verifica-se ser benéfica a transformação dos sinais existentes, comumente na forma de adição de ruído ou alteração da frequência do sinal de voz (FUKUDA et al., 2018). Em posse dessas informações, propõe-se, nesse trabalho, a aplicação de técnicas de aumento de dados para sinais de áudio com o intuito de amenizar o impacto da quantidade desbalanceada de amostras por classe de sentimento.

Dentre as arquiteturas de aprendizado de máquina implementadas no artigo base, encontram-se as redes neurais convolucionais 1D e 2D, bem como uma máquina de vetores de suporte e uma combinação das suas saídas por meio de uma árvore de decisão. No presente estudo, decidiu-se por concentrar o foco dos estudos na máquina de vetor de suporte e na rede convolucional bidimensional, sabendo que essas obtiveram os melhores desempenhos conforme o relato presente no artigo base. Sabe-se também que é de suma importância assegurar que os hiperparâmetros de cada arquitetura sejam os mais adequados, processo conhecido como otimização de hiperparâmetros. Essa etapa é fundamental na construção de um modelo de aprendizado de máquina, pois pode trazer um grande impacto positivo no desempenho obtido (ZHAO; MAO; CHEN, 2019). Assim, o presente estudo também engloba uma etapa de otimização de hiperparâmetros com o intuito de melhorar o desempenho dos modelos implementados.

1.2 Justificativa

Diversas situações podem exigir o conhecimento do estado emocional do locutor para a correta interpretação da fala. No ensino à distância, por exemplo, conhecer a emoção do aluno pode auxiliar na percepção das dificuldades do estudante e contribuir para a melhoria do método de ensino (SHEN; CHANGJUN; CHEN, 2011). Outra significativa aplicação é a identificação do estresse mental de motoristas (PATHAK; KOLHE, 2016), uma vez que sentimentos de baixa excitação podem indicar cansaço e um alerta pode ser emitido para que o motorista dê uma pausa em seu trajeto.

Também se encontram aplicações em centrais de atendimento. Nesse contexto, pode-se utilizar a informação adquirida para identificar o grau de satisfação do cliente ou repassar os mais

insatisfeitos para atendentes mais experientes (XIA; LIU, 2015). Pode-se salientar também que isso pode vir a eliminar a necessidade de questionários após o término do atendimento, o que aumenta a quantidade de *feedbacks* recebidos e melhora a satisfação do cliente quanto ao processo de atendimento.

No ramo da medicina, uma importante aplicação é a identificação de autismo. Sabe-se que pacientes nesse quadro demonstram pouca excitação na maior parte de suas falas e, por isso, o monitoramento das demonstrações de emoções pode ser relevante. Cabe, porém, ressaltar que alguns autistas não conseguem falar, o que inviabiliza o uso dos métodos propostos no monitoramento desses indivíduos. Além disso, é possível o acompanhamento de pacientes com estresse ou depressão e até mesmo monitorar o avanço de um aconselhamento psicológico (REDDY; VIJAYARAJAN, 2017).

Levando em conta essas informações, espera-se que o presente estudo possa contribuir para o avanço da SER e até mesmo do aprendizado de máquina em geral, de maneira a proporcionar melhores ferramentas de comunicação entre humanos e máquinas e facilitar a criação de dispositivos que tragam mais segurança e bem-estar para as pessoas.

1.3 Objetivos

1.3.1 Objetivo Geral

Este projeto tem como objetivo utilizar um arranjo de classificadores para identificação de sentimento em voz, baseando-se, para isso, nas sugestões de trabalhos futuros apresentados por Gering, Ciarelli e Salles (2019), que foram a utilização de técnicas de aumento de dados para equalização da base de dados e na aplicação da otimização de hiperparâmetros para encontrar uma configuração eficaz dos métodos propostos.

1.3.2 Objetivos Específicos

Para alcançar o objetivo geral, definem-se os objetivos específicos, os quais auxiliam a acompanhar o desenvolvimento do projeto. São eles:

- Analisar a contribuição nos resultados ao se utilizar dois níveis de classificadores;

- Verificar o efeito da aplicação de técnicas de aumento de dados para equalização de base de dados;
- Identificar possível melhoria de desempenho de classificação resultante da otimização de hiperparâmetros dos classificadores.

1.4 Estrutura do Trabalho

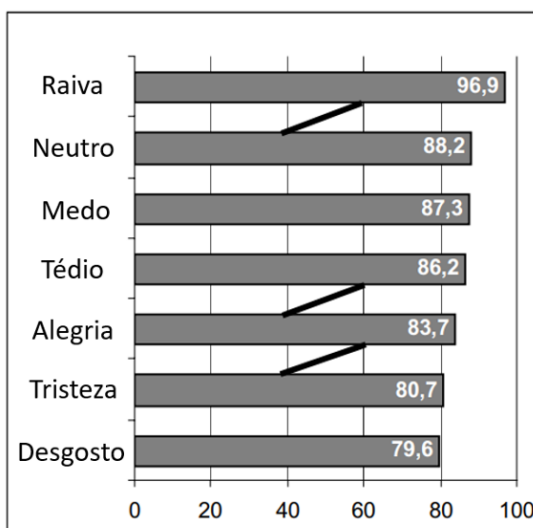
Este trabalho está organizado da seguinte forma: a seção 2 aborda sobre a base de dados utilizada, de forma a identificar suas principais características. A seção 3 detalha os algoritmos de aprendizado de máquina, aumento de dados e otimização de hiperparâmetros utilizados, de forma a apresentar uma breve explicação da teoria e dos casos de aplicação de cada um. A seção 4 descreve a metodologia utilizada. Os experimentos e resultados são apresentados na seção 5 e, por fim, a seção 6 traz as conclusões obtidas, bem como sugestões para trabalhos futuros.

2 BASE DE DADOS

A base de dados utilizada foi a Berlin (BURKHARDT et al., 2005). Disponibilizada ao público de forma aberta, ela é amplamente utilizada em estudos científicos, o que garante que haverá significativas referências de desempenho para comparar com o método proposto no atual projeto. Conforme indicado na Figura 2, ela é composta por sete classes de sentimentos: alegria, desgosto, medo, neutro, raiva, tédio e tristeza.

Para a criação da Berlin, foram escolhidos dez atores (5 homens e 5 mulheres) que pronunciaram 10 frases em alemão (5 curtas e 5 longas). As gravações foram realizadas numa câmara anecóica e com a taxa de amostragem de 48 kHz, sendo depois subamostradas para 16 kHz. Cabe destacar, nesse cenário, que as emoções foram simuladas, dada a dificuldade de se obter as emoções desejadas em situações da vida real e com as mesmas condições de qualidade de gravação (BURKHARDT et al., 2005). Além disso, para garantir que as emoções estavam bem representadas, os autores realizaram um teste de percepção que consistiu em seleccionar 20 pessoas para escutarem os áudios uma vez e classificarem as emoções. O resultado está representado na Figura 3.

Figura 3 – Taxas de reconhecimento e diferenças significantes entre emoções



Fonte: Burkhardt e outros (2005).
Nota: Adaptado pelo autor.

Fica clara a diferença significativa na taxa de reconhecimento das diferentes classes, o que pode impactar no desempenho dos modelos desenvolvidos. Outro ponto muito relevante é que apenas os áudios que obtiveram mais de 80% de taxa de reconhecimento (ou seja, aquelas que foram corretamente classificadas por pelo menos 16 pessoas) no teste de percepção foram mantidos na base. Esse processo fez com que, das 800 gravações inicialmente feitas, apenas 535 estivessem presentes no conjunto final. A média e desvio padrão da duração desses áudios são de 2,78 s e 1,03 s, respectivamente, sendo que o maior áudio possui 8,97 s de duração. A distribuição dos áudios por classe se encontra na Tabela 1.

Tabela 1 – Distribuição dos áudios por classe

Sentimento	Nº de áudios	% do total
Raiva	127	23,7
Tédio	81	15,1
Neutro	79	14,8
Alegria	71	13,3
Medo	69	12,9
Tristeza	62	11,6
Desgosto	46	8,6

Fonte: Burkhardt e outros (2005).

Nota: Adaptado pelo autor.

Observa-se que a quantidade de áudios por classe é acentuadamente desbalanceada. Vanucci e Colla (2017) destacam que esse desbalanceamento da base de dados torna a tarefa de classificação complexa, pois os algoritmos de aprendizado de máquina partem do pressuposto de que o número de exemplos de cada classe é uniforme. Se esse não for o caso, vê-se resultados enviesados para as classes majoritárias e, como consequência, as classes minoritárias são classificadas incorretamente.

Para lidar com essa situação, foram implementados os métodos de otimização de hiperparâmetros e balanceamento de base de dados descritos na seção 3.

3 REFERENCIAL TEÓRICO

3.1 Algoritmos de Aprendizado de Máquina

3.1.1 Máquinas de Vetores de Suporte (SVM)

As SVM podem ser utilizadas para distinguir dados provenientes de duas classes diferentes. Conforme destacam Shen, Changjun e Chen (2011), esse algoritmo consiste, de forma mais abrangente, nas seguintes etapas: primeiro, utiliza-se uma função não-linear (conhecida como núcleo) para mapear dados nas condições originais para um espaço de alta dimensão. Em seguida, traçam-se dois hiperplanos que representem a fronteira dos dados de cada classe. Por fim, otimiza-se a configuração que maximiza a distância entre os hiperplanos anteriores, de forma a obter um hiperplano que separe os dados de diferentes classes com a maior margem possível.

O modelo mais simples desse algoritmo é a SVM linear com margens rígidas. Nele, não é utilizada nenhuma função núcleo (*kernel*, em inglês) e a divisão entre as classes é realizada por meio de hiperplanos no espaço dos dados. Para exemplificar o algoritmo, seja um conjunto T de treinamento com n dados $\mathbf{x}_i \in X$ com rótulos $y_i \in Y$, de forma que X constitua o espaço dos dados e $Y = \{-1, +1\}$. A equação básica de um hiperplano se encontra em (1). Para esse conjunto de dados, as margens de separação são H_1 e H_2 , definidos nas equações (2) e (3), na qual \mathbf{w} é o vetor normal às fronteiras, \cdot representa a operação de produto escalar e b um escalar real (LORENA; CARVALHO, 2007). A Figura 4 contém uma representação gráfica desse exemplo.

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b \quad (1)$$

$$H_1: \mathbf{w} \cdot \mathbf{x} + b = 1 \quad (2)$$

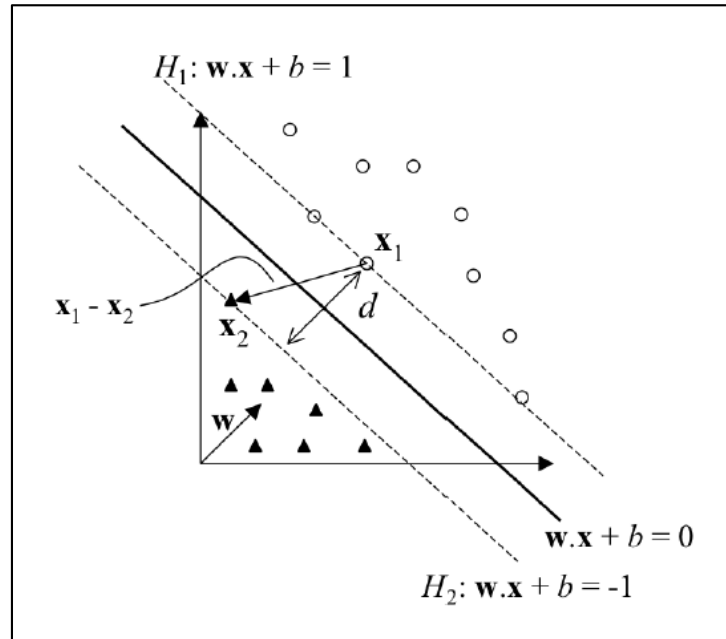
$$H_2: \mathbf{w} \cdot \mathbf{x} + b = -1 \quad (3)$$

Como H_1 e H_2 são as fronteiras de separação, as inequações em (4) descrevem a posição dos dados em relação aos hiperplanos. As inequações presentes em (4) estão resumidas em (5).

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \text{ se } y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \text{ se } y_i = -1 \end{cases} \quad (4)$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall (\mathbf{x}_i, y_i) \in T \quad (5)$$

Figura 4 – Separação de classes com SVM de margens rígidas



Fonte: Lorena e Carvalho (2007).

Considerando pontos \mathbf{x}_1 e \mathbf{x}_2 pertencentes a H_1 e H_2 , respectivamente, pode-se projetar o vetor $\mathbf{x}_1 - \mathbf{x}_2$ na direção do vetor \mathbf{w} que é normal aos hiperplanos. Na Figura 4, essa projeção é chamada de \mathbf{d} e possui comprimento $2/\|\mathbf{w}\|$. Assim, para se maximizar a margem de separação dos dados, deve-se minimizar $\|\mathbf{w}\|$ com a restrição em (5), o que está descrito em (6) (LORENA; CARVALHO, 2017).

$$\begin{aligned} & \text{Minimizar } \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{Com restrições: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall (\mathbf{x}_i, y_i) \in T \end{aligned} \quad (6)$$

Nesse contexto, as restrições não permitem nenhum dado de treino entre H_1 e H_2 , o que é um caso raro na prática devido à natureza dos sinais reais, nos quais se observa ruído, pontos fora da curva ou mesmo dados não linearmente separáveis (LORENA; CARVALHO, 2017). Para suavizar as margens de separação, pode-se introduzir uma variável de folga ξ em (4), de forma a obter a nova restrição que se vê em (7).

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i = 1, \dots, n \quad (7)$$

Esse modelo é conhecido como SVM de margens suaves, que se destaca pela capacidade de aceitar que alguns elementos fiquem entre as margens de separação. Ainda assim, esse algoritmo não é capaz de modelar dados com forte característica não-linear. Uma maneira de contornar essa adversidade, segundo Bishop (2006, p. 326), é utilizar uma função não-linear para mapear os dados para um espaço de dimensão maior e então aplicar a SVM linear nesse espaço de características. A equação (8), modificação de (1), mostra como se define um hiperplano nesse novo espaço, em que ϕ é a função aplicada. Dessa maneira,

$$f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b \quad (8)$$

Computacionalmente falando, esse mapeamento pode ser extremamente custoso, pois o espaço de características pode ter dimensão muito alta. Na prática, sabe-se que a única informação necessária sobre esse mapeamento é como realizar produtos escalares entre dois dados no espaço de características (LORENA; CARVALHO, 2017). Para isso, lança-se mão da função *kernel*. Como ela não realiza o mapeamento direto para o espaço de características, diz-se que ela realiza o mapeamento implícito para esse novo espaço. A equação (9) descreve matematicamente o que foi dito, sendo K o *kernel*, \mathbf{x}_i e \mathbf{x}_j dados no espaço de entradas. Alguns *kernels* presentes no ramo de SER são o gaussiano e o linear, que estão representados nas equações (10) e (11), respectivamente. Eles serão utilizados no presente estudo, conforme se explica na seção 4.3. Ressalta-se que γ é um hiperparâmetro desse *kernel*.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \quad (9)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)} \quad (10)$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j \quad (11)$$

Em posse das ferramentas citadas, possui-se um classificador binário. Ocorre que, na prática, é comum a presença de problemas que envolvem separação de múltiplas classes. Como a SVM é, por definição, um classificador binário, existem duas abordagens que são comumente implementadas para o caso multi-classe: um contra todos e um contra um.

A abordagem um contra todos consiste em criar k SVM distintas, sendo que o modelo $y_k(\mathbf{x})$ é treinado utilizando os dados da k -ésima classe como exemplos positivos e os dados de todas as outras $k - 1$ classes como exemplos negativos. Um dado a ser analisado é testado em todos os modelos e considera-se que sua classe é a que obtiver o maior valor de y_k . Um problema para essa alternativa é que os classificadores foram treinados em tarefas diferentes e não há garantia que os valores de y_k para diferentes classificadores terão escalas apropriadas. Além disso, nessa abordagem, os conjuntos de dados de treino são desbalanceados, já que, por exemplo, para um problema com 10 classes, 90% dos dados de treino seriam exemplos negativos e apenas 10% exemplos positivos (BISHOP, 2006, p. 338).

Por outro lado, a abordagem um contra um requer o treino de $(k^2 - k)/2$ SVM de duas classes para todos os pares possíveis de classes e então classificar os dados de teste de acordo com a classe que possuir maior número de “votos”. O ponto negativo dessa abordagem é que, para grandes valores de k , o tempo de treino se torna significativamente maior, assim como o tempo para realizar testes. Todavia, estudos apontam que essa abordagem tem obtido melhores resultados (CHANG; LIN, 2013) e, por essa razão, ela será utilizada no presente estudo.

3.1.2 Redes Neurais Convolucionais

Redes neurais convolucionais (CNN) são tipos especiais de redes neurais especializadas no processamento de dados com topologia de grade. Alguns exemplos são séries temporais, vistos como uma grade unidimensional (1D) com espaçamento uniforme, e imagens, que são grades bidimensionais (2D) de *pixels* (GOODFELLOW; BENGIO; COURVILLE, 2016, p. 326). As CNN possuem esse nome por utilizarem a operação matemática de convolução em pelo menos uma camada ao invés de multiplicação matricial. A formulação matemática para a convolução discreta 1D se encontra na equação (12), sendo x o sinal de entrada, $*$ a operação de convolução, k um vetor núcleo (*kernel*, na língua inglesa), s o sinal de saída e t o índice da grandeza (normalmente tempo).

$$s(t) = (x * k)(t) = \sum_{a=-\infty}^{\infty} x(a)k(t - a) \quad (12)$$

Quanto à convolução discreta 2D, sua equação se encontra em (13), na qual I é o sinal de entrada, K a matriz núcleo (*kernel*), m o tamanho da primeira dimensão, n o tamanho da segunda dimensão, i o índice da primeira dimensão, j o índice da segunda dimensão e S o sinal de saída.

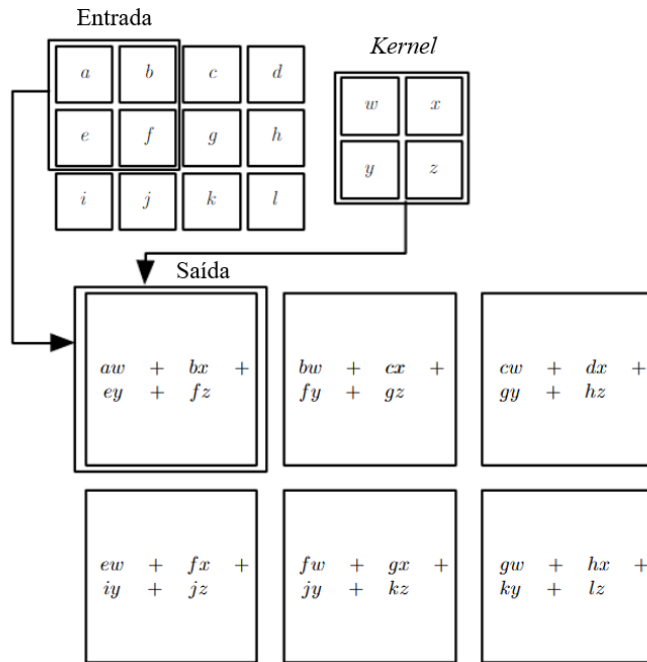
$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \quad (13)$$

Como uma forma de representar de maneira mais visual o cálculo descrito, a Figura 5 ilustra uma operação de convolução sobre uma imagem, onde o termo *input* corresponde ao dado de entrada e *output* ao dado de saída. Como mostra a Figura 5, o *kernel*, por ser aplicado em regiões da imagem, é capaz de extrair características locais dos dados. Isso é muito útil por permitir que uma mesma função extraia essas informações em toda a imagem (ZHAO; MAO; CHEN, 2019). Os pesos que definem um *kernel* de uma rede neural tipicamente começam aleatórios e são ajustados durante a fase de treino.

Além de camadas que realizam operações convolucionais, redes neurais convolucionais possuem outros tipos de camada como *pooling* e *batch normalization*. Uma camada de *pooling* é utilizada para substituir a saída da camada anterior por uma estatística de seus dados de saída (GOODFELLOW; BENGIO; COURVILLE, 2016, p. 335). Isso é proveitoso porque, ao considerar toda a vizinhança e não apenas o elemento central, as características de saída serão robustas no que diz respeito ao ruído e à distorção (ZHAO; MAO; CHEN, 2019). O *max pooling* é uma variante bastante popular, que consiste em manter apenas o maior valor da vizinhança analisada. A Figura 6 esclarece a aplicação de uma camada de *pooling*, onde cada retângulo contínuo corresponde a um *pixel* de uma imagem e a região tracejada é a vizinhança em análise, no caso uma vizinhança 2 x 2.

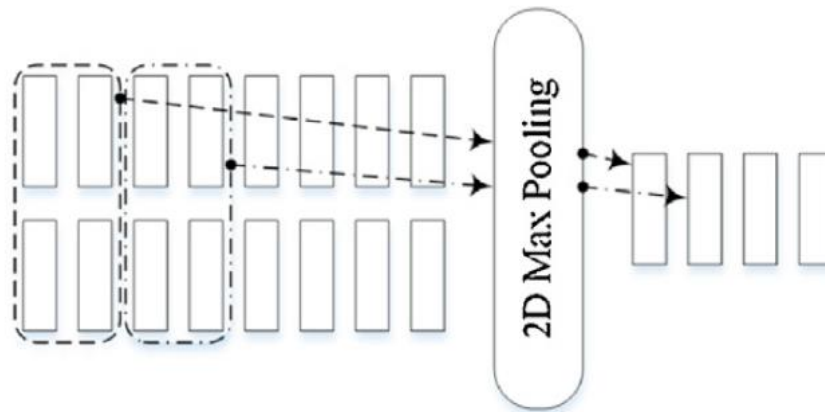
Por sua vez, a *batch normalization* é uma camada aplicada para normalizar a saída da camada precedente, subtraindo a média e dividindo pelo desvio padrão os valores de entrada. Apesar da aparente simplicidade do método, Zhao, Mao e Chen (2019) afirmam que ele é capaz de aumentar o desempenho e a estabilidade de redes neurais.

Figura 5 – Uma operação de convolução



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

Nota: Adaptado pelo autor.

Figura 6 – Esquema de uma operação de *max pooling* 2D

Fonte: Zhao, Mao e Chen (2019).

3.1.3 Redes Neurais Recorrentes

Uma rede neural recorrente é um tipo de arquitetura especializada no processamento de dados sequenciais. Goodfellow, Bengio e Courville (2016, p. 369) destacam que, dados uma função f parametrizada por um vetor de parâmetros θ e o estado do sistema s no instante $t - 1$, deseja-se encontrar os valores de θ de maneira a estimar de forma precisa o estado do sistema s no

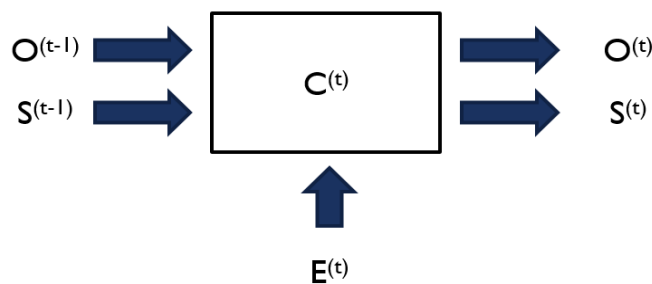
instante t . Nessas condições, f corresponde ao mapeamento que uma rede neural recorrente faz da entrada para a saída. Vê-se em (14) essa descrição genérica na forma de equação.

$$s^{(t)} = f(s^{(t-1)}; \theta) \quad (14)$$

Isso justifica a utilização de redes neurais recorrentes para o tratamento de séries temporais, posto que essas possuem forte dependência do instante anterior. Vale observar, nesse quadro, que uma característica indesejada é que a composição da mesma função múltiplas vezes, como se observa em redes recorrentes, pode resultar em comportamentos extremamente não-lineares. Além disso, se os pesos de f forem muito pequenos, multiplicá-los por eles mesmos repetidas vezes levará ao desaparecimento do valor (GOODFELLOW; BENGIO; COURVILLE, 2016, p. 369).

Dentre os algoritmos propostos para resolver as dificuldades citadas, um tipo de rede recorrente que ganhou destaque nos últimos anos é a LSTM (do inglês *long short-term memory*), que possui a capacidade de memória de longo termo (ZHAO; MAO; CHEN, 2019). Através de suas portas (que são pontos de entrada de dados), a LSTM é capaz de modelar dados novos, esquecer dados indesejados e guardar características de longo termo. A Figura 7 traz uma visão simplificada de uma célula da LSTM. Nela, t é o instante atual, E representa o vetor de entrada, O representa o vetor de estado oculto que contém o agrupamento dos vetores memorizados por cada célula da LSTM, C representa uma única célula da LSTM e S indica o agrupamento da saída de todas as células da LSTM. Um ponto a se destacar é a presença da recursão, conforme indicado na parte esquerda da Figura 7, pois cada célula recebe os estados ocultos e as saídas anteriores. Essa abordagem faz com que esse tipo de rede neural possua memória dos dados anteriores e, com isso, aprenda características globais dos dados.

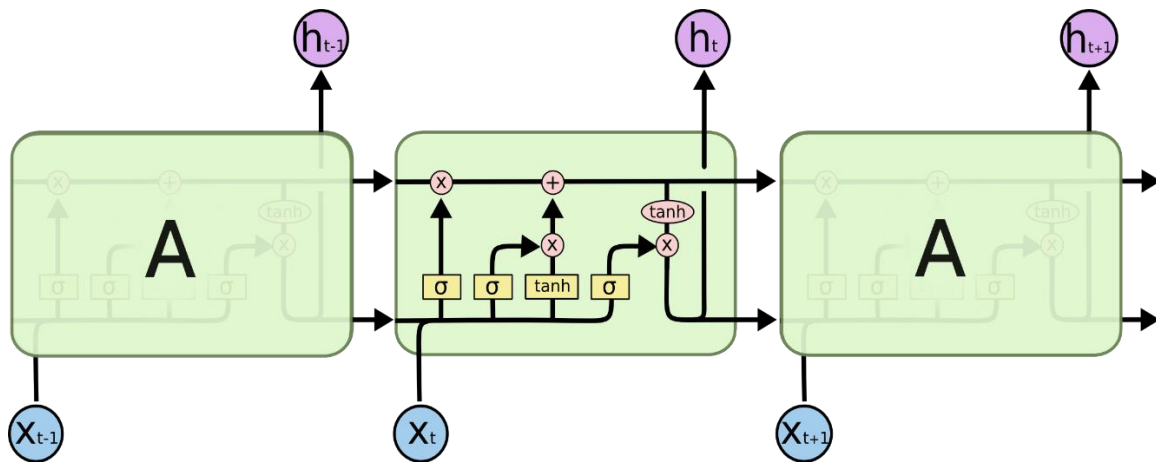
Figura 7 – Esquema simplificado de uma célula da LSTM



Fonte: Elaborado pelo autor.

Para demonstrar de forma mais específica a estrutura de uma célula da LSTM, tem-se a Figura 8. Destaca-se que cada bloco da imagem é uma mesma célula da LSTM em instantes de tempo diferentes.

Figura 8 – Diagrama interno de uma célula LSTM



Fonte: Olah (2015).

Cada célula possui um vetor de estado, um vetor de entrada e um de saída. Além disso, todas as portas têm como entrada as saídas anteriores de todas as células do sistema e os dados de entrada atuais. Assim, da esquerda para a direita na Figura 8, destacadas em amarelo, tem-se a porta do esquecimento (*forget gate*, em inglês), a porta de entrada (*input gate*, em inglês), uma camada com tangente hiperbólica e a porta de saída (*output gate*, em inglês). As portas utilizam, por padrão, uma sigmoide, como descreve a equação (15). O novo estado da célula em questão e o cálculo da camada com tangente hiperbólica são definidos na equação (16), enquanto a saída é calculada através da equação (17).

$$f^{(t)} = \sigma(\mathbf{U} \cdot \mathbf{x}^{(t)} + \mathbf{V} \cdot \mathbf{h}^{(t-1)}) \quad (15)$$

$$\mathbf{s}^{(t)} = f^{(t)}\mathbf{s}^{(t-1)} + g^{(t)}\tanh(\mathbf{U} \cdot \mathbf{x}^{(t)} + \mathbf{V} \cdot \mathbf{h}^{(t-1)}) \quad (16)$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{s}^{(t)})q^{(t)} \quad (17)$$

Nas equações acima, $f(t)$ é o valor da porta de esquecimento, $g(t)$ é o valor da porta de entrada, $q(t)$ é o valor da porta de saída, $\mathbf{s}(t)$ é o vetor estado da célula, $\mathbf{x}(t)$ o vetor de entrada, $\mathbf{h}(t)$ é o vetor de saída de todas as células, $\mathbf{U}(t)$ e $\mathbf{V}(t)$ são, respectivamente, os pesos para os dados

de entrada e as saídas anteriores, aprendidos separadamente para cada porta e $\sigma(t)$ é a função sigmoide.

Nos últimos anos, diversos estudos apontam que o uso da LSTM traz um bom desempenho quando comparado a métodos tradicionais de processamento de fala. Graves e Jaitly (2014) propuseram o uso da LSTM com o objetivo de minimizar a necessidade da utilização prévia de técnicas de extração de características quando da análise do sinal de voz. Em seus experimentos, concluíram que é viável a utilização de redes LSTM para realização dessa tarefa e que ela é capaz de processar o sinal de voz diretamente, sem necessidade de utilizar outros artifícios para extração de características. Mais recentemente, Zhao, Mao e Chen (2019) propuseram camadas convolucionais como forma de extração de características locais do espectrograma do sinal (2D) e a LSTM em sequência para modelar correlações locais e de contexto global. Ao término dos estudos, os autores concluíram que essa arquitetura é capaz de obter desempenho de classificação superior se comparada a outros métodos estabelecidos na literatura.

3.2 Algoritmo de Otimização de Hiperparâmetros

É comum que algoritmos de aprendizado de máquina precisem de alguns parâmetros antes de serem iniciados, os quais são chamados de hiperparâmetros. Nessa subseção, deseja-se escolher um conjunto de hiperparâmetros que minimize o erro de generalização de um modelo (BERGSTRÄ; BENGIO, 2012). Na prática, a otimização de hiperparâmetros é dificultada pelo fato de que se trata de uma otimização interna, posto que a própria tarefa de aprendizado já é um problema de otimização. Sabendo disso, Bergstra e Bengio (2012) afirmam que é usual a adoção de métodos iterativos e aproximados. Nesses, a tarefa crítica é a escolha do conjunto de valores que serão testados a cada iteração. Nesse contexto, os mesmos autores citam os dois métodos mais populares para essa escolha: a busca em grade e a busca aleatória.

A busca em grade consiste na definição de um conjunto de valores a serem testados para cada hiperparâmetro. Por sua vez, a busca aleatória consiste na definição da distribuição de probabilidade que guiará a amostragem de valores aleatórios para cada parâmetro. É indispensável pontuar que, na prática, não se sabe, inicialmente, qual parâmetro é mais importante para o desempenho do algoritmo. Sabendo disso, cabe afirmar que o ponto negativo da busca em grade é o fato de que, sem informação a priori, a mesma variação é considerada

para todos os parâmetros. Isso torna possível que parte das execuções não contribua para encontrar melhores configurações dos hiperparâmetros.

Em seu trabalho, Bergstra e Bengio (2012) demonstraram que a curva de desempenho do aprendizado em função dos hiperparâmetros costuma ter baixa “dimensionalidade efetiva”, ou seja, poucos hiperparâmetros possuem influência significativa nos resultados. Bergstra e Bengio (2012) escolheram diversos conjuntos de dados e observaram que, para todos eles, a busca aleatória foi mais eficiente computacionalmente e obteve melhores resultados do que a busca em grade.

Um ponto a se verificar, em virtude das informações apresentadas, é a eficácia da distribuição de probabilidade escolhida para a busca aleatória. Como a amostragem tem caráter probabilístico, o resultado obtido numa otimização aleatória não é reprodutível e pode resultar em alta variabilidade de desempenho caso o número de iterações seja insuficiente (BERGSTRA et al., 2011). Mesmo que ainda não tão difundida quanto as anteriores, uma abordagem que contorna essa dificuldade é a otimização bayesiana (BERGSTRA et al., 2011). Como o nome sugere, ela é baseada na célebre fórmula de Bayes, que se observa na equação (18).

$$p(u|\alpha) = \frac{p(\alpha|u)p(u)}{p(\alpha)} \quad (18)$$

Cabe notar que (18) está no formato para aplicação na otimização bayesiana, na qual α é um vetor de hiperparâmetros, u é a medida de desempenho do algoritmo de aprendizado de máquina, $p(\alpha)$ é a probabilidade de se escolher o vetor de hiperparâmetros α , $p(u)$ é a probabilidade de se obter um desempenho de aprendizado de máquina u , $p(\alpha|u)$ é a probabilidade condicional de escolha de α dado u e $p(u|\alpha)$ a probabilidade condicional de se obter u utilizando α . Bergstra e outros (2011) trazem a formulação de um otimizador bayesiano, começando pela definição da função de custo a ser otimizada: a melhoria esperada (EI, do inglês *expected improvement*), que tem sua equação registrada em (19). Assim como em (18), α é um vetor de hiperparâmetros, u é a medida $f(\alpha)$ de desempenho do algoritmo de aprendizado de máquina f e u^* o menor valor de custo já encontrado (posto que é uma função de custo, quanto menor esse valor, melhor o resultado). Nesse caso, a EI representa a esperança de que um modelo de aprendizado de máquina f possua custo u menor que u^* .

$$EI_{u^*}(\boldsymbol{\alpha}) := \int_{-\infty}^{\infty} \max(u^* - u, 0) p(u|\boldsymbol{\alpha}) du \quad (19)$$

Bergstra e outros (2011) escolheram como abordagem de otimização o estimador de Parzen com estrutura em árvore (TPE, do inglês *tree-structured Parzen estimator*). Para que a otimização seja possível, é necessário possuir os 3 termos do lado direito da equação (18) de forma a realizar a substituição de (18) em (19). O TPE define o $p(\boldsymbol{\alpha}|u)$ segundo a equação (20), em que $l(\boldsymbol{\alpha})$ é a densidade de probabilidade formada usando as observações que resultam em um valor de u maior que u^* , enquanto $g(\boldsymbol{\alpha})$ é formada com o restante das observações.

$$p(\boldsymbol{\alpha}|u) = \begin{cases} l(\boldsymbol{\alpha}) & \text{se } u < u^* \\ g(\boldsymbol{\alpha}) & \text{se } u \geq u^* \end{cases} \quad (20)$$

O TPE escolhe u^* para ser um quantil q dos valores obtidos tal que $p(u < u^*) = q$, de maneira a não necessitar de uma definição específica de $p(u)$. Em posse dessas informações, encontra-se (21) pela substituição de (18) em (19). Lança-se mão também das equações (22) e (23), sendo que esta última aplica a definição de (20). Com essas informações, Bergstra e outros (2011) realizam mais algumas operações matemáticas e, então, chegam em (24).

$$EI_{u^*}(\boldsymbol{\alpha}) = \int_{-\infty}^{u^*} (u^* - u) \frac{p(\boldsymbol{\alpha}|u)p(u)}{p(\boldsymbol{\alpha})} du \quad (21)$$

$$q = p(u < u^*) \quad (22)$$

$$p(\boldsymbol{\alpha}) = \int_{\mathbb{R}} p(\boldsymbol{\alpha}|u)p(u) du = ql(\boldsymbol{\alpha}) + (1 - q)g(\boldsymbol{\alpha}) \quad (23)$$

$$EI_{u^*}(\boldsymbol{\alpha}) = \frac{qu^*l(\boldsymbol{\alpha}) - l(\boldsymbol{\alpha}) \int_{-\infty}^{\infty} p(u) du}{ql(\boldsymbol{\alpha}) + (1 - q)g(\boldsymbol{\alpha})} \propto \left(q + \frac{g(\boldsymbol{\alpha})}{l(\boldsymbol{\alpha})} (1 - q) \right)^{-1} \quad (24)$$

O ponto importante a se notar aqui é que para minimizar a EI, desejam-se vetores $\boldsymbol{\alpha}$ com alta probabilidade em $l(\boldsymbol{\alpha})$ e baixa probabilidade em $g(\boldsymbol{\alpha})$. Assim, a partir da análise de diversos vetores de hiperparâmetros $\boldsymbol{\alpha}$, o algoritmo busca minimizar a esperança de que um resultado melhor seja alcançado. Bergstra e outros (2011) realizaram 4 experimentos com bases de dados

diferentes e puderam comprovar que em todos os testes o TPE teve desempenho igual ou superior à busca aleatória e, por isso, esse algoritmo foi utilizado no presente estudo.

3.3 Algoritmos de Aumento de Dados

É comum encontrar bases de dados desbalanceadas no domínio de SER, segundo Chatziagapi e outros (2019). Isso significa que, para melhorar o desempenho do aprendizado de máquina, algumas técnicas podem ser implementadas. Nesse contexto, uma opção comum é a utilização do aumento de dados (CHATZIAGAPI et al., 2019). Essa abordagem consiste na criação de dados artificiais, normalmente a partir de transformações aplicadas sobre os dados existentes.

No que tange ao aumento de dados na análise de sinais de fala, Fukuda e outros (2018) propuseram a inserção de ruído gaussiano branco, posto que é uma abordagem simples e que adiciona robustez ao ruído no processo de aprendizagem. Nesse caso, uma nova amostra é gerada a partir da escolha aleatória de um elemento já existente na base de dados seguido da adição de ruído gaussiano branco em uma potência tal que o áudio não deixe de ser interpretável.

Ainda nesse contexto, outro trabalho notável é o de Aldeneh e Provost (2017). Os autores realizaram o aumento de dados através da alteração da velocidade do sinal de voz para SER e obtiveram melhora no desempenho de classificação. Dois fatores de velocidade foram utilizados: 0,9 e 1,1. Fatores abaixo de 1 promovem retardo na velocidade do sinal enquanto fatores acima a aceleram. Uma maneira simples de implementar esse algoritmo consiste na conversão da frequência de amostragem: o sinal é reamostrado para uma frequência menor ou maior, dependendo do fator de velocidade. Isso provoca também a modificação da frequência do sinal escutado, visto que alterações no sinal no domínio do tempo também o modificam se observado no domínio da frequência, o que pode ser considerado um efeito colateral. Alguns métodos mais complexos procuram amenizar esse efeito através da análise da fase da decomposição em frequência, porém tal aprofundamento foge ao escopo do presente projeto.

3.4 Métricas de Desempenho

No contexto de algoritmos de aprendizado de máquina, é interessante o uso de métricas para avaliar o desempenho dos métodos propostos. Isso possibilita a identificação de pontos de

melhoria e verificar se o projeto em desenvolvimento tem desempenho próximo ao de trabalhos similares que já se encontram na literatura. Para tarefas de classificação, as métricas são baseadas nos valores obtidos na matriz de confusão, como o exemplo demonstrado no Quadro 1, onde se tem a possibilidade das amostras pertencerem ou não à classe analisada. O valor real indica a condição das amostras analisadas por seres humanos: o caso positivo engloba as amostras que pertencem à classe em questão e o caso negativo engloba as amostras que não pertencem a essa classe. Por outro lado, o valor previsto mostra o resultado do classificador: o positivo indica as amostras que foram classificadas como pertencentes à classe analisada e o negativo aponta as amostras classificadas como não pertencentes a essa classe. Dessa forma, os verdadeiros positivos (VP) são amostras positivas rotuladas corretamente pelo classificador, assim como os verdadeiros negativos (VN) são as amostras negativas identificadas de modo correto. Os falsos positivos (FP) e os falsos negativos (FN) acontecem quando as amostras negativas e positivas, respectivamente, são identificadas de forma errônea.

Quadro 1 – Exemplo de matriz de confusão

		Valor Real	
		Positivo	Negativo
Valor Previsto	Positivo	VP	FN
	Negativo	FP	VN

Fonte: Elaborado pelo autor.

Com base nos valores obtidos para a matriz de confusão, pode-se calcular a métrica de desempenho F1 de acordo com a equação (25).

$$F1 = \frac{VP}{VP + 0,5(FP + FN)} \quad (25)$$

A partir de (25), calcula-se o valor F1 para cada classe separadamente. Para resumir o desempenho do algoritmo em apenas um número, é usual a adoção de algum tipo de média. Uma escolha típica, como se observa no trabalho de Gering, Ciarelli e Salles (2019), é o F1 ponderado, que consiste na média dos valores F1 de cada classe ponderados pela quantidade de amostras verdadeiras de cada uma.

4 METODOLOGIA

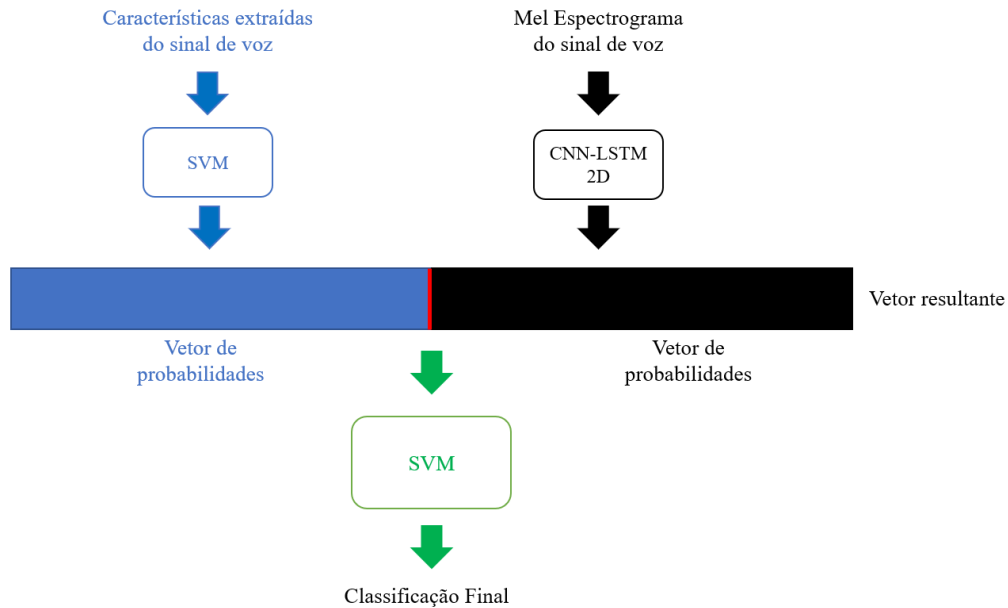
O presente estudo tem como objetivo principal otimizar um arranjo de classificadores criados com base no método proposto por Gering, Ciarelli e Salles (2019). Como se trata de um estudo de caso, pode-se dizer que a pesquisa é de natureza aplicada. Quanto aos objetivos, identifica-se que a pesquisa é explicativa, pois busca-se encontrar um método eficaz e justificar porque cada etapa realizada foi bem-sucedida ou não. Além disso, a forma de abordagem do problema tem caráter quantitativo, pois a avaliação do desempenho do trabalho e de cada método específico se dá por medidas numéricas.

A abordagem proposta neste trabalho se baseia em classificadores utilizados em dois níveis. No primeiro nível, dois classificadores são aplicados sobre os dados, cada um recebendo características diferentes do sinal como entrada. No segundo nível há um classificador que utiliza as saídas dos classificadores anteriores para identificar a emoção no sinal.

Para melhor compreensão da abordagem proposta, seu esquemático é apresentado na Figura 9. Ressalta-se que uma SVM e uma CNN-LSTM-2D compõem, em paralelo, o primeiro nível de classificação e possuem saídas que contém a probabilidade de o áudio pertencer a cada classe. O classificador de segundo nível também é uma SVM e possui como entrada a concatenação das probabilidades obtidas por cada elemento do nível anterior, sendo que sua saída indica a classificação final. Essa abordagem em dois níveis de classificação foi inspirada no trabalho de Gering, Ciarelli e Salles (2019).

A fim de validar o desempenho dos classificadores, é importante realizar a validação cruzada. Nela, separam-se diversos grupos de dados de treino, validação e teste. Os algoritmos de aprendizado de máquina são treinados nos grupos de treino, ajustados sobre o grupo de validação e, por fim, avaliados sobre o grupo de teste. Cada divisão da base de dados em grupos de treino, validação e teste será chamada de execução.

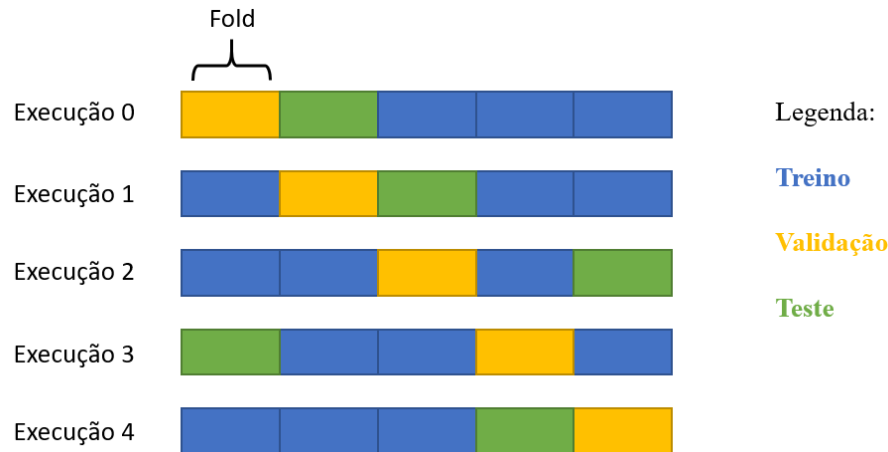
Figura 9 – Esquema explicativo do classificador proposto



Fonte: Elaborado pelo autor.

No presente trabalho, escolheu-se a metodologia *k-fold cross validation* para validação dos resultados, conforme descrita por Gering, Ciarelli e Salles (2019). Nesta metodologia, os dados da base são divididos em k partes (*folds*) de aproximadamente mesmo tamanho. Na primeira execução dos algoritmos, uma destas partes será usada para teste, uma para validação e as demais para treino. Na próxima execução partes diferentes serão usadas para teste e validação e o resto para treino. Esse procedimento se repete k , ou seja, haverá k execuções, de forma que todas as partes sejam usadas uma vez como teste e validação. Neste trabalho será utilizado o valor de $k = 5$. A Figura 10 ilustra o procedimento. Doravante, será utilizado o termo inglês *fold* para se referir à parte de validação ou teste, sabendo que cada um corresponde a uma subdivisão dos dados da execução. Dessa forma, *fold 0* de validação, por exemplo, se refere ao conjunto de validação da execução 0.

É importante destacar que a divisão em *folds* ocorre de forma estratificada, ou seja, de maneira a manter a distribuição de amostras por classe constante. Ocorre que, dada a pequena quantidade de amostras da base, não foi possível fazer a estratificação de maneira precisa. Em todas as execuções, o conjunto de validação ficou com distribuição um pouco diferente dos conjuntos de treino e teste. O Quadro 2 mostra como ficou a porcentagem de amostras para a primeira execução a título de exemplo.

Figura 10 – Procedimento do *k-fold cross validation* para $k = 5$ 

Fonte: Elaborado pelo autor.

Quadro 2 – Porcentagem de amostras por classe em cada conjunto da execução 0

Sentimento	Original Berlin	Treino	Validação	Teste
Raiva	23,7%	24,0%	26,2%	24,3%
Tédio	15,1%	14,3%	13,1%	15,0%
Neutro	14,8%	14,6%	17,8%	14,0%
Alegria	13,3%	12,5%	15,9%	14,0%
Medo	12,9%	15,3%	7,5%	13,1%
Tristeza	11,6%	10,6%	15,0%	11,2%
Desgosto	8,6%	8,7%	4,7%	8,4%

Fonte: Elaborado pelo autor.

Tendo em mente a arquitetura proposta, a ordem de realização das tarefas foi a seguinte:

- Pré-processamento dos áudios;
- Extração de características do sinal de voz;
- Implementação e ajuste da SVM do primeiro nível (superior);
- Extração do mel espectrograma dos sinais de voz (entrada da CNN-LSTM 2D);
- Implementação e ajuste da CNN-LSTM 2D;
- Aumento de dados (*data augmentation*);
- Implementação e ajuste da SVM do segundo nível e classificação do áudio.

4.1 Pré-processamento dos Áudios

Os sinais de voz analisados nesse estudo estão presentes na base de dados Berlin, conforme seção 2. Um dado importante é a duração desses áudios que não é a mesma para todos os

elementos da base de dados. Isso é um impeditivo para a utilização de redes convolucionais, pois ela exige que todos os dados analisados possuam mesmo tamanho em todas as dimensões.

Zhao, Mao e Chen (2019), que também utilizaram redes neurais convolucionais no estudo de SER na base Berlin, propuseram a utilização de áudios com duração de 8 segundos. As gravações com duração menor que 8 segundos seriam preenchidas com valor 0 e as gravações mais longas do que 8 segundos seriam segmentadas para possuir a mesma duração. Um possível problema dessa abordagem é a presença de muitos sinais pequenos na base de dados Berlin, já que, conforme dito na seção 2, a média de duração dos sinais é de 2,78 segundos e desvio padrão de 1,03.

Por sua vez, Gering, Ciarelli e Salles (2019) utilizaram uma janela de 1,2 segundos centralizada na metade do sinal de áudio para gerar o mel espectrograma utilizado na rede convolucional. Apesar dos bons resultados indicados pelos autores, nota-se que o valor de 1,2 segundos é menor que a duração média dos sinais da base Berlin. Isso faz com que uma parte significativa dos dados não seja considerada nessa análise.

Como forma de encontrar uma abordagem que diminua a utilização de preenchimento com zeros ou a não consideração de parte significativa do sinal de voz, escolheu-se o uso da segmentação dos áudios através do uso de janelas de mesmo tamanho com sobreposição (utilizando janelas de Hamming). Fukuda e outros (2018) destacam que é comum o uso de janelas do sinal de áudio para a análise desses em diversas áreas que envolvem processamento de áudio. Como o tamanho da janela e o passo utilizado nessa segmentação não é de trivial escolha, utilizaram-se métodos de otimização bayesiana para estimar os valores que proporcionam o melhor desempenho de classificação.

4.2 Extração de Características do Sinal de Voz

No domínio de SER é usual que se faça a extração de características do sinal de voz para que um algoritmo de aprendizado de máquina realize a classificação. Assim como se observa no artigo base, a implementação da SVM de primeiro nível foi baseada no trabalho de Shen, Changjun e Chen (2011). Esses autores relatam que o melhor desempenho foi alcançado com as características de energia, frequência e os coeficientes da predição linear dos coeficientes

mel cepstrais (LPCMCC, do inglês *linear prediction coefficients mel cepstrum coefficients*) dos áudios. Isso se dá porque todas elas possuem variações significativas em função da emoção expressa pelo locutor. Antes do cálculo desses valores, o áudio é fragmentado em pequenas janelas para então aplicar os cálculos sobre cada janela e se extrair as estatísticas dos valores obtidos. Dessa forma, utilizou-se a metodologia proposta em Shen, Changjun e Chen (2011) para a extração de todas as características usadas na SVM de primeiro nível.

Quanto às características de energia, calculou-se a energia de cada janela de áudio. Em seguida, extraem-se as 19 características de interesse. São elas:

- O máximo, a média e a variância da energia;
- A duração máxima, média e mediana de inclinações ascendentes e descendentes de energia;
- Os valores máximos, médios e medianos de inclinações ascendentes e descendentes de energia;
- O intervalo interquartil de inclinações ascendentes e descendentes de energia;
- O intervalo interquartil da duração de inclinações ascendentes e descendentes de energia.

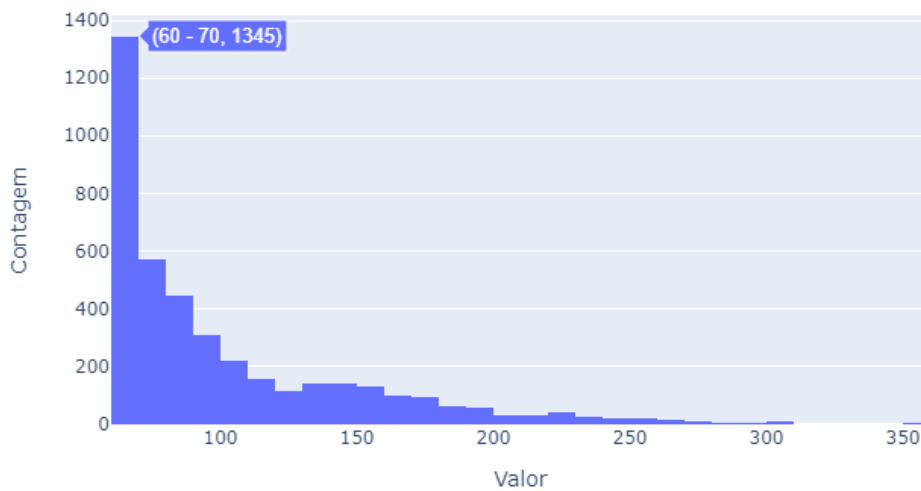
No que tange às características de frequência, o parâmetro a ser considerado é a frequência fundamental do sinal de voz. Para isso, utiliza-se o método Yin, que é baseado em funções de autocorrelação (DE CHEVEIGNÉ; KAWAHARA, 2002). Em posse da frequência fundamental de cada janela, calculam-se as mesmas 19 estatísticas que foram utilizadas para a energia, conforme descrito acima.

Por fim, tem-se a LPCMCC. Essa é uma característica espectral que combina os pontos positivos de dois outros métodos de análise de características espectrais: coeficientes cepstrais de predição linear e coeficientes mel-cepstrais (SHEN; CHANGJUN; CHEN, 2011). Extraem-se os coeficientes LPCMCC até a décima quarta ordem para cada janela do sinal e então calculam-se as seguintes estatísticas: média, variância, máximo e mínimo valor de cada coeficiente ao longo das janelas. Assim, verifica-se que o vetor resultante terá 56 valores.

Ao todo, são extraídas 94 características para o sinal de voz. Durante a realização dos experimentos, porém, algumas dessas características apresentaram uma distribuição muito

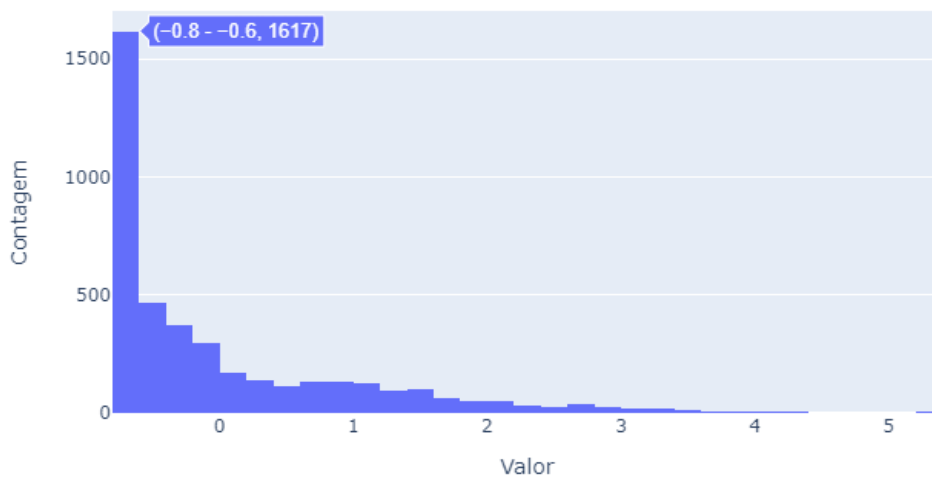
assimétrica, conforme se nota no exemplo da Figura 11, onde a simbologia 60 - 70 indica que valores entre 60 Hz e 70 Hz foram os mais comuns, mas existem valores de frequência muito mais elevados (por exemplo, maiores que 200 Hz). Essa distribuição assimétrica afeta muito a padronização dos dados pela média e desvio padrão, como se observa na Figura 12, devido à presença de pontos fora da curva (*outliers*, em inglês), como se vê na Figura 11. De forma a obter uma normalização mais robusta a ruídos, substitui-se a média pela mediana e o desvio padrão pelo desvio padrão em relação à mediana no momento da normalização. O resultado se observa na Figura 13. Nota-se que os dados ganharam aspecto menos disperso.

Figura 11 – Histograma de uma característica com distribuição assimétrica



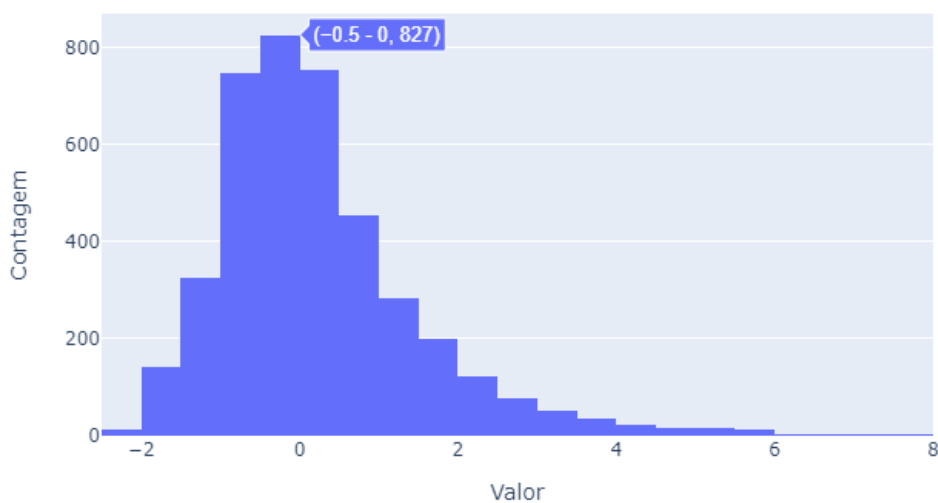
Fonte: Elaborado pelo autor.

Figura 12 – Histograma da característica normalizada com desvio padrão



Fonte: Elaborado pelo autor.

Figura 13 – Normalização da característica com o desvio padrão da mediana



Fonte: Elaborado pelo autor.

4.3 Implementação e Ajuste da SVM do Primeiro Nível

Conforme descrito na seção 4.2, os dados de entrada da SVM do primeiro nível são as características extraídas dos sinais de voz. Nos estudos de Gering, Ciarelli e Salles (2019), a função *kernel* utilizada foi a gaussiana. Sabendo disso, para verificar se essa é a melhor escolha, decidiu-se ajustar a SVM com *kernel* gaussiano e linear. A seguinte ordem foi seguida para o ajuste desses algoritmos:

- i. Ajuste conjunto do parâmetro de regularização (normalmente representado pela letra *C*) e, para o caso do *kernel* gaussiano, o parâmetro *gama* (inverso do raio de influência dos vetores suporte);
- ii. Ajuste da escolha de características mais importantes;
- iii. Ajuste do tamanho da janela a se utilizar na segmentação;
- iv. Ajuste do tamanho do passo entre janelas na segmentação.

Convém salientar que o passo entre janelas é definido como o deslocamento em segundos que se faz, a partir do início da janela anterior, para delimitar o início da janela seguinte. Destaca-se também que, a partir da segunda etapa, apenas o *kernel* gaussiano foi ajustado, posto que foi verificado que seu desempenho é normalmente superior. Além disso, a implementação do otimizador bayesiano utilizada foi a da biblioteca *hyperopt* para a linguagem *python*.

Como é necessário adotar algum valor inicial para o tamanho da janela e o passo entre janelas para fazer os ajustes até a terceira etapa, utilizou-se uma janela de 0,7 s e passo 0,3 s. Conforme descrito na primeira etapa, começou-se pelo ajuste dos hiperparâmetros das SVM linear e gaussiana. O Quadro 3 mostra a faixa de valores estudada para cada parâmetro. Vale mencionar que, para todas os ajustes realizadas, a distribuição a priori utilizada no otimizador bayesiano foi a uniforme.

Quadro 3 – Faixa de valores do ajuste de hiperparâmetros da SVM de primeiro nível

Parâmetro	Valor mínimo	Valor máximo
<i>C</i> da SVM Linear	0	2
<i>C</i> da SVM Gaussiana	0	2
<i>gama</i>	0	1

Fonte: Elaborado pelo autor.

Quanto à segunda etapa, a análise visa identificar se todas as 94 características são relevantes para a classificação. Para realizar essa análise, a otimização bayesiana permite que cada característica, de forma independente, seja escolhida ou não para a classificação. O critério de escolha seguiu essas etapas: primeiro, executar a otimização bayesiana com 150 iterações para cada uma das 5 execuções (utilizando os conjuntos de treino e validação). Em seguida, tomam-se os resultados das últimas 30 iterações de cada execução, pois ao final do ajuste os resultados tendem a convergir em torno da melhor escolha. Logo após, une-se os resultados do passo anterior realizados em todas as execuções, totalizando 150 escolhas. Por fim, as características presentes em pelo menos 40% das escolhas são consideradas relevantes.

Um importante parâmetro a se ajustar é o tamanho da janela para a segmentação do sinal de áudio. Como não se tem nenhuma informação a priori, escolhe-se na etapa 3 uma ampla faixa de valores para a otimização bayesiana. Juntamente com o tamanho da janela, o passo entre janelas também foi ajustado, mas esse parâmetro demonstrou sofrer grande influência do tamanho da janela e, por isso, foi ajustado em uma outra etapa. O Quadro 3 contém a faixa de valores utilizada nesse ajuste.

Por fim, a quarta etapa consiste no ajuste do tamanho do passo entre janelas. Após a definição do tamanho de janela que proporciona o melhor desempenho, pode-se escolher uma faixa de

valores para otimizar o valor do passo. Os valores máximo e mínimo considerados estão no Quadro 4.

Quadro 4 – Faixa de valores do ajuste do tamanho da janela de áudio

Parâmetro	Valor mínimo	Valor máximo
Tamanho da janela	0,3 s	0,9 s
Tamanho do passo	0,15 s	0,5 s

Fonte: Elaborado pelo autor.

É de suma importância ressaltar que a SVM do primeiro nível possui saída probabilística. Como esse algoritmo é, por natureza, um classificador binário, é necessária a implementação de alguma abordagem que estime as probabilidades. Platt (1999) propôs uma regressão logística para o cálculo das probabilidades em um caso de classificação binária. Alguns anos depois, Wu, Lin e Weng (2004) conseguiram generalizar o método de Platt para classificadores de múltiplas classes. Esse último está implementado na popular biblioteca *scikit-learn* para a linguagem *python*, a qual foi utilizada no presente estudo.

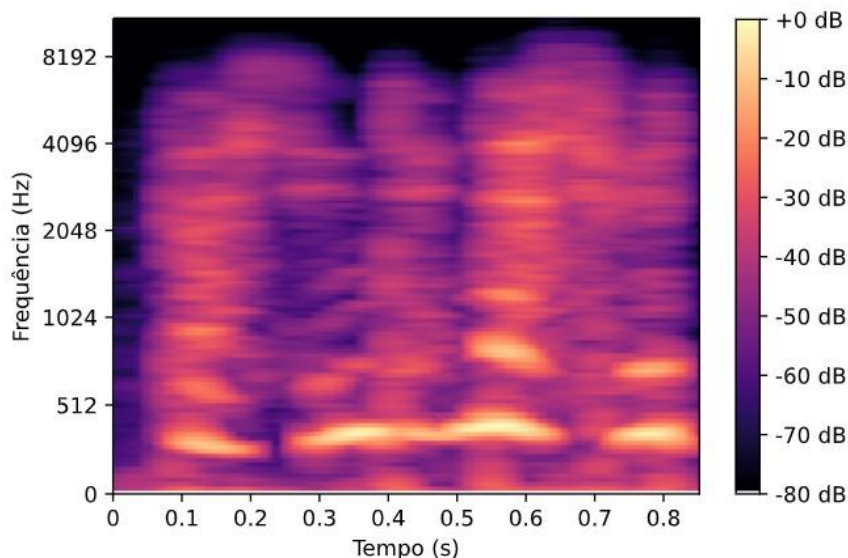
4.4 Extração do Mel Espectrograma dos Sinais de Voz (Entrada da SVM)

O espectrograma é uma ferramenta para análise das frequências presentes em um sinal. Nele, é possível observar a potência presente em uma gama de frequências ao longo do tempo. Essa é uma poderosa ferramenta para fins de SER, como indicam Zhao, Mao e Chen (2019) e seus bons resultados. No caso de áudios, é comum a visualização do espectrograma não na escala linear, mas na escala Mel, o que é conhecido como Mel espectrograma. A equação (26) indica o procedimento para converter um valor F_{Hz} de frequência em Hertz para o equivalente Mel F_{mel} . Essa escala de frequência foi obtida experimentalmente por Stevens, Volkman e Newman (1937) e foi baseada na forma como a audição humana distingue diferentes frequências. Pode-se observar também um exemplo de Mel espectrograma de uma janela de áudio da Berlin na Figura 14.

$$F_{mel} = 2295 \log_{10} \left(1 + \frac{F_{Hz}}{700} \right) \quad (26)$$

Na prática, utilizou-se a biblioteca computacional Librosa para o cálculo dos Mel espectrogramas com janelas de Hamming.

Figura 14 – Mel espectrograma de uma janela de áudio



Fonte: Elaborado pelo autor.

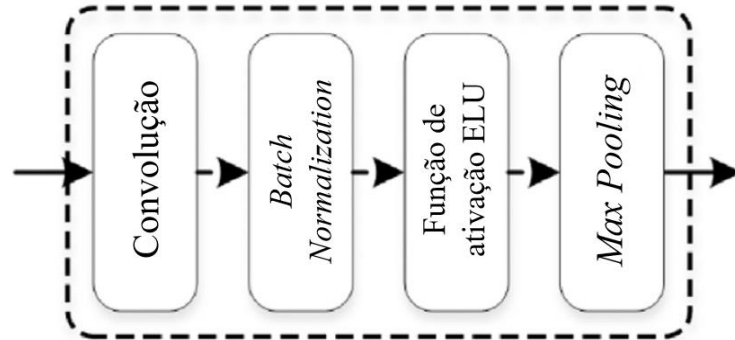
4.5 Implementação e Ajuste da CNN-LSTM 2D

A CNN-LSTM 2D é o classificador que utiliza os Mel espectrogramas como sinal de entrada. Ao contrário da ideia da SVM que realiza apenas a classificação, o objetivo desse método é realizar ambas as tarefas de extração de características e classificação. Zhao, Mao e Chen (2019) frisam que isso se dá através das camadas convolucionais que permitem o ajuste dos pesos que compõem os filtros, tendo esses últimos a responsabilidade de extrair as características locais do sinal de voz. A camada LSTM, por sua vez, tem a capacidade de identificar dependências de longo termo, o que permite o aprendizado de características do áudio em um contexto global.

A CNN-LSTM 2D utilizada no presente projeto foi baseada nos estudos de Gering (2019) e Zhao, Mao e Chen (2019). A fim de esclarecer a arquitetura do algoritmo, a Figura 15 demonstra o que foi chamado de um bloco de aprendizado de características locais (BACL). Em seguida, tem-se a Figura 16 que traz um diagrama de blocos da arquitetura implementada e o Quadro 5 que indica as dimensões de cada camada. No Quadro 5, percebe-se que os Mel espectrogramas gerados possuem dimensões 218 x 213 e vê-se a descrição de todas as camadas

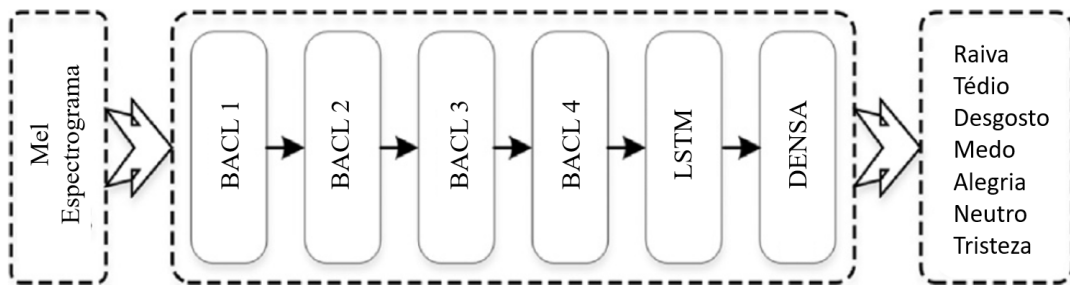
utilizadas para a construção da rede neural em questão. Ele foi colorido de forma a facilitar a distinção de cada BACL.

Figura 15 – Bloco de aprendizado de características locais (BACL)



Fonte: Zhao, Mao e Chen (2019).
Nota: Adaptado pelo autor.

Figura 16 – Diagrama de blocos da CNN-LSTM 2D



Fonte: Zhao, Mao e Chen (2019).
Nota: Adaptado pelo autor.

É conveniente destacar que a camada densa apresentada na Figura 16 e no Quadro 5 é uma camada de neurônios totalmente conectados. Outrossim, a camada de remodelagem é a transformação da saída da última camada de *max pooling* de três para duas dimensões, de forma que a matriz se torne compatível para a utilização na LSTM.

Posto que ambos os autores obtiveram bons resultados quando da utilização dessa arquitetura, optou-se por conservar a estrutura geral e apenas ajustar a taxa de aprendizado através da otimização bayesiana, percorrendo o intervalo de valores de 0 a 0,003. Destaca-se também que a saída dessa rede é um vetor com 7 valores que indicam a probabilidade de o áudio analisado pertencer a cada uma das classes. Isso se obtém a partir da utilização da função *softmax* como função de ativação da camada densa. A fórmula da *softmax* está representada na equação (27), dado que x é o vetor de entrada e C o conjunto das classes de emoções.

Quadro 5 – Descrição das camadas da CNN-LSTM 2D

Nome	Dimensão de saída	Tamanho do <i>kernel</i>	Passo
Convolutacional 1	218 x 213 x 64	3 x 3	1 x 1
<i>Batch Normalization</i> 1	218 x 213 x 64	-	-
ELU 1	218 x 213 x 64	-	-
<i>Pooling</i> 1	109 x 106 x 64	2 x 2	2 x 2
Convolutacional 2	109 x 106 x 64	3 x 3	1 x 1
<i>Batch Normalization</i> 2	109 x 106 x 64	-	-
ELU 2	109 x 106 x 64	-	-
<i>Pooling</i> 2	27 x 26 x 64	4 x 4	4 x 4
Convolutacional 3	27 x 26 x 64	3 x 3	1 x 1
<i>Batch Normalization</i> 3	27 x 26 x 64	-	-
ELU 3	27 x 26 x 64	-	-
<i>Pooling</i> 3	6 x 6 x 128	4 x 4	4 x 4
Convolutacional 4	6 x 6 x 128	3 x 3	1 x 1
<i>Batch Normalization</i> 3	6 x 6 x 128	-	-
ELU 3	6 x 6 x 128	-	-
<i>Pooling</i> 4	1 x 1 x 128	4 x 4	4 x 4
Remodelagem	1 x 128	-	-
LSTM	256	256	-
Densa	7	7	-

Fonte: Zhao, Mao e Chen (2019).

Nota: Adaptado pelo autor.

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^C e^{x_j}} \text{ para } i = 1, \dots, C \text{ e } \mathbf{x} = (x_1, \dots, x_c) \in \mathbb{R}^C \quad (27)$$

4.6 Aumento de Dados (*Data Augmentation*)

A seção 3.3 trouxe uma breve elucidação de alguns algoritmos de aumento de dados utilizados na área de SER. Para o presente estudo, escolheu-se aplicar a adição de ruído gaussiano branco e alteração da velocidade do áudio. Para ambos os métodos, sabendo que o intuito é diminuir o desbalanceamento da base de dados (assim como aumentar a quantidade de dados para treino), calculou-se a quantidade de janelas de áudios a serem geradas em cada método com base na diferença entre a quantidade de dados presentes em cada classe. Nesse contexto, a equação (28) foi utilizada para calcular a diferença da quantidade de janelas de áudio por classe, a equação (29) traz a quantidade de janelas de áudio geradas para a classe majoritária e a equação (30) a quantidade de janelas de áudio geradas para as classes minoritárias.

$$n_i = 0,3(j_M - j_i); i \in C \quad (28)$$

$$B = \frac{\min(n_i)}{2}; i \in (C - M) \quad (29)$$

$$N_i = n_i + B; i \in C \quad (30)$$

Destaca-se que C é o conjunto das 7 classes, M é a classe majoritária, j_i é a quantidade de janelas iniciais da classe i , B é o valor base de janelas a serem geradas, n_i é a quantidade de janelas a serem geradas para a classe i em função da diferença entre o número de janelas de M e i , enquanto N_i é a quantidade de janelas a serem geradas. Cabe ressaltar também a presença do valor 0,3 (ou 30%) que indica a quantidade de amostras a serem geradas em função da diferença do número de janelas entre as classes. Ele foi escolhido porque, para alguns *folds*, a diferença da quantidade de janelas da classe majoritária para a minoritária se mostrou muito expressiva, por exemplo, nos dados de treino da execução 0: 949 janelas na classe majoritária contra 463 na minoritária. A diferença entre elas é maior que a própria quantidade de janelas da classe minoritária. Para que o número de janelas geradas não fosse muito expressivo em comparação à quantidade inicial, escolheu-se esse fator de projeto de 0,3.

Convém pontuar que os dados base utilizados para aplicar as técnicas de aumento de dados são escolhidos aleatoriamente (com distribuição uniforme) entre os dados da classe em questão. Assim, é possível que um áudio não seja escolhido, seja escolhido apenas para um dos métodos ou para ambos, de forma que nesse último caso, ambas as técnicas serão aplicadas em sequência.

No que tange ao método de adição de ruído, para cada classe, obtém-se valores aleatórios (um para cada áudio a ser gerado) entre 50 e 80 decibéis para a relação sinal-ruído desejada. Logo após, calcula-se a amplitude média das janelas daquela classe e, em seguida, encontra-se a amplitude do ruído a ser adicionado a elas. A equação (31) traz a definição da relação sinal ruído, enquanto a operação descrita anteriormente está na equação (32), sendo que (32) resulta da manipulação de (31). Nelas, A_r é a amplitude do ruído a ser inserido, A_m a amplitude média dos sinais da classe em questão, SNR é a relação sinal ruído em decibéis e S o conjunto de amostras de áudio base escolhidas.

$$SNR = 20 \log_{10} \left(\frac{A_m}{A_r} \right) \quad (31)$$

$$A_r = \frac{A_m}{10^{\frac{SNR_s}{20}}}, s \in S \quad (32)$$

O outro método estudado para aumento de dados é a alteração da velocidade do sinal de voz, conforme técnicas descritas na seção 3.3. Para a escolha do fator de velocidade, valores foram amostrados de uma distribuição uniforme entre 0,9 e 1,1, valores esses baseados no trabalho de Aldeneh e Provost (2017). Assim, cada áudio gerado terá um fator de velocidade distinto.

É importante pontuar que esse método altera o tamanho do sinal resultante, o que poderia ser um empecilho para a rede convolucional. Fatores maiores do que 1 diminuem a duração do sinal, enquanto fatores menores do que 1 a aumentam. Para corrigir essa característica indesejável, sinais maiores do que a duração padrão serão segmentados e sinais menores terão adição de zeros. Para esse último caso, escolheu-se aleatoriamente se a adição ocorreria no início ou no fim do áudio. A implementação desse método se deu através dos recursos da biblioteca *librosa* para a linguagem *python*.

4.7 Implementação e Ajuste da SVM de Segundo Nível e Classificação do Áudio

Em posse das saídas de ambos os classificadores do primeiro nível, torna-se necessária a implementação do classificador do segundo nível. Cada vetor de entrada possui 14 valores (7 vindas da SVM e 7 da CNN-LSTM 2D) que indicam as probabilidades de pertinência da amostra às 7 classes do problema em questão. Como se escolheu uma SVM com *kernel* gaussiano, fez-se necessário o ajuste dos parâmetros *C* e *gama*, de maneira similar ao ajuste da SVM do primeiro nível descrita na seção 4.3. O Quadro 6 demonstra os valores analisados.

Quadro 6 – Faixa de valores do ajuste de hiperparâmetros da SVM de segundo nível

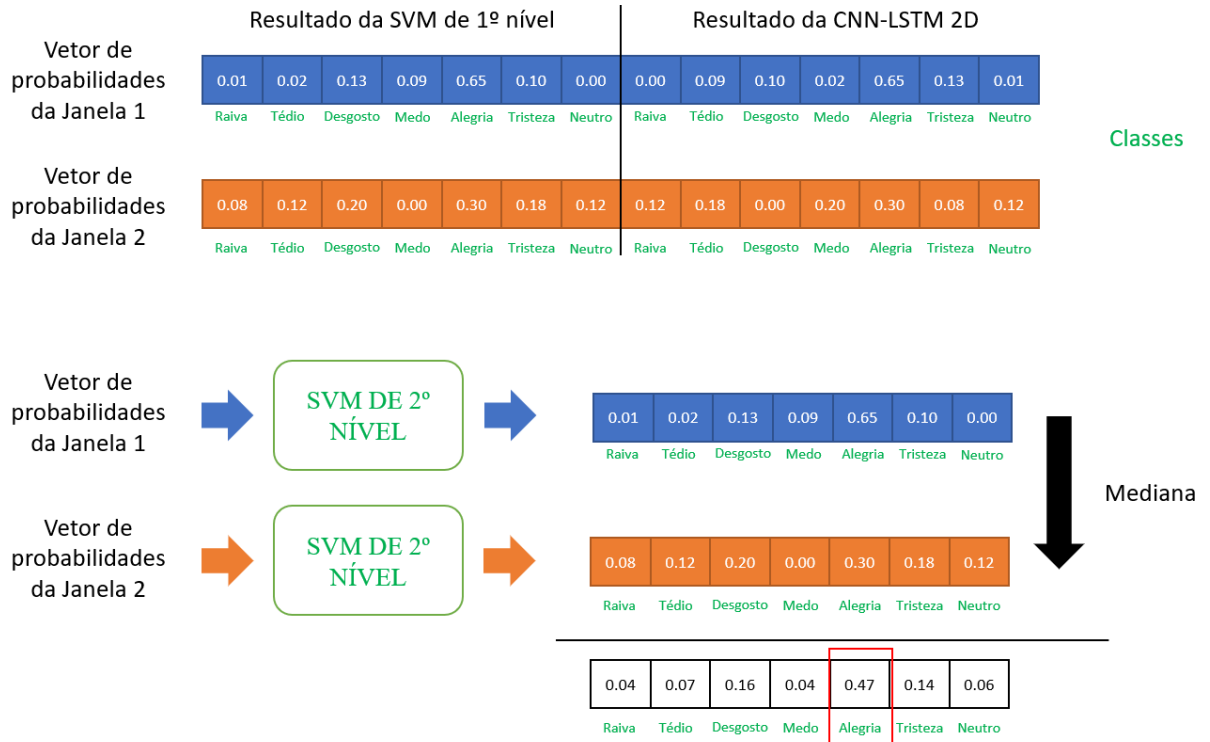
Parâmetro	Valor mínimo	Valor máximo
<i>C</i>	0	2
<i>gama</i>	0	10

Fonte: Elaborado pelo autor.

Obteve-se a saída probabilística dessa SVM para cada janela dos áudios da base de dados. Em seguida, todos os resultados das janelas de cada áudio foram agrupados e calculou-se a mediana das probabilidades de todas as janelas em relação a cada classe. Por fim, a classificação foi

realizada tomando como resposta a classe com maior valor entre as 7 medianas. A Figura 17 traz, a título de exemplo, o detalhamento completo dessas operações para um sinal de áudio completo dividido em 2 janelas.

Figura 17 – Classificação no segundo nível para um áudio segmentado em 2 janelas



Fonte: Elaborado pelo autor.

5 RESULTADOS E DISCUSSÃO

5.1 Recursos Computacionais Utilizados

Todos os experimentos foram realizados no computador pessoal do autor do trabalho, que possui as seguintes configurações: sistema operacional *Windows 10 Home*; processador *Intel Core i7-8750H*, 2,20 GHz com 6 núcleos físicos; memória RAM de 8 GB; unidade de armazenamento de 1 TB (disco rígido); placa de vídeo *Nvidia Geforce GTX 1060*, com 6 GB de memória dedicada.

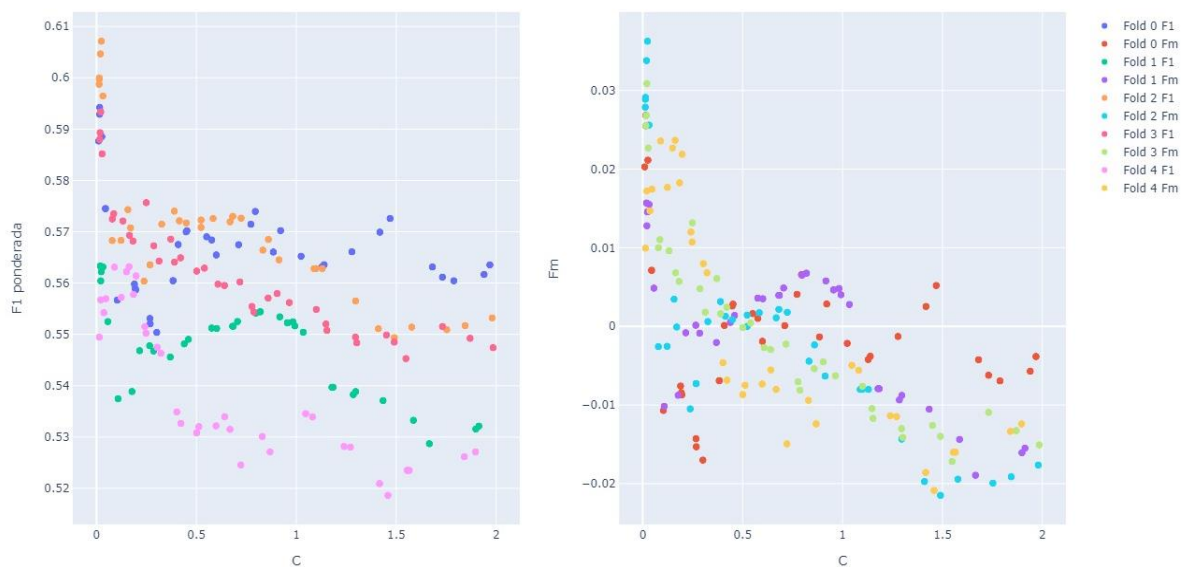
Quanto à execução de códigos de programação, optou-se pela linguagem *Python* e suas bibliotecas: *Scikit-Learn* (que contém a implementação utilizada da SVM), *Hyperopt* (que contém a implementação utilizada da otimização bayesiana com o TPE), *Keras* (que contém a implementação utilizada para CNN e LSTM), *plotly* e *matplotlib* (para gerar os gráficos apresentados).

5.2 Ajuste de Hiperparâmetros da SVM do Primeiro Nível

Tendo a segmentação dos sinais de áudio em janelas e a extração de características do sinal de voz, conforme procedimentos descritos na seção 4, iniciou-se o ajuste dos hiperparâmetros da SVM do primeiro nível. Para a verificação do desempenho do algoritmo, calcularam-se os valores F1 ponderados de acordo com a quantidade de janelas por classe. Ocorre que apenas observando o comportamento do F1 em função das escolhas de valores do hiperparâmetro não fica claro quais escolhas resultam em melhores resultados, pois os *folds* de validação de cada execução possuem dados diferentes. Isso faz com que a média do F1 das iterações de ajuste de uma execução específica seja diferente dos outros. Para uma melhor visualização, definiu-se o valor F_m , que consiste no valor de F1 após a subtração da média do F1 de todas as iterações daquela execução. Assim, o intuito é que F_m demonstre o desempenho de uma escolha de hiperparâmetros em relação à execução analisada. As Figuras 18 e 19 exemplificam o que foi dito. Verifica-se que, para o gráfico com valores de F1, a faixa de valores de C que resulta em melhores resultados fica mascarada pela diferença entre o desempenho das execuções, enquanto o gráfico com valores de F_m a torna mais visível.

No momento da escolha da faixa de valores a ser utilizada para o ajuste de cada hiperparâmetro, percebeu-se que essa tarefa não é trivial. Faixas maiores tendem a indicar o comportamento global da curva de desempenho, mas não se obtém os valores adequados de forma precisa. Por outro lado, uma faixa pequena pode revelar apenas o comportamento em torno de um mínimo local, o que também não é desejado. Uma alternativa seria permitir muitas iterações até que o algoritmo convergisse para uma faixa estreita de valores, mas isso resultaria num elevado tempo de execução. Nesse quadro, a solução encontrada foi executar duas vezes o ajuste do hiperparâmetro: primeiro com valores abrangentes, o que revela o comportamento global da curva de desempenho e, depois, com valores mais refinados, o que auxilia na detecção dos valores adequados de forma mais precisa.

Figura 18 – Ajuste global do hiperparâmetro C da SVM linear

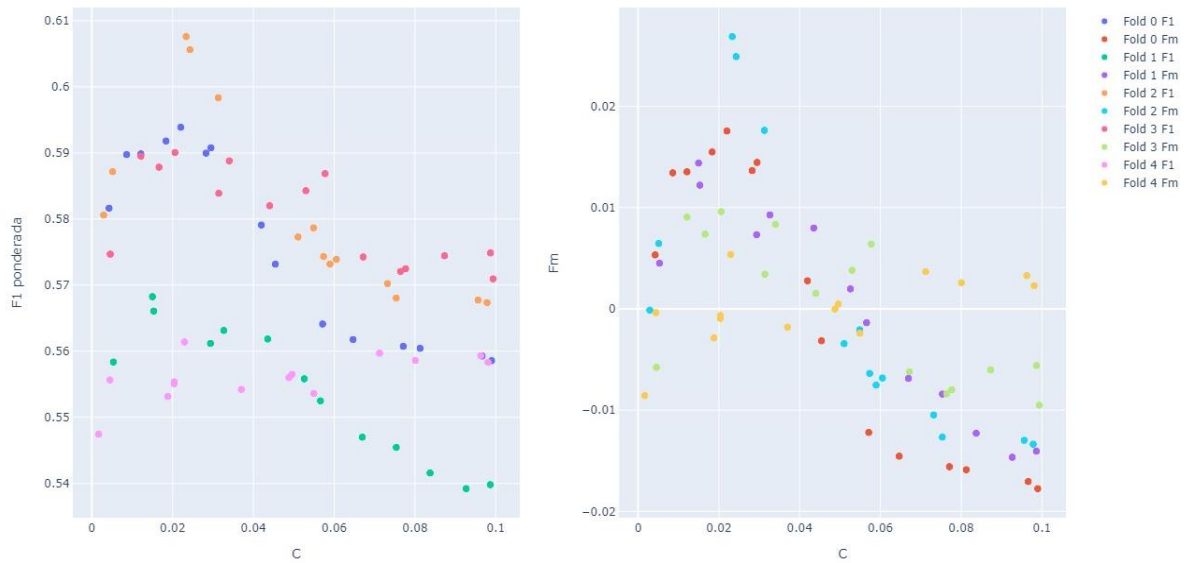


Fonte: Elaborado pelo autor.

A Figura 18 contém também a análise global do hiperparâmetro C da SVM linear, que foi realizada com 50 iterações. Em seguida, reduziram-se os valores permitidos para uma faixa menor, a fim de se obter um resultado mais preciso. Essa segunda análise está registrada na Figura 19.

A partir da interpretação da Figura 19, verifica-se que o melhor desempenho foi obtido com C entre 0,02 e 0,04, de forma que 0,03 foi escolhido como valor a ser utilizado.

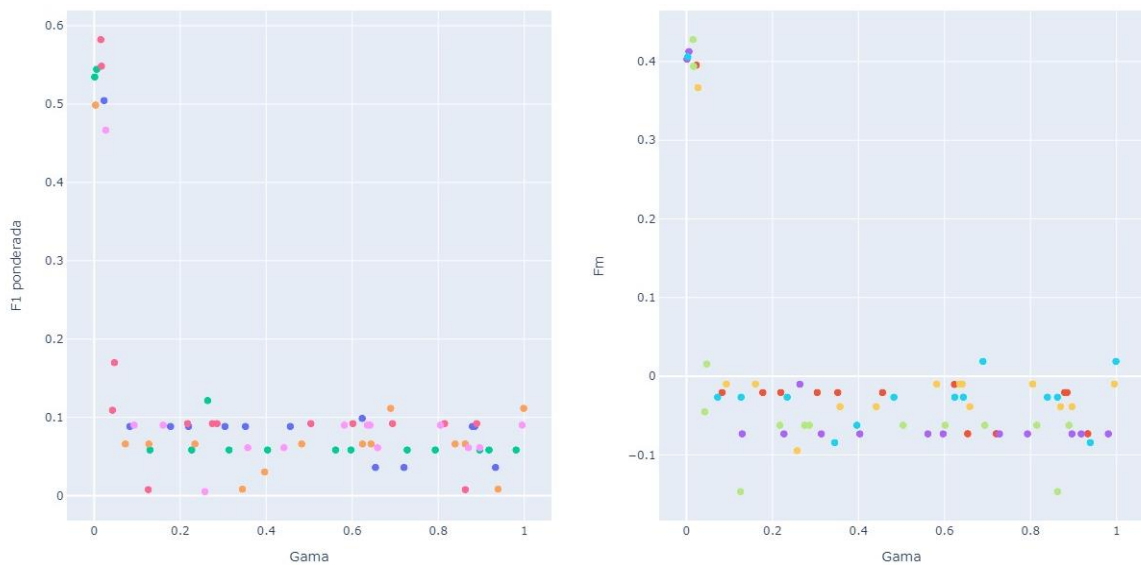
Figura 19 – Ajuste local do hiperparâmetro C da SVM linear



Fonte: Elaborado pelo autor.

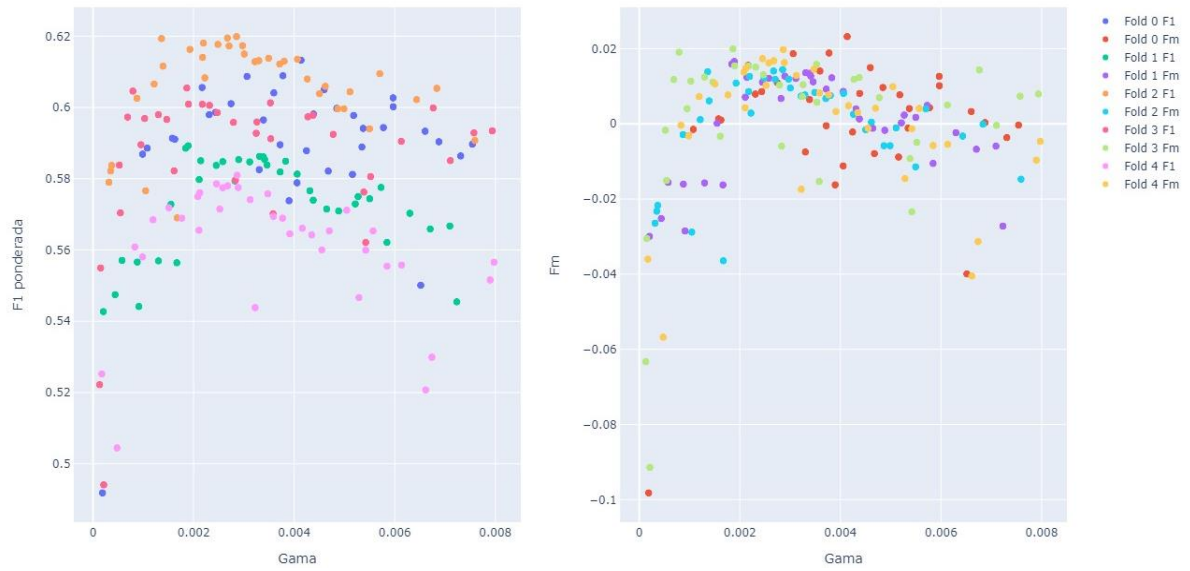
Para a SVM gaussiana, otimizam-se dois hiperparâmetros: o γ e o C . As Figuras 20 e 21 trazem análise global e local do γ , respectivamente, enquanto as Figuras 22 e 23 mostram as mesmas análises para o C . Como a análise global para o C da SVM gaussiana não foi conclusiva, a análise local foi refeita com a mesma faixa de valores, mas fixando o valor de γ .

Figura 20 – Ajuste global do parâmetro γ da SVM Gaussiana



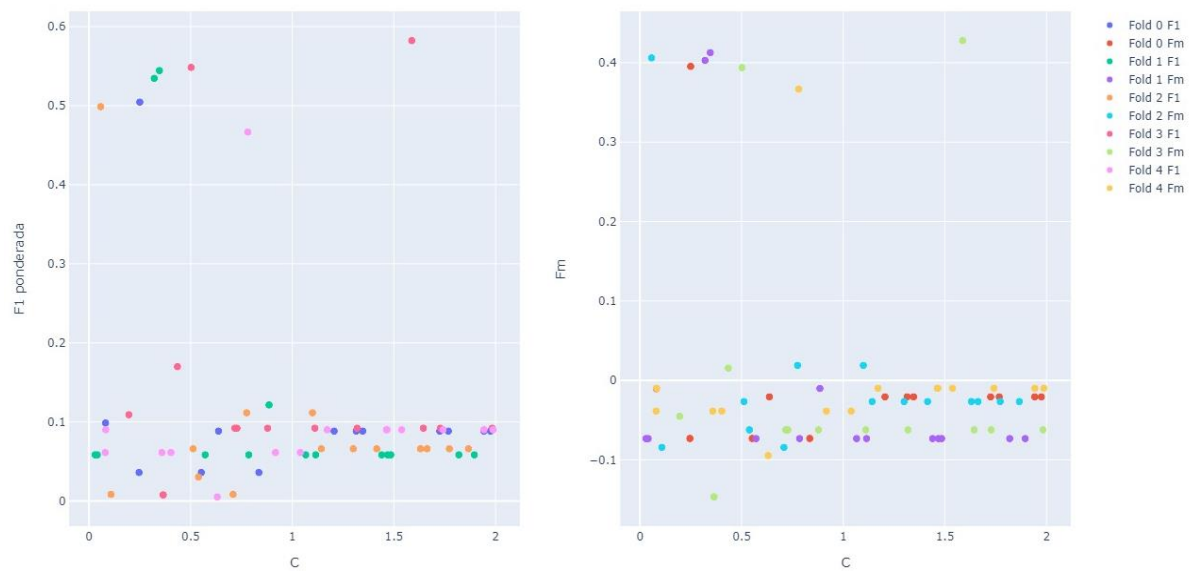
Fonte: Elaborado pelo autor.

Figura 21 – Ajuste local do parâmetro *gamma* da SVM Gaussiana



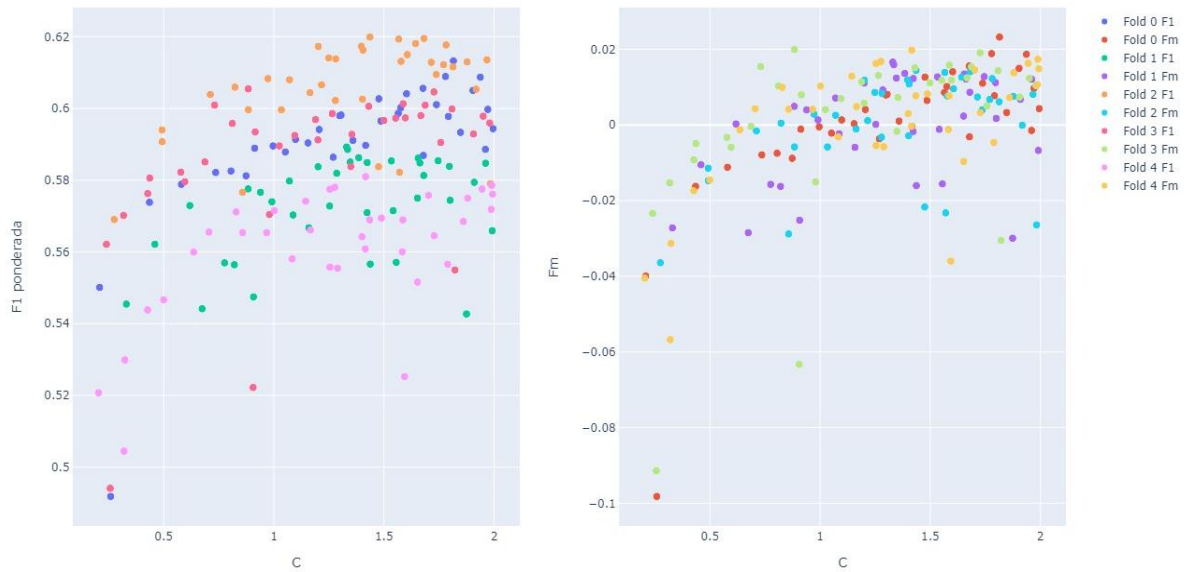
Fonte: Elaborado pelo autor.

Figura 22 – Ajuste global do parâmetro *C* da SVM Gaussiana



Fonte: Elaborado pelo autor.

Figura 23 – Ajuste local do hiperparâmetro C da SVM Gaussiana



Fonte: Elaborado pelo autor.

A partir das observações desses resultados, pode-se construir o Quadro 7 com os valores escolhidos para cada hiperparâmetro.

Quadro 7 – Valores escolhidos para os hiperparâmetros

Parâmetro	Valor obtido
C da SVM Linear	0,03
C da SVM Gaussiana	1,5
γ da SVM Gaussiana	0,005

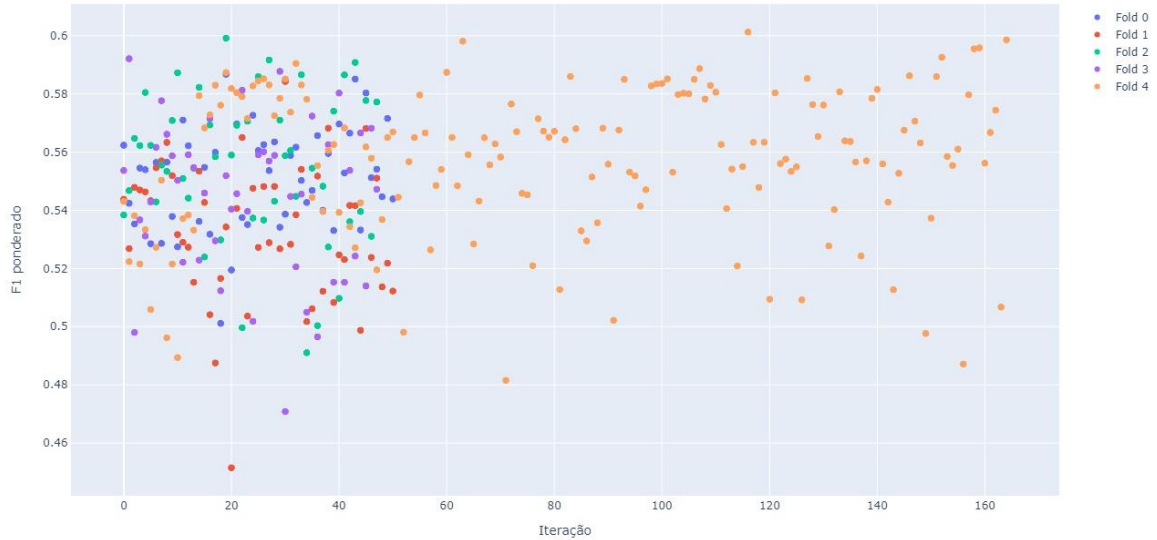
Fonte: Elaborado pelo autor.

O próximo passo consiste no ajuste da escolha das variáveis a serem utilizadas pela SVM do primeiro nível. O procedimento utilizado foi descrito na seção 4.3. As Figuras 24 e 25 contém o comportamento do valor F1 em função do número de iterações para a SVM linear e gaussiana, respectivamente.

O primeiro fato que se observa é a quantidade de iterações bem mais escassa no ajuste da SVM linear. Isso ocorreu porque ambas as otimizações das Figuras 24 e 25 foram feitas de uma só vez. Permitiu-se ao otimizador escolher ambos *kernel* e hiperparâmetros que resultassem no melhor desempenho. Entende-se, pelos valores alcançados de F1 e pela quantidade de iterações

de cada algoritmo, que a SVM gaussiana possui resultados consistentemente melhores que a linear para os 4 primeiros *fold*s de validação.

Figura 24 – Ajuste das variáveis a serem utilizadas na SVM Linear



Fonte: Elaborado pelo autor.

Figura 25 – Ajuste das variáveis a serem utilizadas na SVM Gaussiana

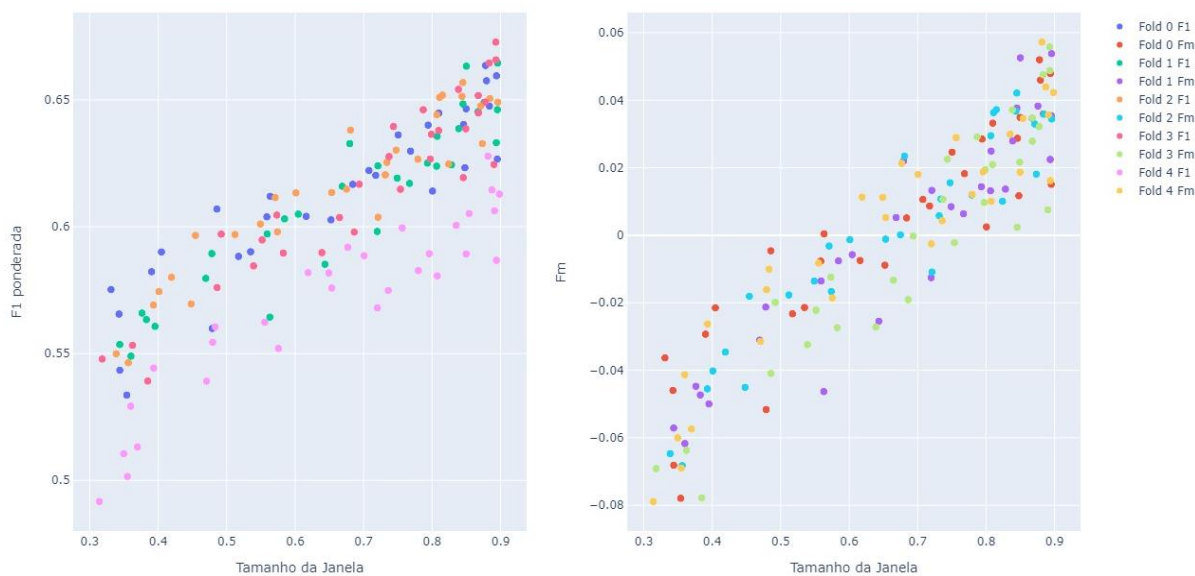


Fonte: Elaborado pelo autor.

Quanto às características selecionadas nesse processo, seu número foi reduzido de 94 para 67 e o melhor valor de F1 obtido foi de 0,648, um aumento considerável em relação aos 0,62 observados antes do ajuste da escolha das variáveis a serem utilizadas.

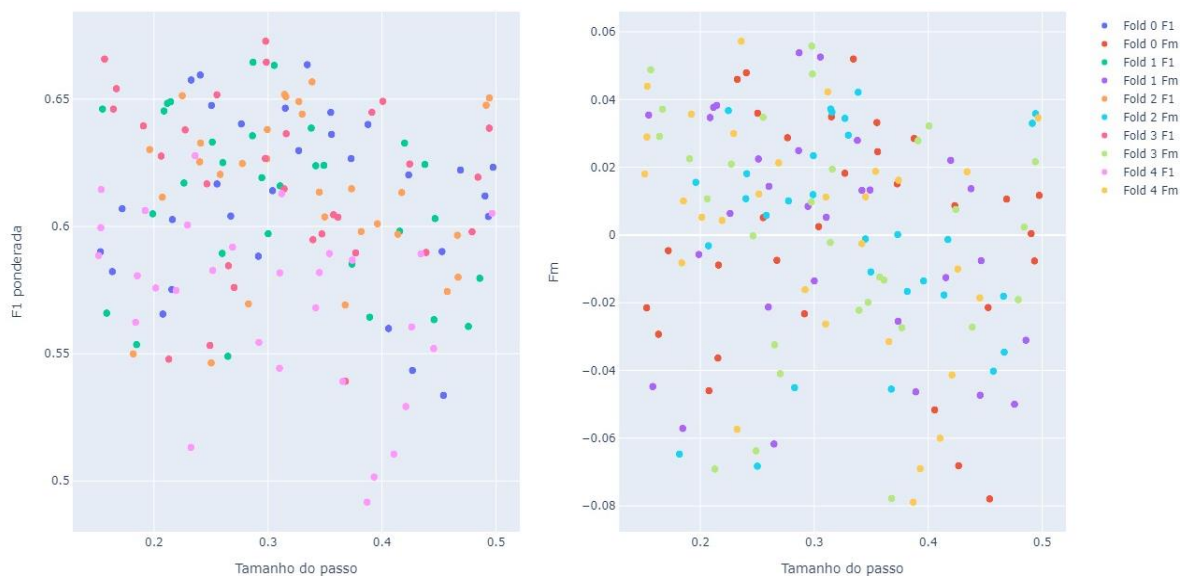
A próxima etapa foi o ajuste conjunto do tamanho da janela e do tamanho do passo da janela. Como dito na seção 4.3, a partir desse ponto, apenas a SVM gaussiana foi avaliada. As Figuras 26 e 27 informam os resultados obtidos.

Figura 26 – Ajuste do tamanho da janela de áudio



Fonte: Elaborado pelo autor.

Figura 27 – Primeiro ajuste do tamanho do passo da janela

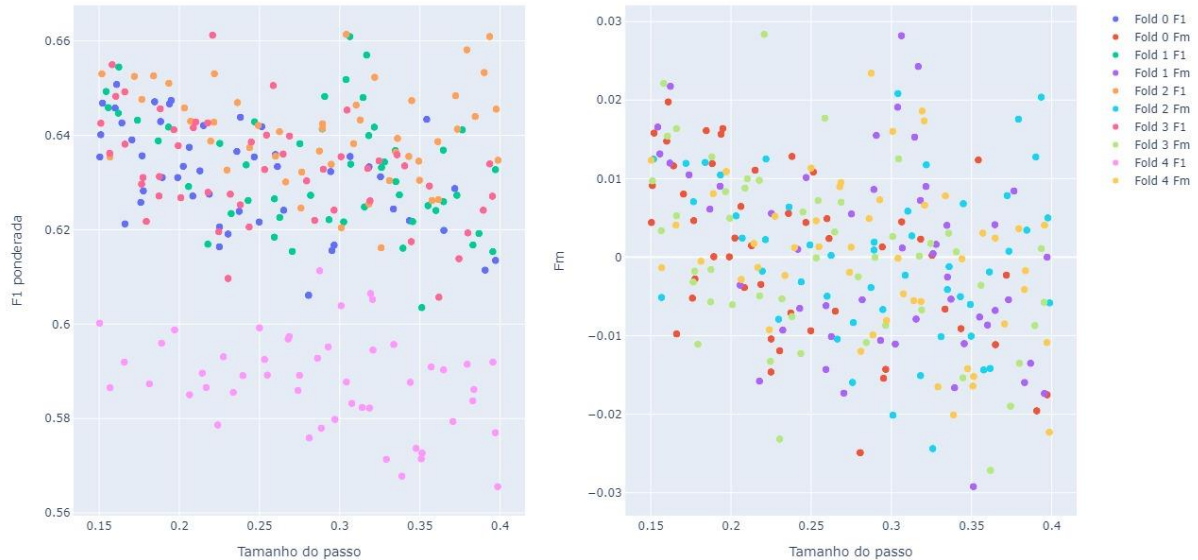


Fonte: Elaborado pelo autor.

Uma característica que fica evidente é a preferência do algoritmo por tamanhos maiores de janela de áudio. A fim de garantir que todos os áudios possuíssem pelo menos 2 janelas, fez-se necessário limitar o tamanho da janela a 0,85 s, valor esse que foi a escolha final. Por sua vez,

o tamanho do passo da janela não demonstrou ter relevância significativa se comparada ao tamanho da janela e, por isso, a etapa seguinte de ajuste considera apenas variações nesse hiperparâmetro, como demonstrado na Figura 28.

Figura 28 – Segundo ajuste do tamanho do passo da janela



Fonte: Elaborado pelo autor.

Como se pode notar, duas faixas de valores de tamanho do passo da janela podem ser escolhidas: ou os valores em torno de 0,3 s ou valores próximos a 0,15 s. Decidiu-se pelo valor de 0,15 s em função da maior consistência de resultados no seu entorno. Dessa forma, todos os hiperparâmetros da SVM do primeiro nível foram ajustados. A configuração final se encontra no Quadro 8.

Quadro 8 – Resultado do ajuste de hiperparâmetros da SVM de primeiro nível

Parâmetro	Valor obtido
<i>kernel</i>	Gaussiano
<i>C</i>	1,5
<i>gama</i>	0,005
Tamanho da janela de áudio	0,85 s

Fonte: Elaborado pelo autor.

Nesse momento, realizaram-se testes a fim de verificar o desempenho obtido quando do agrupamento de todas as janelas de cada áudio para a classificação do conjunto de validação. O resultado está disposto no Quadro 9.

Quadro 9 – Desempenho da SVM ajustada sobre o conjunto de validação

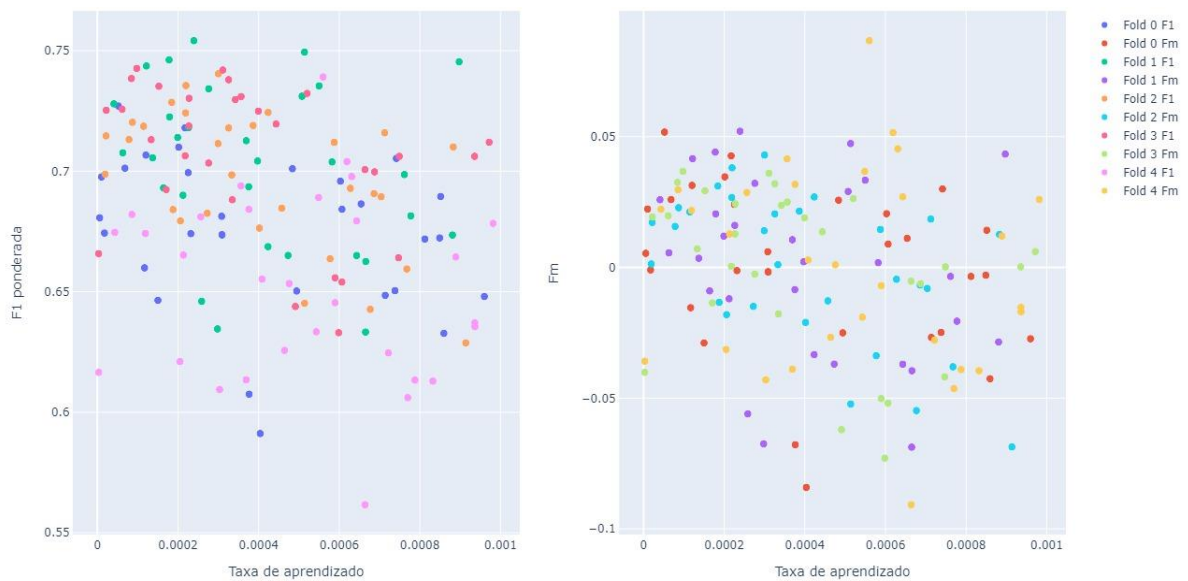
<i>Fold</i> de validação	F1 ponderado
0	0,67
1	0,70
2	0,72
3	0,69
4	0,69

Fonte: Elaborado pelo autor.

5.3 Ajuste dos Hiperparâmetros da CNN-LSTM-2D

No caso da CNN-LSTM-2D, o único parâmetro a ser ajustado é a taxa de aprendizado, posto que a arquitetura foi baseada em Gering (2019) e em Zhao, Mao e Chen (2019). A Figura 29 traz o ajuste da taxa de aprendizado.

Figura 29 – Ajuste da taxa de aprendizado da CNN-LSTM-2D



Fonte: Elaborado pelo autor.

A primeira observação a se realizar é a faixa de valores F1 obtidos durante o ajuste. Eles são bem superiores aos da SVM para todos os *fold*s de validação. Porém, cabe ficar atento ao fato de que, em cada *fold*, um valor distinto de taxa de aprendizado é mais eficiente e, ao se escolher um único valor, ocorrerá um decréscimo de desempenho. Pode-se avaliar também que os resultados mais consistentes ficam entre 2×10^{-4} e 4×10^{-4} . Dessa maneira, escolheu-se o valor de 3×10^{-4} como a taxa de aprendizado a ser utilizada. Essa informação está também representada no Quadro 10.

Quadro 10 – Resultado da otimização de hiperparâmetros da CNN-LSTM-2D

Parâmetro	Valor obtido
Taxa de aprendizado	3×10^{-4}

Fonte: Elaborado pelo autor.

Para verificar o desempenho da rede ajustada, ela foi executada 5 vezes nos conjuntos de validação de cada execução para que o valor médio e desvio padrão de F1 fossem anotados. O valor da média está registrado na frente, seguido pelo desvio padrão entre parênteses, como se encontra no Quadro 11. Destaca-se que o procedimento foi o mesmo realizado para a SVM: agrupam-se todas as janelas do mesmo áudio para obter, através da mediana das probabilidades de cada janela, a classe final.

Quadro 11 – Desempenho da CNN-LSTM 2D ajustada no conjunto de validação

<i>Fold</i> de validação	F1 ponderado
0	0,64 (0,05)
1	0,71 (0,10)
2	0,76 (0,03)
3	0,64 (0,01)
4	0,76 (0,14)

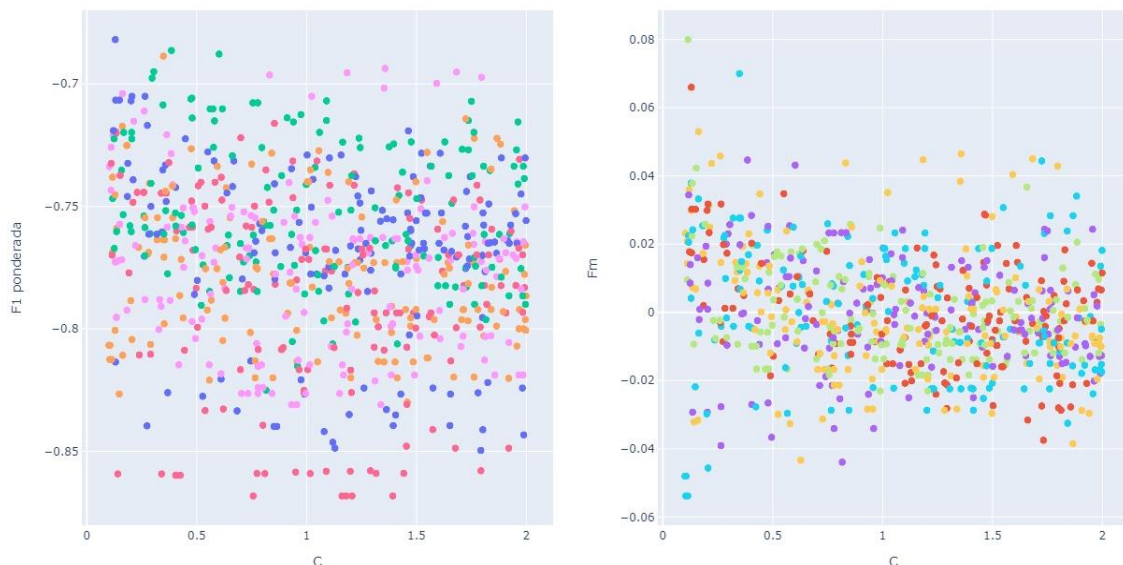
Fonte: Elaborado pelo autor.

5.4 Ajuste da SVM do Segundo Nível

No caso da SVM do segundo nível, escolheu-se otimizar apenas o *kernel* gaussiano. Dessa forma, o procedimento foi o ajuste dos parâmetros C e γ , conforme as Figuras 30 e 31, respectivamente.

Observa-se que C não demonstrou possuir uma faixa de valores que resulte em melhorias consistentes de desempenho. Dessa maneira, escolheu-se com base na repetibilidade, ou seja, muitas iterações de execuções diferentes resultando em valores próximos de F_m . As escolhas feitas estão na Quadro 12.

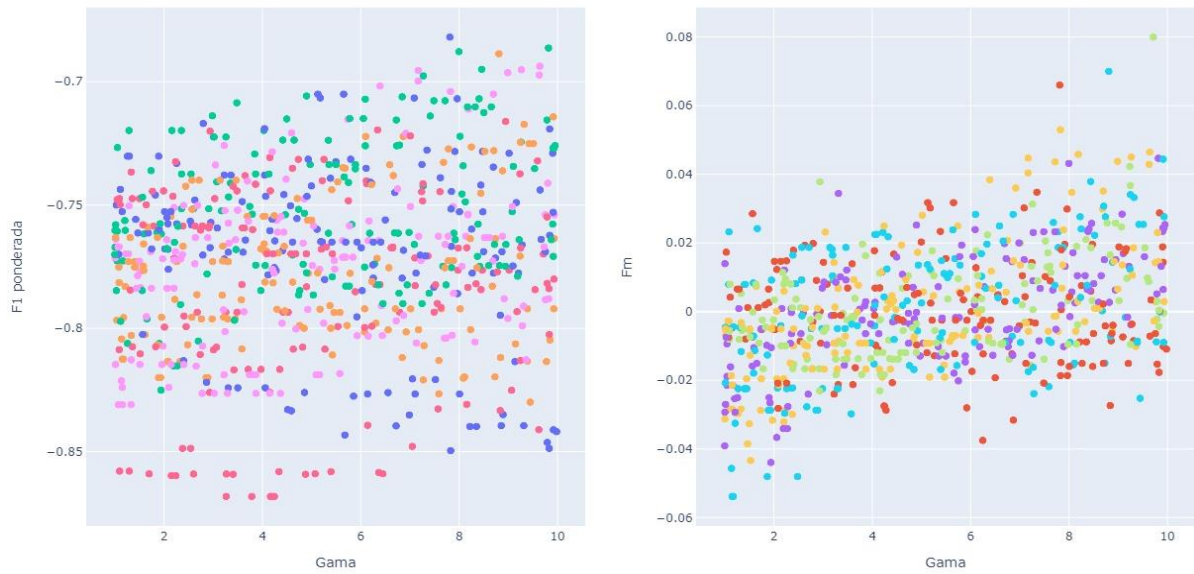
Figura 30 – Ajuste do parâmetro C da SVM de segundo nível



Fonte: Elaborado pelo autor.

Para verificar o desempenho do sistema ajustado, realizaram-se 5 execuções para cada *fold* de teste. A média e o desvio padrão dos desempenhos obtidos estão no Quadro 13. Destaca-se que, a partir desse momento, todos os resultados mostrados foram aferidos sobre o conjunto de teste, visando obter os desempenhos finais. É interessante também avaliar como o algoritmo se comporta em relação a cada classe. Como várias iterações foram realizadas, o Quadro 14 traz a média e o desvio padrão do F1 por classe.

Figura 31 – Ajuste do parâmetro *gamma* da SVM de segundo nível



Fonte: Elaborado pelo autor.

Quadro 12 – Resultado do ajuste da SVM do segundo nível

<i>Parâmetro</i>	Valor obtido
<i>C</i>	0,7
<i>gamma</i>	8,0

Fonte: Elaborado pelo autor.

Quadro 13 – Desempenho do algoritmo completo ajustado nos conjuntos de teste

<i>Fold</i> de teste	F1 ponderado
0	0,90 (0,01)
1	0,84 (0,03)
2	0,85 (0,03)
3	0,85 (0,02)
4	0,90 (0,02)

Fonte: Elaborado pelo autor.

Quadro 14 – Desempenho por classe do algoritmo completo

Classe	F1
Raiva	0,92 (0,03)
Tédio	0,84 (0,06)
Neutro	0,88 (0,07)
Alegria	0,80 (0,05)
Medo	0,86 (0,07)
Tristeza	0,92 (0,06)
Desgosto	0,82 (0,04)

Fonte: Elaborado pelo autor.

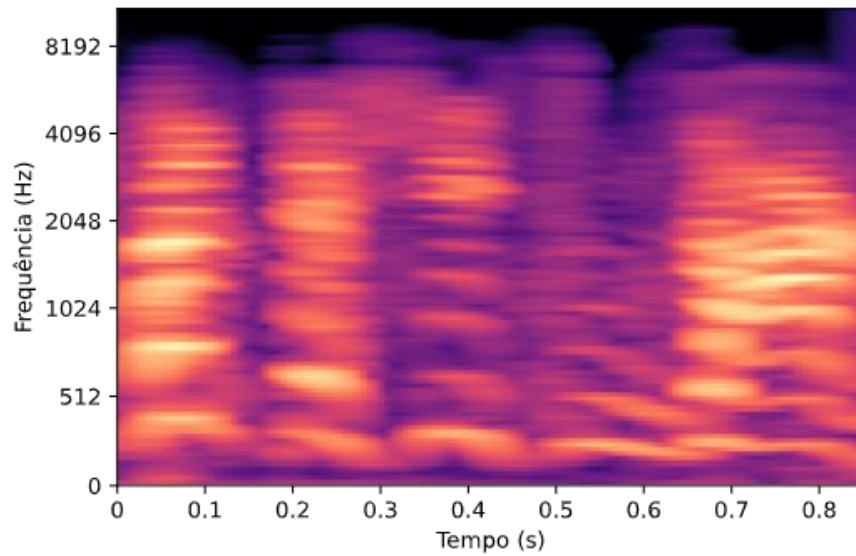
Cabe salientar, nesse contexto, que a abordagem proposta para a otimização de hiperparâmetros foi a análise isolada dos hiperparâmetros de forma sequencial. Isso pode ter influenciado na escolha dos hiperparâmetros que foram ajustados por último e, conseqüentemente, resultado em um decréscimo de desempenho. Uma abordagem de ajuste conjunto de todos os hiperparâmetros seria ideal para garantir a eliminação desse efeito, mas para o presente trabalho, seria necessário possuir maior capacidade de computação e mais tempo disponível para a execução dos algoritmos.

5.5 Aumento de Dados

O aumento de dados foi feito conforme os procedimentos da seção 4.6. Para se ter uma melhor visualização do efeito das transformações realizadas, as Figuras 32, 33, 34 e 35 trazem o Mel espectrograma de uma janela sem adição de ruído, com adição de ruído, com diminuição na velocidade do áudio e com aumento na velocidade do áudio, respectivamente.

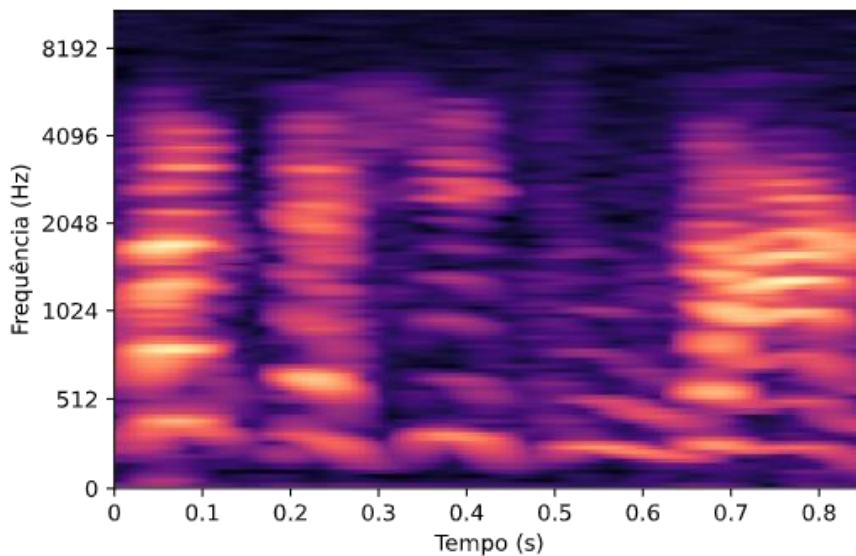
Pode-se notar que a adição do ruído gaussiano branco diminui a clareza das características de frequência do áudio, enquanto a alteração da velocidade do áudio ou expande ou contrai o espectrograma em relação ao eixo do tempo. Em posse desses resultados, testou-se o desempenho de cada algoritmo quando da utilização de cada um dos métodos e da combinação de ambos. O Quadro 15 compara os desempenhos do aumento de dados na SVM do primeiro nível, o Quadro 16 compara os desempenhos na CNN-LSTM 2D e o Quadro 17 mostra os desempenhos do arranjo de classificadores.

Figura 32 – Mel espectrograma original de uma janela de áudio



Fonte: Elaborado pelo autor.

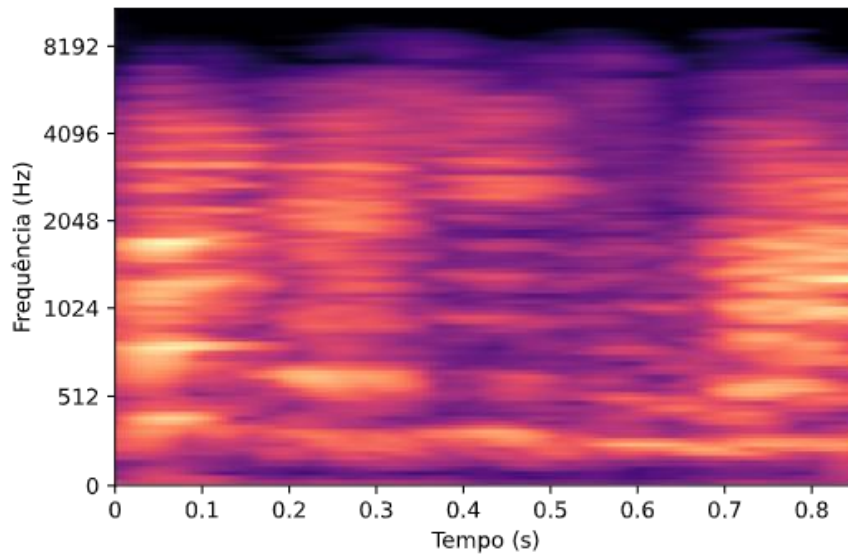
Figura 33 – Mel espectrograma de uma janela de áudio com adição de ruído



Fonte: Elaborado pelo autor.

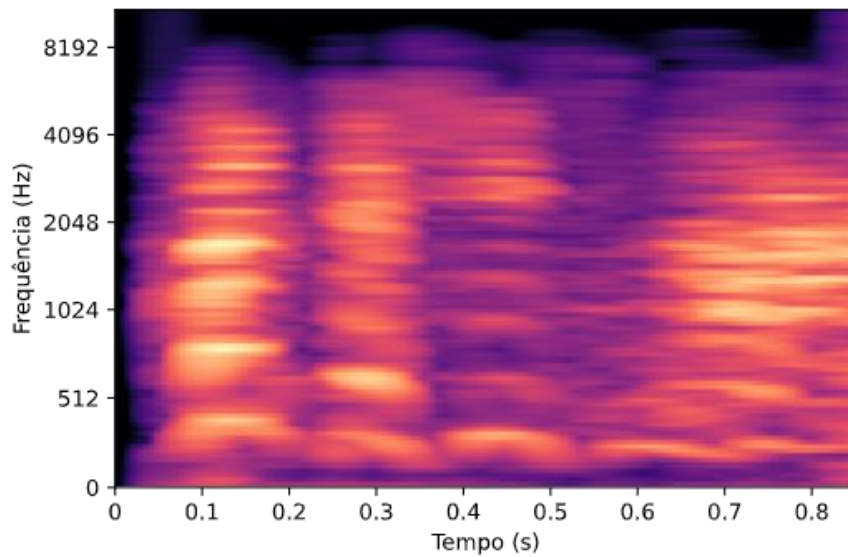
Avaliando os resultados dos Quadros 15, 16 e 17, verifica-se que, para a SVM do primeiro nível e para a CNN-LSTM 2D, a melhor técnica foi utilizar apenas a adição do ruído. Como indicado anteriormente na Figura 33, o ponto positivo da adição do ruído é que ele mascara as características de frequência do sinal de áudio que possuem menor amplitude. Isso pode ter colaborado para uma maior capacidade de generalização dos algoritmos.

Figura 34 – Mel espectrograma de uma janela de áudio com diminuição de velocidade



Fonte: Elaborado pelo autor.

Figura 35 – Mel espectrograma de uma janela de áudio com aumento de velocidade



Fonte: Elaborado pelo autor.

Por sua vez, o arranjo completo dos classificadores não observou melhoria expressiva de desempenho quando da aplicação do aumento de dados. Um raciocínio plausível é o de que, por mais que tenha ocorrido alguma melhoria no desempenho, ela foi menor que o desvio padrão do desempenho dos algoritmos do primeiro nível, o que faz com que o ganho de desempenho global seja inexpressivo.

Quadro 15 – Resultado da aplicação do aumento de dados na SVM de primeiro nível analisada nos folds de teste

<i>Fold</i>	F1 anterior	F1 com ruído	F1 com alteração da velocidade	F1 com as duas técnicas
0	0,67	0,72 (0,01)	0,68 (0,02)	0,71 (0,02)
1	0,70	0,67 (0,01)	0,71 (0,02)	0,70 (0,03)
2	0,72	0,72 (0,01)	0,72 (0,01)	0,70 (0,03)
3	0,69	0,73 (0,02)	0,72 (0,02)	0,71 (0,01)
4	0,69	0,71 (0,01)	0,67 (0,01)	0,70 (0,02)

Fonte: Elaborado pelo autor.

Quadro 16 – Resultado da aplicação do aumento de dados na CNN-LSTM 2D analisada nos folds de teste

<i>Fold</i>	F1 anterior	F1 com ruído	F1 com alteração da velocidade	F1 com as duas técnicas
0	0,64 (0,05)	0,73 (0,04)	0,71 (0,04)	0,71 (0,08)
1	0,71 (0,10)	0,76 (0,05)	0,73 (0,03)	0,76 (0,02)
2	0,76 (0,03)	0,75 (0,06)	0,66 (0,09)	0,75 (0,05)
3	0,64 (0,02)	0,74 (0,10)	0,74 (0,06)	0,68 (0,02)
4	0,76 (0,01)	0,72 (0,07)	0,67 (0,05)	0,70 (0,07)

Fonte: Elaborado pelo autor.

Quadro 17 – Resultado da aplicação do aumento de dados no arranjo de classificadores

<i>Fold</i>	F1 anterior	F1 com ruído	F1 com alteração da velocidade	F1 com as duas técnicas
0	0,90 (0,01)	0,90 (0,01)	0,91 (0,02)	0,90 (0,03)
1	0,84 (0,03)	0,86 (0,02)	0,86 (0,02)	0,87 (0,02)
2	0,85 (0,03)	0,79 (0,02)	0,84 (0,02)	0,85 (0,03)
3	0,85 (0,02)	0,87 (0,03)	0,88 (0,02)	0,86 (0,02)
4	0,90 (0,02)	0,90 (0,03)	0,90 (0,01)	0,90 (0,02)

Fonte: Elaborado pelo autor.

5.6 Comparação de Resultados

Com o intuito de comparar os resultados obtidos aos demais presentes no ramo de SER, construiu-se o Quadro 18, considerando como resultado do presente trabalho o arranjo de classificadores sem a utilização do aumento de dados.

A partir dessa comparação, nota-se que houve melhoria no desempenho de classificação do modelo proposto em relação a Gering, Ciarelli e Salles (2019). Ainda que para algumas classes o F1 tenha sido inferior, o valor F1 médio foi superior. Convém destacar também que o arranjo proposto em Gering, Ciarelli e Salles (2019) foi de maior complexidade, o que também ressalta o benefício da metodologia proposta.

Quadro 18 – Comparação do F1 obtido no modelo proposto com resultados de outros trabalhos

Classe	Proposto	Gering, Ciarelli e Salles	Zhao, Mao e Chen
Raiva	0,92	0,88	0,95
Tédio	0,84	0,92	0,94
Neutro	0,88	0,89	0,92
Alegria	0,80	0,88	0,93
Medo	0,86	0,77	0,96
Tristeza	0,92	1,00	0,99
Desgosto	0,82	0,50	0,96
Média	0,86	0,83	0,95

Fonte: Elaborado pelo autor.

Como a principal diferença entre a abordagem proposta nesse trabalho e a proposta por Zhao, Mao e Chen (2019) é a realização de *padding* do sinal, foi feita uma breve análise de como esse procedimento pode ter ajudado na classificação. Conforme pode ser observado no Quadro 19, os tempos médios e os desvios padrão de duração dos áudios de cada classe são ligeiramente diferentes uns dos outros, com destaque para as classes Tristeza e Desgosto. É possível que o preenchimento com zeros para completar um áudio de 8 segundos possa ter sido uma fonte de informação indireta para a rede profunda, que pode ter aprendido algum padrão por meio do número de zeros utilizado no preenchimento do áudio. Entretanto, uma análise mais aprofundada desses pontos foge ao escopo do presente trabalho, sendo sugerido esse estudo para trabalhos futuros.

Quadro 19 – Média e desvio padrão da duração dos áudios de cada classe

Classe	Média e desvio padrão da duração dos áudios
Raiva	2,64 (0,73)
Tédio	2,78 (0,80)
Neutro	2,36 (0,65)
Alegria	2,54 (0,68)
Medo	2,23 (0,63)
Tristeza	4,05 (1,52)
Desgosto	3,35 (1,06)

Fonte: Elaborado pelo autor.

6 CONCLUSÕES E PROJETOS FUTUROS

6.1 Conclusões

O objetivo desse trabalho foi propor um arranjo de classificadores para identificação de sentimento em voz, com base no trabalho de Gering, Ciarelli e Salles (2019), dando foco ao ajuste de hiperparâmetros e uso de técnicas de aumento de dados para equalização da base de dados. Para isso, utilizou-se a base de dados Berlin, que é de domínio público. A abordagem proposta se baseou em algoritmos de otimização bayesiana, no caso do ajuste de hiperparâmetros, assim como adição de ruído gaussiano branco e alteração na velocidade do sinal de voz para o caso de aumento de dados.

Os objetivos propostos foram alcançados, com destaque para a melhoria no desempenho F1 em relação aos resultados alcançados em Gering, Ciarelli e Salles (2019), mesmo com a proposição de um arranjo de classificadores mais simples do que o do artigo base. Verificou-se também que a utilização do arranjo de classificadores trouxe melhora significativa dos resultados quando comparada com a aplicação de apenas um classificador isoladamente. A otimização bayesiana, por sua vez, se mostrou uma efetiva forma de se encontrar valores adequados para os hiperparâmetros dos algoritmos estudados. No que tange ao uso do aumento de dados para a equalização da base de dados, não se observou melhoria significativa no desempenho da arquitetura implementada, de forma que o melhor resultado considerado foi obtido sem o uso do aumento de dados.

6.2 Sugestões para Projetos Futuros

Como trabalhos futuros, sugere-se implementar o ajuste de todos os hiperparâmetros de forma conjunta para evitar que o ajuste isolado dos primeiros hiperparâmetros influencie nos valores obtidos para os seguintes. Além disso, como os métodos de aumento de dados não resultaram em melhorias significativas de desempenho, podem-se avaliar outros métodos para lidar com bases de dados desbalanceadas, como técnicas de sobreamostragem ou subamostragem, que também são observadas em trabalhos científicos na área de SER. Por fim, verificou-se que a metodologia proposta por Zhao, Mao e Chen (2019) para a CNN-LSTM 2D obteve melhores resultados, mas o método de *padding* utilizado pelos autores pode ter influenciado no

aprendizado de características específicas da base de dados Berlin, mais especificamente, na diferença da duração dos áudios de cada classe. Assim, sugere-se a análise desse fenômeno para verificar se o método proposto por esses autores seria capaz de obter bons resultados para outras bases de dados.

REFERÊNCIAS BIBLIOGRÁFICAS

- ALDENEH, Z.; PROVOST M. E. Using regional saliency for speech emotion recognition. *In: INTERNATIONAL CONFERENCE ON ACOUSTICS SPEECH AND SIGNAL PROCESSING, 2017, New Orleans. **Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2017)***. New Orleans: IEEE, 2017. p. 2741-2745.
- AYADI, M. E.; KAMEL, M. S.; KARRAY, F. Survey on speech emotion recognition: Features, classification schemes, and databases. **Pattern Recognition**. v. 44, n. 3, p. 572-587. set. 2010.
- BADSHAH, A. M; AHMAD, J; RAHIM, N; BAIK, S. W. Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network. *In: INTERNATIONAL CONFERENCE ON PLATFORM TECHNOLOGY AND SERVICE (PLATCON), 2017, Busan. **Proceedings** [...]*. Busan: IEEE, 2017. p. 1-5.
- BERGSTRA, J.; BENGIO, Y. Random Search for Hyper-Parameter Optimization. **Journal of Machine Learning Research**, v. 13, n. 2, p. 281-305, fev. 2012.
- BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; BALÁZS, K. Algorithms for Hyper-Parameter Optimization. *In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 24., 2011, Red Hook. **Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)***. Red Hook: Curran Associates Inc., 2011. p. 2546-2554
- BISHOP, C. M. **Pattern Recognition and Machine Learning**. 1. ed. New York: Springer, 2006.
- BURKHARDT, F.; PAESCHKE, A.; ROLFES, M.; SENDLMEIER, W.; WEISS, B. A Database of German Emotional Speech. *In: EUROPEAN CONFERENCE ON SPEECH COMMUNICATION AND TECHNOLOGY, 9., 2005, Lisbon. **Proceedings of Interspeech 2005 – Eurospeech, 9th European Conference on Speech Communication and Technology***. Lisbon: ISCA, 2005. p. 1517-1520.
- CHANG, C. C.; LIN, C. J. LIBSVM: A Library for Support Vector Machines. **ACM Transactions on Intelligent Systems and Technology**, New York, v. 2, n. 3, p. 1-27, abr. 2011.
- CHATZIAGAPI, A.; PARASKEVOPOULOS, G.; SGOUROPOULOS, D.; PANTAZOPOULOS, G.; NIKANDROU, M.; GIANNAKOPOULOS, T; KATSAMANIS, A.; POTAMIANOS, A.; NARAYANAN, S. Data Augmentation using GANs for Speech Emotion Recognition. *In: INTERSPEECH, 2019, Graz. **Proceedings of Interspeech 2019***. Graz: ISCA, 2019. p. 171-175.
- DE CHEVEIGNÉ, A.; KAWAHARA, H. YIN, a fundamental frequency estimator for speech and music. **The Journal of the Acoustical Society of America**, v. 111, n. 4, p. 1917-1930, abr. 2002.

ERAY, O.; TOKAT, S.; IPLIKCI, S. An application of speech recognition with support vector machines. *In: INTERNATIONAL SYMPOSIUM ON DIGITAL FORENSIC AND SECURITY (ISDFS)*, 6., 2018, Antalya. **Proceedings** [...]. Antalya: IEEE, 2018. p. 1-6.

FUKUDA, T.; FERNANDEZ, R.; ROSENBERG, A.; THOMAS, S.; RAMABHADRAN, B.; SORIN, A.; KURATA, G. Data Augmentation Improves Recognition of Foreign Accented Speech. *In: INTERSPEECH*, 2018, Hyderabad. **Proceedings of Interspeech 2018**. Hyderabad: ISCA, 2018, p. 2409-2413.

GERING, G. B. S. **Identificação de sentimento em voz por meio da combinação de classificações intermediárias dos sinais em excitação, valência e quadrante**. 2019. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Espírito Santo, Vitória, 2019.

GERING, G.; CIARELLI, P.; SALLES, E. Identificação de sentimento em voz por meio da combinação de classificações intermediárias dos sinais em excitação, valência e quadrante. *In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO APLICADA À SAÚDE (SBCAS)*, 19., 2019, Niterói. **Anais do XIX Simpósio Brasileiro de Computação Aplicada à Saúde**. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 152-163.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. 1. ed. Cambridge: MIT Press, 2016.

GRAVES, A.; JAITLY N. Towards End-to-End Speech Recognition with Recurrent Neural Networks. *In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, 31., 2014, Beijing. **Proceedings of the 31st International Conference on Machine Learning**. Beijing: JMLR: W&CP, 2014. p. 1764-1772.

HEIDER, F. **The Psychology of Interpersonal Relations**. 3. ed. New York: John Wiley & Sons, Inc., 1958.

HIRSCHBERG, J.; MANNING, C. Advances in natural language processing. **Science**, v. 349, n. 6245, p. 261-266, jul. 2015.

LORENA, A.; CARVALHO, A; Uma introdução às Support Vector Machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43-67, fev. 2007.

MOUNTASSIR, A.; BENBRAHIM, H.; BERRADA, I. An empirical study to address the problem of Unbalanced Data Sets in sentiment classification. *In: INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC)*, 2012, Seoul. **Proceedings** [...]. Seoul: IEEE, 2012. p. 3298-3303.

OLAH, C. **Understanding LSTM Networks**. Github, 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 16 nov. 2020.

PATHAK, S.; KOLHE, V. Emotion Recognition from Speech Signals Using Deep Learning Models. **Imperial Journey of Interdisciplinary Research**, v. 2, n. 9, p. 19-24, set. 2016.

PLATT, J. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *In: SMOLA, A.; BARTLETT, P.; SCHÖLKOPF, B.; SCHUURMANS, D. **Advances in Large-Margin Classifiers***. 1. ed. London: MIT Press, 1999. p. 61-74.

REDDY, A.; VIJAYARAJAN, V. Extraction of Emotions from Speech – A Survey. **International Journey of Applied Engineering Research**, v. 12, n. 16, p. 5760-5767, set. 2017.

RUSSELL, J. A Circumplex Model of Affect. **Journal of Personality and Social Psychology**, v. 29, n. 6, p. 1161-1178, jun. 1980.

SHEN, P.; CHANGJUN, Z.; CHEN, X. Automatic Speech Emotion Recognition using Support Vector Machine. *In: INTERNATIONAL CONFERENCE ON ELECTRONIC & MECHANICAL ENGINEERING AND INFORMATION TECHNOLOGY, 2011, Harbin. **Proceedings of 2011 International Conference on Electronic & Mechanical Engineering and Information Technology***. Harbin: IEEE, 2011. p. 621-625.

STEVENS, S. S.; VOLKMANN J. A Scale for the Measurement of the Psychological Magnitude Pitch. **The Journal of the Acoustical Society of America**, v. 8, n. 185, p. 185-190, jun. 1937.

VANNUCCI, M.; COLLA, V. Genetic Algorithms Based Resampling for the Classification of Unbalanced Datasets. *In: INTERNATIONAL CONFERENCE ON INTELLIGENT DECISION TECHNOLOGIES, 9., 2017, Vilamoura. **Proceedings of the 9th KES International Conference on Intelligent Decision Technologies (KES-IDT 2017) - Part II***. Vilamoura: Springer, 2017. p. 23-32.

WU, T. F.; LIN, C. J.; WENG, R. C. Probability Estimates for Multi-class Classification by Pairwise Coupling. **The Journal of Machine Learning Research**, v. 5, p. 975-1005, ago. 2004.

XIA, R.; LIU, Y. A Multi-Task Learning Framework for Emotion Recognition Using 2D Continuous Space. **IEEE Transactions on Affective Computing**, v. 8, n. 1, p. 3-14. jan-mar 2017.

ZHAO, J.; MAO, X.; CHEN, L. Speech emotion recognition using deep 1D & 2D CNN networks. **Biomedical Signal Processing and Control**, v. 47, n. 1, p. 312-323, jan. 2019.

ZHU, X.; LIU, Y.; LI, J.; WAN, T.; QIN, Z. Emotion Classification with Data Augmentation Using Generative Adversarial Advances in Knowledge Discovery and Data Mining. *In: PACIFIC-ASIA CONFERENCE, 22., 2018, Melbourne. **Proceedings** [...]*. Melbourne: Springer, 2018. p. 349-360