UNIVERSIDADE FEDERAL DO ABC

Trabalho de Graduação III

EXTRAÇÃO DE SINAIS POR MEIO DE MASCARAMENTO TEMPO-FREQUENCIAL E TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

MATEUS DOS SANTOS MOURA

ORIENTAÇÃO:
RICARDO SUYAMA

Santo André
Dezembro de 2023

Universidade Federal do ABC

MATEUS DOS SANTOS MOURA

ORIENTAÇÃO:
RICARDO SUYAMA

EXTRAÇÃO DE SINAIS POR MEIO DE MASCARAMENTO TEMPO-FREQUENCIAL E TÉCNICAS DE APRENDIZAGEM DE MÁQUINA

Trabalho de Graduação

Santo André

Dezembro de 2023

Agradecimentos

Cursar Engenharia de Informação na UFABC é um sonho que está se realizando, pois foi com muito esforço e estudos que durante três longos anos, após o ensino médio, eu consegui ingressar em uma instituição pública, no caso, a Universidade Federal do ABC, com o objetivo ainda maior, que é o de concluir o curso de graduação.

Realizar esse objetivo só foi possível com o apoio dos meus pais e da minha namorada, que durante essa longa jornada, estiveram ao meu lado nos momentos mais difíceis, sempre acreditando nas minhas escolhas e ainda, me encorajando a enfrentar os desafios quando tive algum tipo de medo ou receio, portanto sou imensamente grato a toda ajuda que eles me ofereceram. Agradeço também aos meus familiares e amigos de graduação, em especial, ao Pedro, Beatriz, Fernando, Ariana e Leandro, que enfrentaram os mesmos desafios que eu e fizeram essa trajetória ficar mais leve e alegre. Também gostaria de agradecer ao meu amigo de infância, Bruno Issamu, que se tornou uma referência de pessoa estudiosa desde o ensino fundamental, me mostrando a importância dos estudos para a realização dos nossos sonhos e objetivos.

Agradeço também o meu orientador de Iniciação Científica e Trabalho de Graduação, Ricardo Suyama, que teve um papel fundamental na minha jornada acadêmica e profissional, me apresentando o fascinante mundo da tecnologia e me desafiando a adquirir novos conhecimentos durante a Iniciação Científica, contribuindo para a minha carreira no mercado de trabalho, além de todos os conselhos que sempre me passou em relação ao trabalho, graduação e vida pessoal.

E por fim, agradeço a Deus pela realização desse objetivo, que sempre me iluminou e me deu forças para enfrentar os desafios da universidade, que muitas vezes, pareciam impossíveis.

Sumário

1 Introdução **4**

2 Fundamentação teórica **7**

2.1 Representação Tempo-Frequencial 7

2.1.1 Transformada Discreta de Fourier 7

2.1.2 Transformada de Fourier de Tempo Curto 8

2.1.3 Espectrograma 10

2.1.4 Cocleagrama 11

2.1.5 Coeficientes Cepstrais de Frequência mel 13

2.2 Mascaramento Tempo-Frequencial 13

2.3 Métricas para avaliação da qualidade dos sinais 16

2.4 Algoritmos de Classificação 17

2.4.1 Árvore de Decisão e Floresta Aleatória 17

2.4.2 Máquinas de Vetores de Suporte 21

2.5 Redes Neurais 25

2.5.1 Funções de ativação 26

2.5.2 Funções Custo 28

2.5.3 Algoritmos de Treinamento 29

2.6 Redes Neurais Convolucionais 32

2.7 Redes Neurais Recorrentes 34

3 Metodologia **37**

3.1 Base de dados 37

3.2 Mascaramento a partir do Espectrograma 39

3.3 Mascaramento a partir do Cocleagrama 41

3.4 Processamento dos dados de áudio 46

4 Resultados e discussão dos resultados **48**

4.1 Análise de desempenho da rede Multilayer Perceptron 48

4.2 Análise de desempenho das Florestas Aleatórias 51

4.3 Análise de desempenho Máquinas de Vetores de Suporte 53

4.4 Análise de desempenho Redes Neurais Recorrentes 55

5 Conclusão **58**

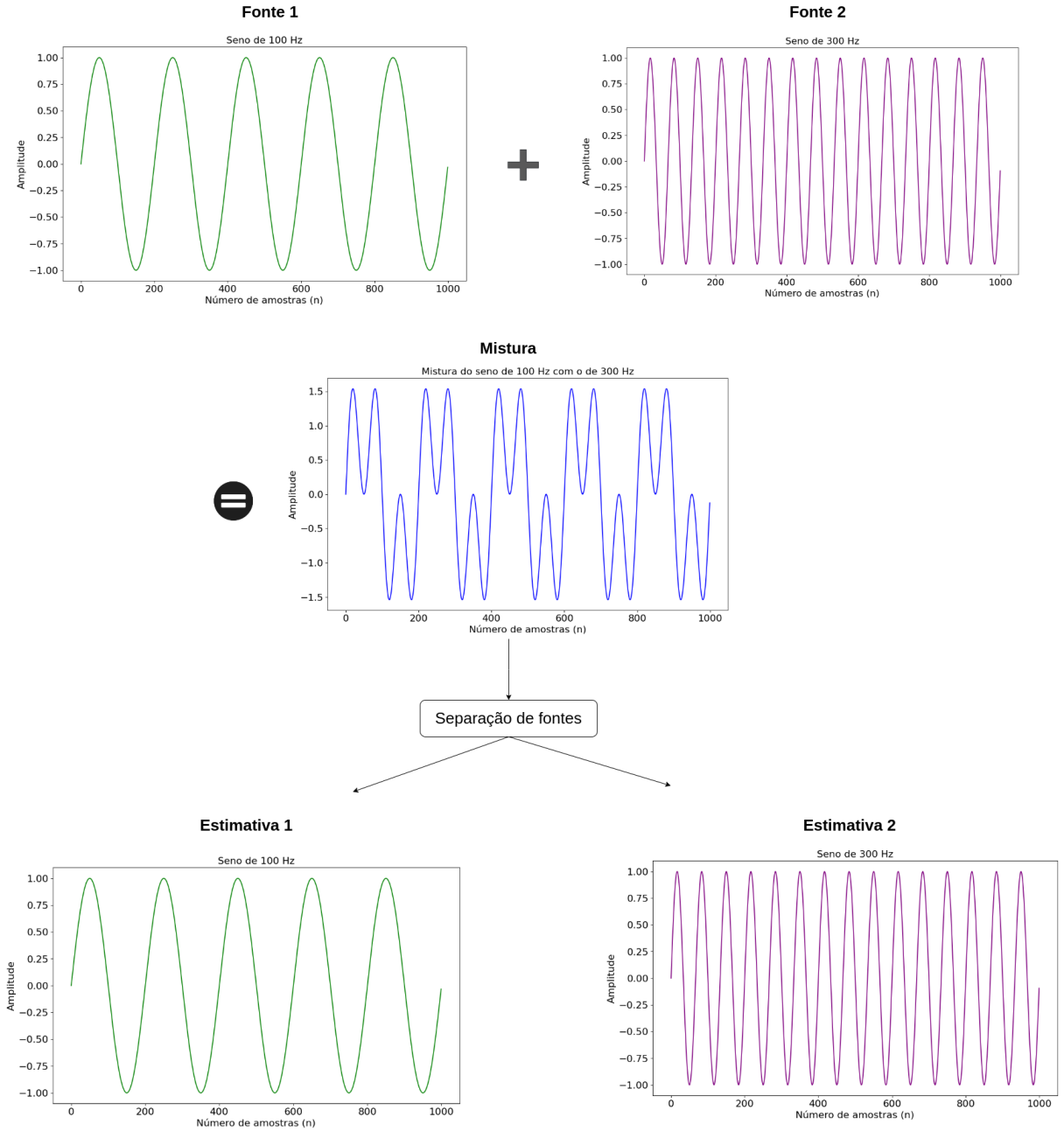
1. *Introdução*

O ser humano possui uma incrível capacidade de perceber, identificar e localizar fontes sonoras. Por exemplo, imagine que uma pessoa está em uma festa, com música ambiente e várias outras fontes sonoras. Mesmo nessa confusão de sons, somos capazes de focar nossa atenção na voz de um interlocutor específico e manter uma conversa sem maiores problemas, exceto quando o volume do ruído é muito alto e impossibilita a comunicação através da fala. Nesse sentido, pode-se imaginar que nosso cérebro é capaz de isolar o sinal de interesse, separando-o dos demais sinais interferentes [1]. Essa incrível capacidade intrínseca ao ser humano seria de grande utilidade se fosse implementada em um sistema automático que fosse capaz de evidenciar um sinal de interesse e mascarar sinais externos ao mesmo, sendo que essa função automática poderia ser utilizada em próteses auditivas, sistemas robustos de reconhecimento automático de voz, entre outras aplicações [1].

O exemplo descrito anteriormente ilustra o problema geral de separação de fontes, apresentado na Figura 1.1. O objetivo nesse problema é recuperar os sinais originais de cada emissor a partir do sinal misturado e combinado. Tradicionalmente, o problema tem sido abordado por meio de técnicas de processamento de sinais como a Análise de Componentes Independentes [2] e Análise de Componentes Esparsas [3]. Entretanto, em algumas situações, principalmente quando há um número reduzido de sensores para estimar os sinais individuais, as técnicas mencionadas não apresentam bom desempenho [4]. Com os avanços tecnológicos e o desenvolvimento cada vez maior de algoritmos, esse problema pode ser abordado usando-se técnicas de Aprendizado de Máquina, e com isso ser capaz de extrair os sinais de interesse mesmo em condições que os métodos clássicos não são eficazes. Por meio destas técnicas é possível obter estruturas capazes de identificar padrões nos sinais captados, como por exemplo a fala de uma pessoa, e a partir dessa informação processar os sinais a fim de mascarar os demais sons interferentes e ressaltar a fala [1].

Uma das abordagens promissoras nesse sentido explora as redes neurais profundas

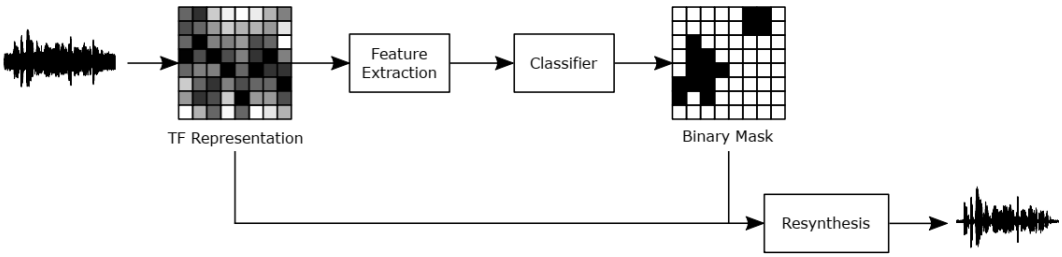
Figura 1.1: Ilustração representando o problema de separação de fontes



(DNN - *deep neural networks*), e se baseia no diagrama apresentado na Figura 1.2. No exemplo ilustrado, o sinal observado corresponde ao sinal de interesse imerso em ruído, e o objetivo é extrair apenas a componente correspondente ao sinal de interesse. Para isso, é possível aplicar uma máscara binária à representação tempo-frequencial do sinal observado, de maneira que os *bins* tempo-frequenciais que apresentam uma relação sinal ruído abaixo de um certo limiar têm seu valor anulado, removendo assim componentes onde o ruído é predominante [4]. A dificuldade adicional, no entanto, reside em determinar quais *bins* tempo-frequenciais contém informação

sobre o sinal de interesse para determinar a máscara e, para isso, trabalhos na literatura têm utilizado DNNs para compor o sistema classificador.

Figura 1.2: Esquema ilustrativo evidenciando o mascaramento binário de um sinal de entrada genérico.



Fonte: adaptado de [4]

Desse modo, o principal objetivo do presente projeto é estudar o problema de extração de sinais utilizando a abordagem descrita anteriormente, criando um sistema automático que consiga realizar a tarefa de mascarar sinais de maneira apropriada para ressaltar apenas o sinal de interesse.

Além do uso das redes neurais, também será explorado variações das redes neurais, como as redes neurais recorrentes e algoritmos mais clássicos de aprendizado de máquina para a tarefa de classificação, como por exemplo máquinas de vetores de suporte, árvores de decisão e florestas aleatórias, com o intuito de explorar algoritmos mais simples e compará-los com arquiteturas mais complexas, como redes neurais profundas.

2. *Fundamentação teórica*

2.1 Representação Tempo-Frequencial

O sinal de áudio representa a variação na amplitude de uma onda mecânica (o som) ao longo do tempo. Embora a representação como uma função no tempo permita extrair algumas informações sobre o sinal, em geral se explora a representação no domínio tempo-frequencial, i.e., descrevendo o sinal em termos de suas componentes espectrais, e como estas variam ao longo do tempo. Nesse sentido, no presente trabalho adotaremos duas representações distintas: a primeira delas dada pelo Espectrograma, e a segunda dada pelo Cocleagrama. Na sequência apresentaremos os principais conceitos relacionados a ambas as representações.

2.1.1 Transformada Discreta de Fourier

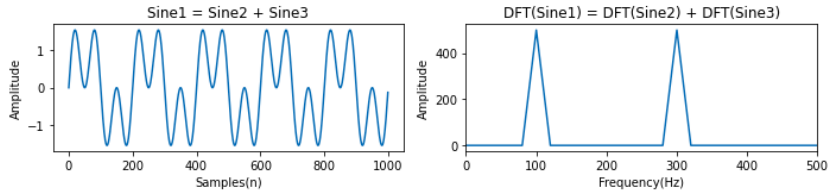
Para representarmos um sinal $x(n)$ de uma forma que seja possível analisarmos quais componentes frequenciais estão presentes, podemos utilizar a transformada discreta de Fourier (DFT - *Discrete Fourier Transform*) que é definida como [5]:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(\frac{2\pi}{N})kn}. \qquad 0 \leq k \leq N-1 \tag{2.1}$$

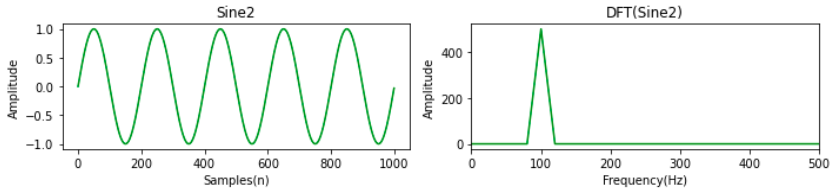
Ao aplicarmos a DFT em um sinal, o que estamos fazendo é basicamente decompondo o mesmo em componentes exponenciais complexas, ou seja, estamos representando o sinal por uma soma de sequências exponenciais complexas [5]. A saída da DFT é um vetor complexo que nos dá informações sobre a fase e a magnitude desse sinal. Se considerarmos o módulo desse vetor, vamos ter uma representação da energia do k-ésimo elemento que corresponde ao coeficiente associado à frequência $F_s \times k/N$, onde F_s é a frequência de amostragem e N o número de amostras da janela que o respectivo sinal tem ao longo do áudio, que é conhecido como

Espectro do sinal, como mostra a Figura 2.1 [1]. Vale ressaltar que a implementação eficiente da DFT é conhecida computacionalmente como Transformada Rápida de Fourier (FFT - *Fast Fourier Transform*), sendo a mesma utilizada para análise de sinais.

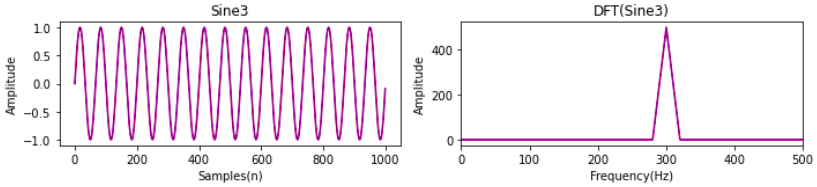
Figura 2.1: Representação de um sinal senoidal, suas componentes senoidais e os respectivos Espectros.



(a) sine1 = sine2 + sine3



(b) sine2



(c) sine3

Vale destacar que a Transformada Discreta de Fourier possui inversa, ou seja, dado um espectro é possível retornar ao domínio do tempo aplicando a equação (2.2).

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j(\frac{2\pi}{N})kn} \quad 0 \leq n \leq N-1 \tag{2.2}$$

2.1.2 Transformada de Fourier de Tempo Curto

A informação fornecida pela DFT revela quais componentes espectrais estão presentes em um dado sinal $x[n]$ mas não indica em que momento cada uma das componentes ocorre dentro da janela de análise. Essa informação é particularmente relevante quando analisamos sinais não estacionários, nos quais as características do espectro variam ao longo do tempo. Nesse contexto, a Transformada de Fourier de Tempo Curto (STFT - *Short Time Fourier Transform*) se torna uma escolha melhor para a análise dos sinais [6].

[1] Exceto quando indicado, todas as figuras foram produzidas pelo autor.

O intuito da STFT é dividir o sinal em pequenos blocos, chamados de *frames*, e realizar a DFT para cada segmento e assumir que, em intervalos curtos de tempo as propriedades do sinal não têm grandes variações, logo ele se comporta como um sinal estacionário.

O primeiro procedimento para o cálculo da STFT consiste em subdividir o sinal $x[n]$ em blocos, multiplicando o sinal de entrada contínuo no tempo por uma *função janela*, como mostra a Figura 2.2. Esse procedimento se repete com o deslocamento da janela até o final do sinal, sendo que as sucessivas janelas se sobrepõem e o tamanho do deslocamento é dado pelo parâmetro *hop length*. A operação de janelamento corresponde à multiplicação do sinal pela função janela $w(n)$, i.e.:

$$x_w(n) = x(n+m) \cdot w(n) \tag{2.3}$$

onde m representa o *frame* atual e n o deslocamento da janela, variando de 0 até $N-1$, sendo N o tamanho do respectivo frame. Uma das funções mais utilizadas para o janelamento é a *Hanning Window*, e a razão para o seu uso é que como a multiplicação no tempo corresponde à convolução na frequência, é preciso multiplicar por uma janela que minimize a distorção que é introduzida no espectro da janela analisada, sendo a mesma definida pela equação:

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N}\right) \qquad 0 \leq n \leq N-1 \tag{2.4}$$

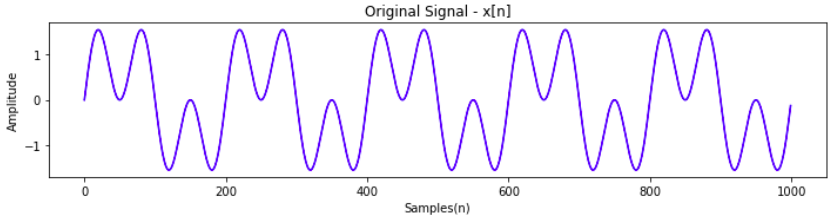
Após o janelamento, aplica-se a transformada discreta de Fourier a cada bloco de dados, o que dá origem à Transformada de Fourier de Tempo Curto [7], expressa matematicamente por:

$$X_n(k) = \sum_{n=0}^{N-1} w(n)x(n+m)e^{-j\frac{2\pi}{N}nk} \tag{2.5}$$

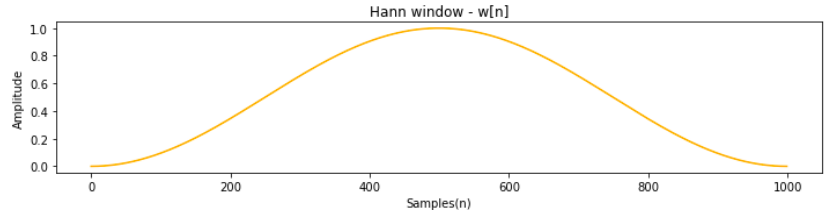
ou seja, para cada índice temporal n temos um espectro. Dessa forma, diferentemente da DFT, a saída da STFT consiste de uma matriz de valores complexos, correspondendo aos espectros dos diferentes frames analisados.

Vale ressaltar que os algoritmos que utilizam a STFT como parte do processamento dos sinais realizam modificações na representação tempo-frequencial dos sinais para então obter o sinal correspondente no tempo. Esse procedimento, entretanto, pode gerar artificialmente uma representação que não corresponde a nenhum sinal real (i.e., não existe um sinal no tempo que possua a STFT modificada). Dessa forma, ao se definir a operação inversa da STFT, em geral, emprega-se a abordagem descrita em [8], que busca encontrar um sinal $\tilde{x}(n)$ que possua a representação tempo frequencial mais próxima possível da STFT dada (no sentido do erro

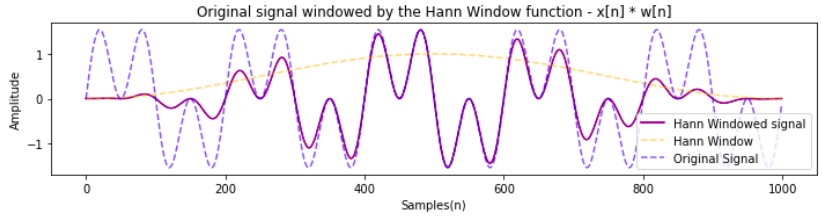
Figura 2.2: Processo de janelamento de um sinal.



(a) Sinal original



(b) Função de janelamento



(c) Sinal original janelado pela função Hanning.

quadrático). A estimativa do sinal, nesse caso, é dada por:

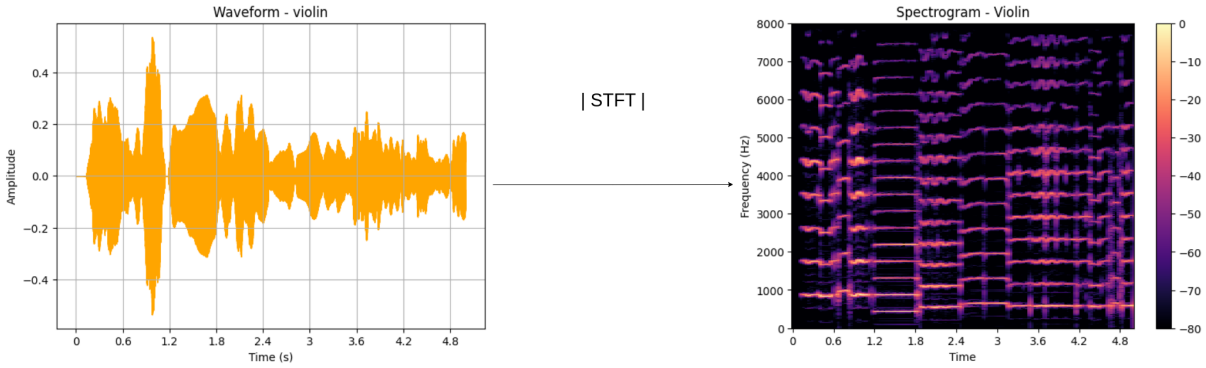
$$x(n) = \frac{\sum_{m=0}^{N-1} x(m) e^{j(\frac{2\pi}{N})kn} w(n-m)}{\sum_{m=0}^{N-1} w^2(n-m)} \tag{2.6}$$

2.1.3 Espectrograma

Calculando-se o módulo da matriz complexa de saída da STFT, obtém-se o *espectrograma*, que fornece uma representação visual da energia de cada componente frequencial do sinal ao longo do tempo. Portanto, a informação matricial pode ser apresentada na forma de uma imagem, em que os eixos representam a frequência e o tempo, e a terceira dimensão, representada pela cor, indica a energia do sinal presente em cada faixa de frequência [9] e cada intervalo de tempo considerado, também denominado de *bin* tempo-frequencial.

Para exemplificar tal conceito, a Figura 2.3 mostra um sinal que corresponde ao som de um violino, que foi dividido em janelas de 2048 amostras com uma sobreposição de 512 amostras. Após o procedimento de janelamento, que neste caso considerou uma janela retangular, para cada conjunto de dados foi calculada a DFT, resultando finalmente no espectrograma do sinal mostrado na Figura 2.3, no qual a energia é geralmente expressa em decibéis.

Figura 2.3: Espectrograma gerado a partir de um sinal áudio de violino.



Ademais, vale destacar que, diferente da DFT e STFT, não é possível reconstruir um sinal a partir do espectrograma, porém utilizamos as informações fornecidas pelo mesmo para construir máscaras que serão posteriormente multiplicadas pela STFT da mistura sinal-ruído com o intuito de separar o sinal de interesse. Esse procedimento será detalhado na seção descrevendo o procedimento de mascaramento 2.2.

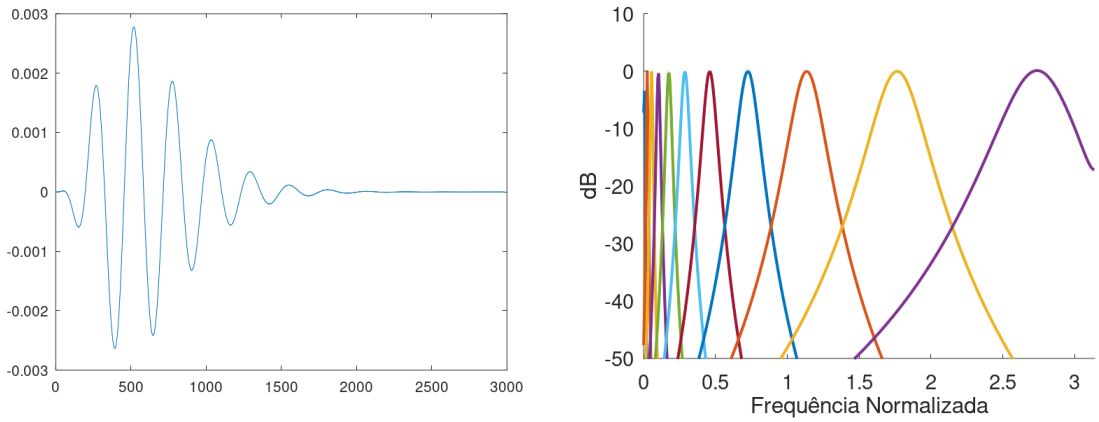
2.1.4 Cocleagrama

O cocleagrama possui um objetivo semelhante ao do espectograma, e busca representar um sinal que varia no tempo na forma Tempo-Frequencial com o propósito de fornecer informações sobre como a energia de cada componente frequencial do sinal varia ao longo do tempo [10], como mostra a Figura 2.5. Entretanto, ele usa o modelo do sistema auditivo humano para determinar a largura de banda entre as frequências, utilizando uma série de filtros passa-faixas, denominados de *gammatones*, que são filtros lineares descritos por uma resposta ao impulso que é o produto entre a distribuição gama e um tom senoidal, i.e.,

$$h(t) = at^{n-1}e^{-2\pi bt}\cos(2\pi ft + \phi) \tag{2.7}$$

onde f determina a frequência central, ϕ é a fase, a a amplitude, n é a ordem do filtro e b é a largura de banda. A Figura 2.4 ilustra a resposta ao impulso de um filtro para uma frequência central específica, bem como a resposta em frequência de um banco de filtros utilizados para a análise com o cocleagrama. As larguras de banda de cada um dos filtros é determinada de maneira que os filtros apresentem frequências centrais igualmente espaçadas na escala ERB (*Equivalent Rectangular Bandwidth*) [11], sendo essa uma escala onde a frequência de entrada deve estar em kHz e a saída retornará em Hz, é válida para sinais de intensidade moderada, foi

Figura 2.4: Resposta ao Impulso e Resposta em Frequência para análise com o cocleagrama

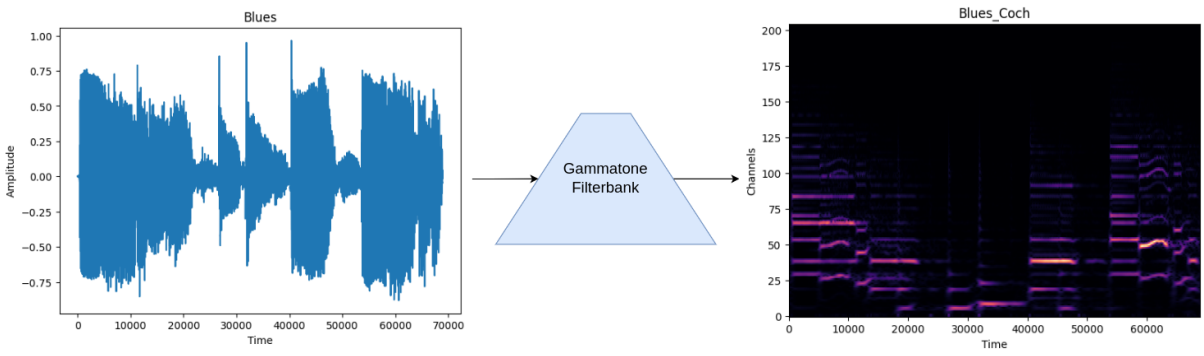


elaborada levando em consideração o ouvido de pessoas jovens e sua primeira versão considera componentes frequenciais entre $0.1kHz$ e $6.5kHz$, enquanto que a segunda versão considera componentes entre $0.1kHz$ e $10kHz$. A relação entre a frequência f e o equivalente na escala ERB é definida matematicamente por:

$$ERB(f) = 6.23f^2 + 93.39f + 28.52 \tag{2.8}$$

Então, para gerar o cocleagrama, a forma de onda do sinal é decomposta em bandas espectrais pelo banco de filtros e a energia dos sinais de saída de cada filtro (canais de frequência) fornece uma representação análoga ao espectrograma [12]. A principal diferença entre as duas representações, é que no espectrograma a largura de banda é constante entre todos os canais de frequências, enquanto que no cocleagrama essa largura aumenta à medida que a frequência aumenta.

Figura 2.5: Representação de um áudio no domínio Tempo-Frequência gerado através do módulo de um banco de filtros *gammatone*. cocleagrama realizado em Python.

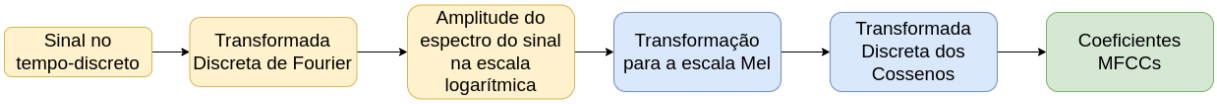


2.1.5 Coeficientes Cepstrais de Frequência mel

Os Coeficientes Cepstrais de Frequência mel (*Mel-Frequency Cepstral Coefficients* - MFCC) são atributos amplamente utilizados na análise de sinais de áudio, em especial pelas áreas de reconhecimento de fala, classificação de gêneros musicais e na comunidade musical [13]. Esses coeficientes capturam estruturas importantes no espectro de frequência de um sinal, reduzindo a dimensionalidade de sua representação.

Para extrair os coeficientes MFCCs de sinal no tempo, primeiro aplicamos a Transformada Discreta de Fourier (DFT) para obter a representação frequencial correspondente. Em seguida, calculamos o logaritmo da amplitude do espectro para obter uma representação logarítmica das amplitudes em função das frequências. Esse passo é crucial, pois os seres humanos percebem a amplitude do som de maneira logarítmica, não linear. Após a aplicação do logaritmo, convertemos as frequências para a escala Mel, uma etapa perceptualmente importante, pois os humanos são mais sensíveis às variações nas componentes de baixa frequência do que nas altas frequências. Dessa forma, a escala Mel busca realizar um mapeamento em escala logarítmica, tornando as distâncias entre as frequências perceptualmente iguais para nós, seres humanos [13]. Finalmente, aplicamos a Transformada Discreta dos Cossenos para extrair os coeficientes MFCCs do sinal [13]. O processo de extração dos MFCCs de um sinal é ilustrado na Figura 2.6.

Figura 2.6: Fluxograma de extração dos coeficientes MFCCs de um sinal no tempo



2.2 Mascaramento Tempo-Frequencial

A representação tempo-frequencial (seja na forma de um espectrograma ou cocleagrama) pode revelar estruturas importantes no sinal de interesse, além de ressaltar conteúdo que gostaríamos de desprezar, como o ruído. Dessa forma, sabendo qual porção da representação está associada ao sinal de interesse e qual está associada ao ruído, é possível realizar um mascaramento da representação tempo-frequencial, removendo o ruído e preservando, tanto quanto possível, a informação do sinal. Para realizar esse processo, é possível implementar máscaras

que vão atuar como um filtro sobre o sinal original, destacando ou atenuando características específicas do sinal. A máscara pode ser representada como uma matriz de pesos, onde cada elemento indica a influência relativa de uma determinada frequência em um determinado intervalo de tempo. Para isso, duas máscaras foram utilizadas, sendo elas a máscara binária ideal (IBM - *Ideal Binary Mask*) [14] e a *Ideal Ratio Mask* (IRM) [15].

A *máscara binária ideal* é inspirada no mascaramento que o sistema auditivo humano realiza de forma automática. A ideia subjacente ao método consiste em preservar a informação do espectrograma em unidades tempo-frequenciais que apresentem energia acima de um limiar, e anular a informação onde a energia for desprezível. Dessa forma, matematicamente, define-se a máscara ideal binária como uma matriz, onde cada elemento é dado por:

$$IBM = \begin{cases} 1, \text{ se } SNR(t,f) > \text{Critério Local} \\ 0, \text{ caso contrário} \end{cases} \tag{2.9}$$

Onde (t,f) representa uma unidade (*bin*) tempo-frequencial e o SNR (*Signal-to-Noise Ratio*) é a razão da energia do sinal limpo pela energia do ruído. O "Critério Local", atua como o limiar para decidir se um *bin* tempo-frequencial deve ser considerado como sinal de interesse ou ruído, sendo o mesmo calculado como a média da energia do sinal presente no espectrograma.

Portanto, valor da IBM será igual a 1 (sinal de interesse) para a respectiva unidade (t,f) caso a razão sinal-ruído daquele ponto exceda o critério local, e 0 caso contrário (ruído) [1]. No entanto, é importante ressaltar que a IBM pode apresentar limitações significativas em cenários nos quais a potência do ruído excede significativamente a potência da fala, nestes casos, o uso da IBM pode resultar em uma atenuação inadequada do sinal de fala, prejudicando a qualidade do mascaramento. Também vale destacar que para criar a IBM, é preciso ter conhecimento sobre o sinal limpo, o que muitas vezes não temos. As Figuras 2.7a e 2.7b mostram o cocleagrama da Figura 2.5 imerso em ruído e a máscara binária estimada a partir da mistura, respectivamente.

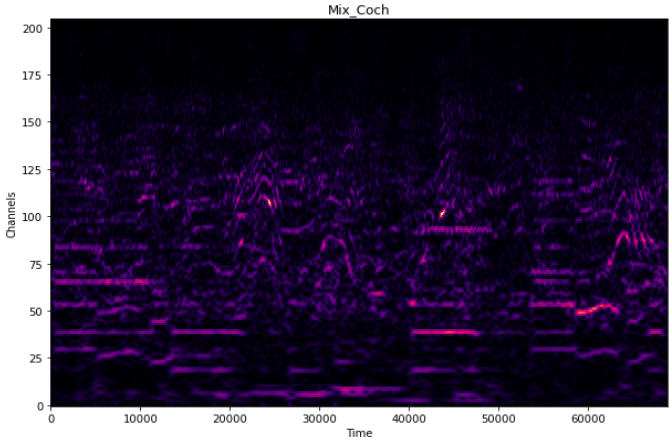
A *ideal ratio mask*, por outro lado, é uma versão suavizada da IBM, e seu valor, para cada unidade tempo-frequencial, é definido pela seguinte expressão:

$$IRM = \frac{S(t,f)^2}{S(t,f)^2 + N(t,f)^2} \tag{2.10}$$

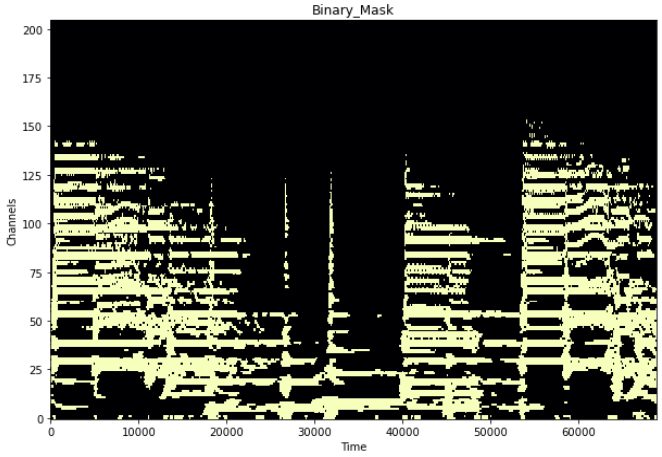
onde $S(t,f)^2$ e $N(t,f)^2$ representam a energia do sinal de interesse e do ruído em cada unidade tempo-frequencial, respectivamente.

A máscara binária apresenta certa perda na qualidade do áudio (quando comparado ao sinal limpo), porém mantém a inteligibilidade alta. Por outro lado, utilizando a *ratio mask*,

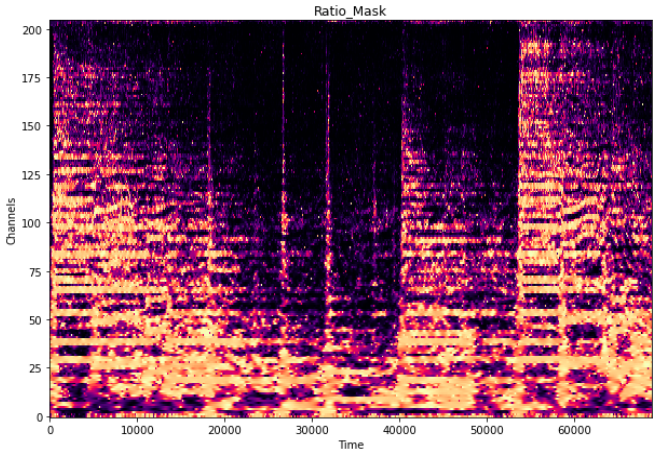
Figura 2.7: Cocleagrama, Máscara Binária e *Ratio Mask* estimadas a partir do cocleagrama da Figura 2.5 imerso em ruído.



(a) Mix do cocleagrama da Figura 2.5 com um ruído externo.



(b) Máscara binária estimada a partir do cocleagrama da Figura 2.7a.



(c) *Ideal Ratio Mask* para o cocleagrama da Figura 2.7a.

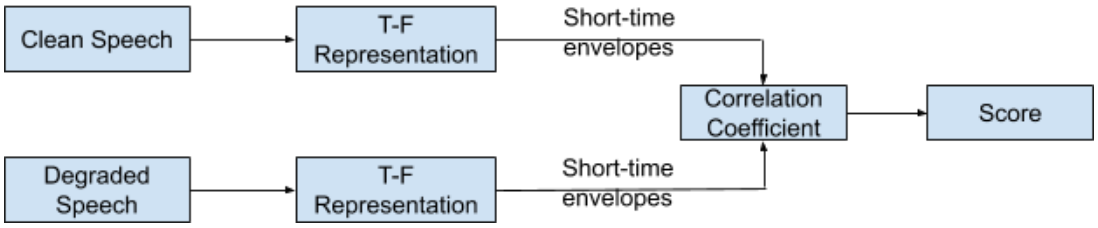
tanto a qualidade quanto a inteligibilidade do áudio são altas. A Figura 2.7c mostra a *ideal ratio mask* para o cocleagrama da Figura 2.7a.

2.3 Métricas para avaliação da qualidade dos sinais

Para analisar a performance das máscaras, em termos da qualidade do sinal reconstruído após aplicação do método, pode-se utilizar uma métrica de inteligibilidade como o *Short Term Objective Intelligibility* (STOI). O objetivo do mesmo é comparar um áudio que sofreu algum tipo de degradação com o de fala original. Para isso, os dois sinais são representados na forma tempo-frequencial, sendo que os *bins* que não apresentam nenhum conteúdo são retirados da representação. Envelopes de curto prazo são comparados por meio de coeficientes de correlação para estimar a medida de inteligibilidade do áudio [12], um valor que varia de 0 a 1, como mostra a Figura 2.8. É importante ressaltar que a métrica STOI apresenta algumas restrições, tais como:

- Todas as janelas cuja intensidade sonora da voz limpa esteja 40 dB abaixo da intensidade da janela com a maior intensidade sonora da voz limpa serão ignoradas no cálculo;
- A frequência de amostragem utilizada é de 10 kHz, embora seja possível utilizar frequências de amostragem maiores.

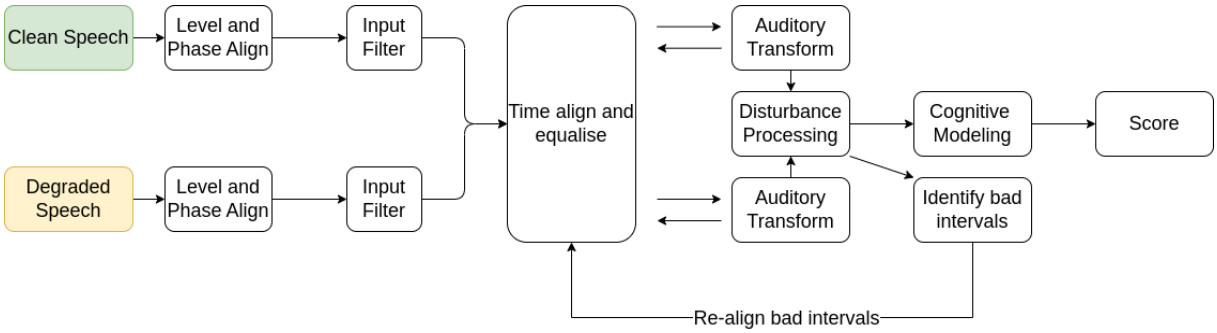
Figura 2.8: Diagrama de blocos representando o processo para extração do STOI.



Uma outra métrica que pode ser utilizada para validar o uso de máscaras é o *Perceptual Evaluation of Speech Quality* (PESQ), que analisa a qualidade do áudio pós-processamento. Para isso, os dois sinais (limpo e com degradação) têm suas amplitudes e fases alinhadas. Em seguida, são filtrados com um filtro de entrada para modelar um fone de ouvido telefônico padrão. Então os sinais são alinhados e equalizados no tempo, e então parâmetros de distorção são extraídos da diferença dos dois sinais em uma representação tempo-frequencial e a nota de qualidade é estimada [16], como mostra o esquemático na Figura 2.9, onde esta nota varia de 0.5 (muita distorção) até 4.5 (sem distorção). Assim como a métrica STOI, a PESQ também apresenta restrições, como:

- A PESQ trabalha na escala Bark;
- O janelamento utiliza a janela de Hann com 32 ms;
- Pode assumir valores entre $-0,5$ e $4,5$ em cenários com distorção muito elevada, mas costuma se restringir ao intervalo de $1,0$ a $4,5$ para a maioria dos cenários.

Figura 2.9: Diagrama de blocos representando o processo para extração do PESQ.



Tanto o STOI quanto o PESQ, são métricas que foram baseados na métrica *Mean Opinion Score (MOS)*. Essa métrica foi definida pela *International Telecommunication Union (ITU)*, que a defini como sendo a média das notas atribuídas à um sistema, com base na opinião de um grupo de avaliadores [17], sendo que as notas podem variar entre 1 (ruim) e 5 (excelente). Porém, obter tais métricas pode ser muito caro, exige tempo para recrutar pessoas e realizar as avaliações, por isso as métricas detalhadas nessa seção tem um papel crucial ao tentar avaliar essas notas de forma automática.

Vale salientar que existem outras métricas para analisar a qualidade de sinais além das quais foram apresentadas até o momento, como por exemplo as métricas *Source-to-Distortion Ratio* (SDR), *Source-to-Interferences Ratio* (SIR) e *Source-to-Artifacts Ratio* (SAR) [18]. No entanto, essas métricas não levam em consideração a maneira que o ouvido humano percebe o som, mas sim as características estatísticas presentes nas ondas dos sinais.

2.4 Algoritmos de Classificação

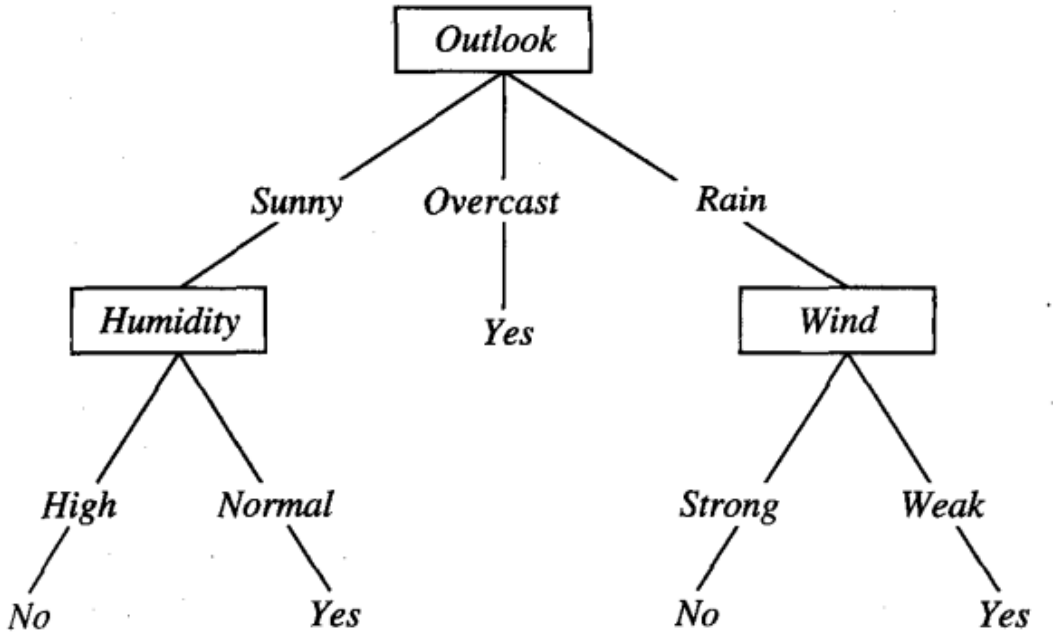
2.4.1 Árvore de Decisão e Floresta Aleatória

A árvore de decisão é um algoritmo de classificação onde a função aprendida pelo modelo é representada por uma série de regras do tipo ”se...então”[19], podendo ser representado

por uma árvore de cabeça para baixo, onde cada nó ou folha da árvore é uma regra de validação do algoritmo.

Para exemplificar a ideia do algoritmo, podemos observar a Figura 2.10, sendo esse um exemplo de árvore de decisão onde o objetivo é avaliar se o clima está adequado para jogar tênis ou não. Podemos interpretar essa árvore da seguinte forma: *Outlook* é o fator mais importante para determinar se o dia está bom para jogar tênis ou não. Por exemplo, se selecionarmos um dia nublado, de acordo com o algoritmo, o clima está adequado para a prática de tênis. Agora, se o clima estiver ensolarado, vamos depender de um outro fator que determinará a probabilidade ou não de se jogar tênis, nesse caso é a umidade do ar. E, se o clima estiver chuvoso, a intensidade do vento que determinará a probabilidade de se jogar tênis.

Figura 2.10: Árvore de decisão que indica a possibilidade de se jogar tênis.



O processo de criação de uma árvore de decisão depende de dois procedimentos. O primeiro é dividir o espaço de predição em regiões diferentes e que não se sobrepõem. O segundo passo seria avaliar em qual das regiões uma dada amostra é pertencente.

Para a construção da árvore, utiliza-se o processo de divisão binária, isso significa que as divisões se iniciam no topo da árvore, onde, inicialmente, todas as amostras pertencem a mesma região. Após a primeira divisão, dois novos ramos são criados e a regra ”se ... então”se aplica, depois o procedimento se repete para novas divisões sucessivas [20], até a criação de uma árvore de decisão.

A equação que define a divisão das regiões é definida por 2.11:

$$R_1(j,s) = \{X|X_j < s\} \ \& \ R_2(j,s) = \{X|X_j \geq s\} \tag{2.11}$$

A equação 2.11 diz que R_1 é a região que contém todos os pontos X_j menores do que s e R_2 é a região de todos os pontos maiores ou iguais a s.

Após a definição das duas regiões, escolhemos uma delas e realizamos o mesmo procedimento de divisão binária, onde cada atributo é avaliado de forma independente através de um teste estatístico, sendo que a métrica a ser avaliada é a entropia, a fim de determinar o quão bom aquele atributo é para realizar a tarefa de classificação, logo, o atributo que possuir menor entropia, apresentará maior pureza e, portanto, é selecionado como nó da árvore, depois os demais atributos serão testados e novamente os que possuírem valores de entropia mais próximos de zero, se tornarão novos nós da árvore [20].

A entropia é uma medida da teoria da informação que mede pureza de um conjunto de dados, definida por 2.12:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk}) \tag{2.12}$$

Onde D é a entropia do nó, K é o número de classes, $\hat{p}_{mk}$ é a proporção de observações no nó que pertencem à classe k. Quanto menor a entropia, maior a pureza do nó, indicando que todas as observações pertencem à mesma classe.

Apesar de gerar bons resultados, as árvores de decisão podem sofrer de sobreajuste nos dados de treinamento, gerando uma baixa performance nos dados de teste. Para contornar esse problema, pode-se criar uma árvore de decisão com vários ramos e ao final realizar o processo de poda de complexidade de custo [20], que funciona da seguinte forma: inicialmente, é construído a árvore de decisão completa usando os dados de treinamento. Após realizar uma avaliação da performance da árvore, um parâmetro de poda é introduzido, denotado por α, quanto maior esse valor, mais complexo será a árvore. Após a poda de alguns ramos da árvore, uma análise no desempenho é realizada, a fim de validar se houve ou não uma melhora no desempenho do algoritmo. O objetivo desse procedimento é diminuir o sobreajuste e encontrar uma árvore que possui uma boa generalização.

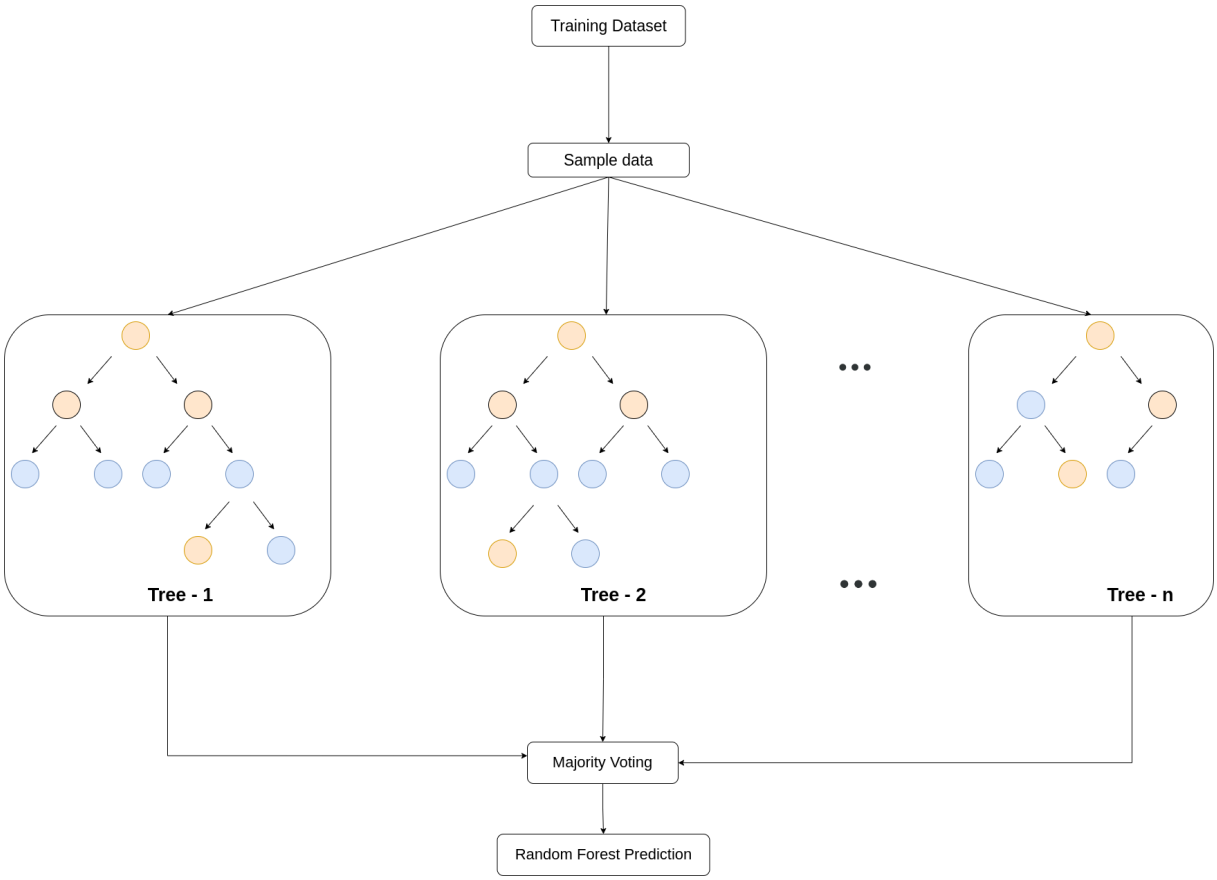
Dado as desvantagem da árvore de decisão em ser propícia ao sobreajuste, a floresta aleatória tenta corrigir esse comportamento construindo múltiplas árvores de decisão nos dados de treinamento.

A princípio, dados de treinamento são aleatoriamente selecionados e amostrados em

lotes de subamostras distintas. Em cada uma dessas subamostras, uma árvore de decisão é treinada utilizando um subconjunto de preditores em cada divisão, e o número de subamostras é tão grande quanto a raiz quadrada do número de preditores.

Com as árvores de decisão construídas, a previsão para o valor de uma nova amostra x é feita pelo valor mais frequente das árvores de decisão, como exemplifica a Figura 2.11. Dessa forma, ao utilizar vários modelos de baixo viés, mas de alta variância, e analisar o valor mais frequente, conseguimos reduzir a variância geral, resultando em um modelo mais preciso e robusto.

Figura 2.11: Diagrama do fluxo de uma Floresta Aleatória.



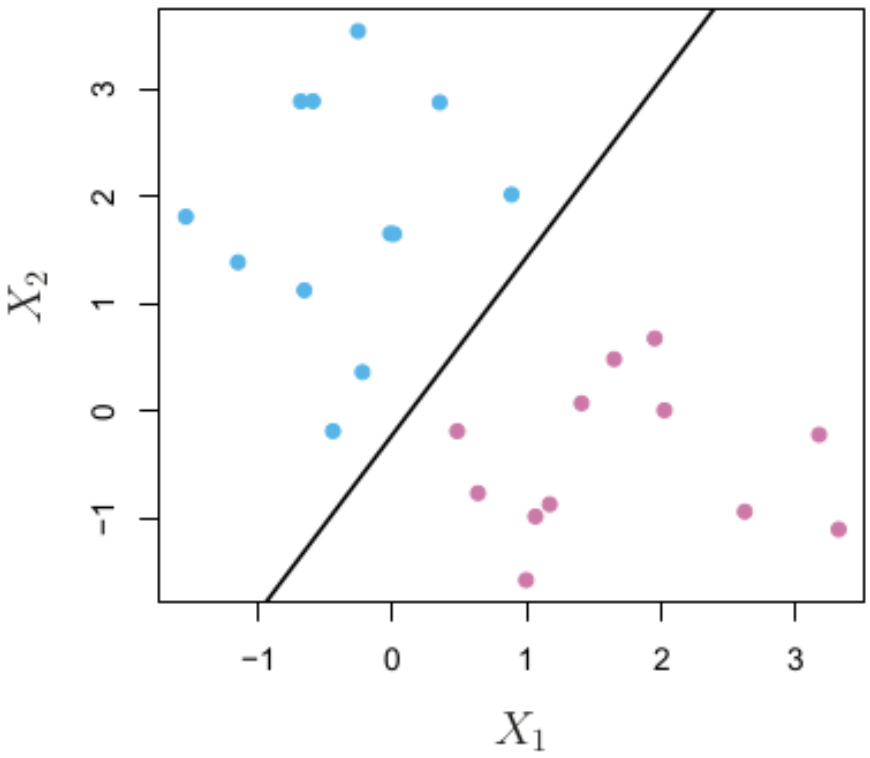
2.4.2 Máquinas de Vetores de Suporte

As máquinas de vetores de suporte (*Support Vector Machine - SVM*) são usadas para problemas de aprendizado supervisionada, como por exemplo tarefas de classificação, clusterização e regressão [21]. O objetivo do algoritmo é criar um classificador que gera um hiperplano capaz de separar o conjunto de dados em duas ou mais classes, tendo sempre a dimensão $p-1$, onde p é o espaço representando as *features* do nosso modelo. Por exemplo, se estamos trabalhando em um espaço tridimisional, o hiperplano terá duas dimensões. A equação geral do hiperplano pode ser representada por 2.13 [20].

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0 \tag{2.13}$$

A Figura 2.12 ilustra um conjuto de dados linearmente separáveis, nesse caso, o espaço p é de duas dimensões e então, por definição, o hiperplano é unidimensional, nesse caso, uma reta que separa os conjuntos de dados em duas partes.

Figura 2.12: Exemplo de um hiperplano unidimensional

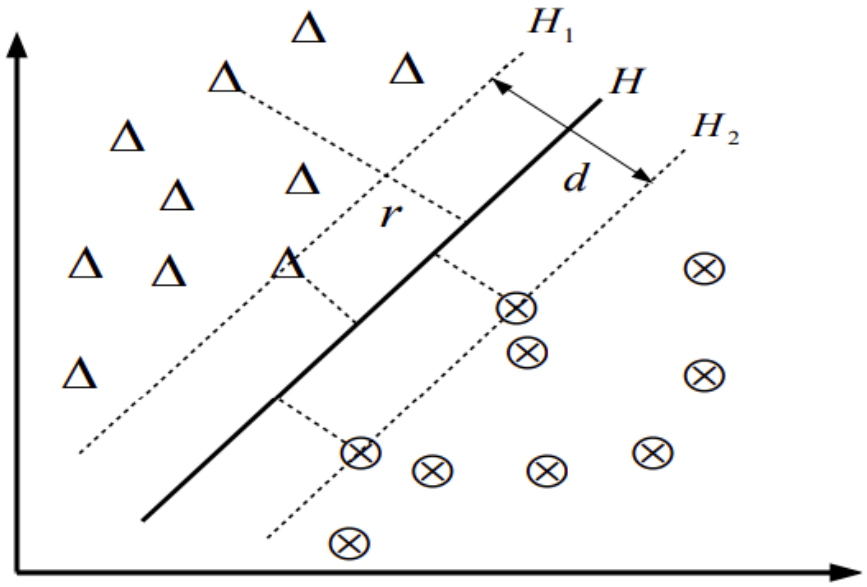


Fonte: adaptado de [20]

Porém, existem infinitas possibilidades de hiperplanos para a Figura 2.12, já que po-

demos rotacionar o hiperplano gerado, deslocá-lo e assim por diante, portanto, o objetivo do algoritmo é encontrar o plano que possui maior margem entre as diferentes classes, ou seja, maximizar a distância entre os vetores de suporte, que são pontos mais próximos dos hiperplanos, como mostra a Figura 2.13, sendo H o hiperplano de sepração dos dados e d a largura da margem.

Figura 2.13: Exemplo de um hiperplano do SVM.



Fonte: adaptado de [21]

O problema da abordagem dos hiperplanos de margem máxima é que o algoritmo é sensível a novos dados, indicando uma provável tendência ao sobreajsute e de alta variância, não sendo um modelo robusto para a tarefa de classificação. Nesse caso, devemos considerar hiperplanos que não separam os dados perfeitamente em duas classes, melhorando a performance do modelo e sua robustez. Esse processo de permitir a classificação errada de alguns dados em detrimento a uma melhor generalização por parte do modelo é chamado de ***soft margin classifier*** [20].

No entando, essa tolerância a classificação errada aos dados pode ser ajustada pelo parâmetro C na restrição apresentada por 2.17. Sendo que a 2.14 informa o objetivo do algoritmo, que é maximar a margem H entre as classes, e a equação 2.15 representa uma restrição de que a soma dos quadrados dos coeficientes do hiperplano deve ser igual a 1. Já a variável ε_i indica onde o ponto i_{th} está localizado em relação ao hiperplano e a margem. Se $\varepsilon_i = 0$, então o ponto está no lado correto da margem, se $\varepsilon_i > 0$, então está no lado incoreto da margem, porém no lado certo do hiperplano, agora, se $\varepsilon_i > 1$, significa que o ponto está o lado errado do hiperplano.

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\varepsilon_1,\ldots,\varepsilon_n,H}{\text{maximize}} \quad H \tag{2.14}$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{2.15}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i), \tag{2.16}$$

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^{n} \varepsilon_i \leq C \tag{2.17}$$

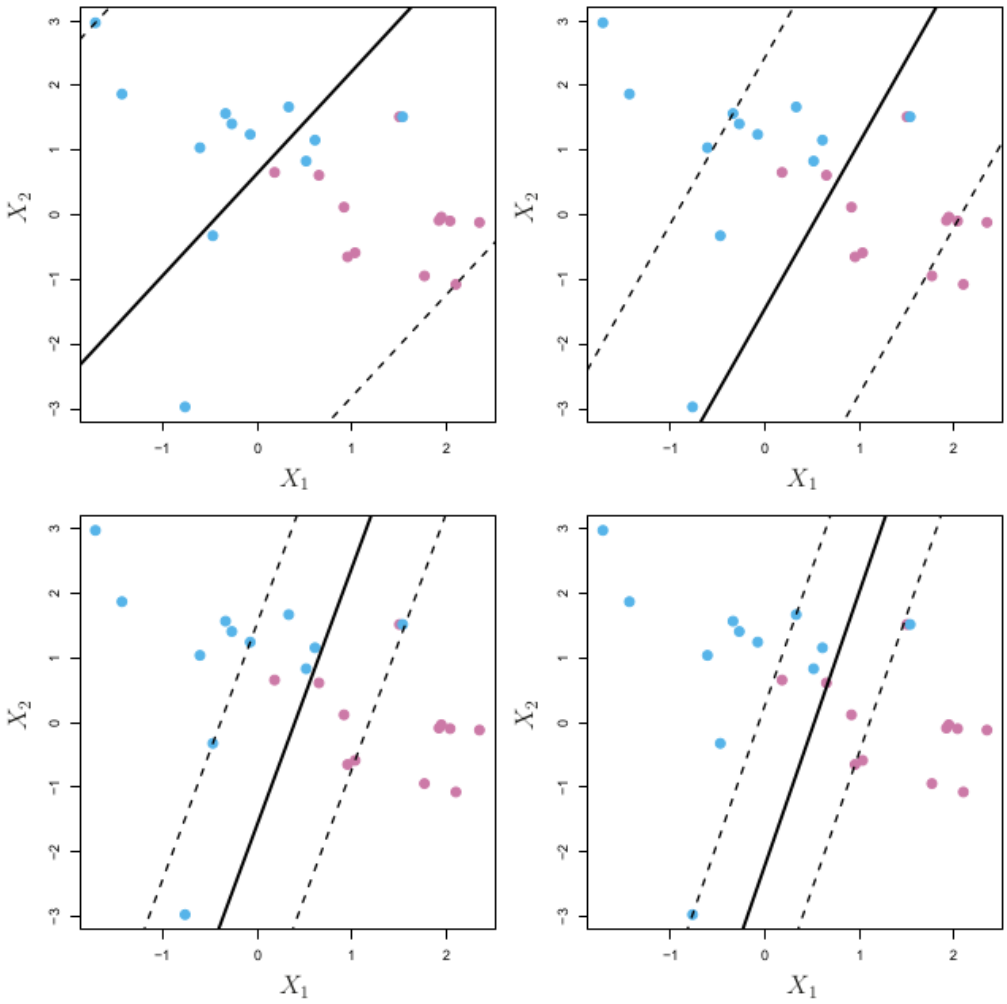
Em relação ao parâmetro de ajuste C, temos que ele limita e determina o nível de violação da margem. À medida que o valor de C aumenta, tornamos o modelo mais tolerante a violações, e desse forma, a largura da mesma será maior. Agora, se C diminui, temos uma margem menor, portanto baixa tolerância a erros de classificação.

O ajuste do parâmetro C está diretamente ligado ao controle de viés-variância, isto é, a medida que aumenta-se o valor do parâmetro, o modelo terá um viés maior, porém baixa variância. Enquanto que, se C apresenta um valor menor, a largura da margem diminui e o modelo apresenta um baixo viés, porém alta variância. Portanto, o objetivo é encontrar o valor de C que aumenta a performance do modelo em generalizar a predição para novos dados e evita tanto o subajuste quanto o sobreajuste do algoritmo.

A Figura 2.14 exemplifica a influência do parâmetro de ajuste C na largura da margem em um conjunto de dados.

Na figura, da esquerda para direita, o valor do parâmetro C diminui e consequentemente, a margem também fica mais estreita.

Figura 2.14: Ajuste do parâmetro *C* no algoritmo do SVM.



Fonte: adaptado de [20]

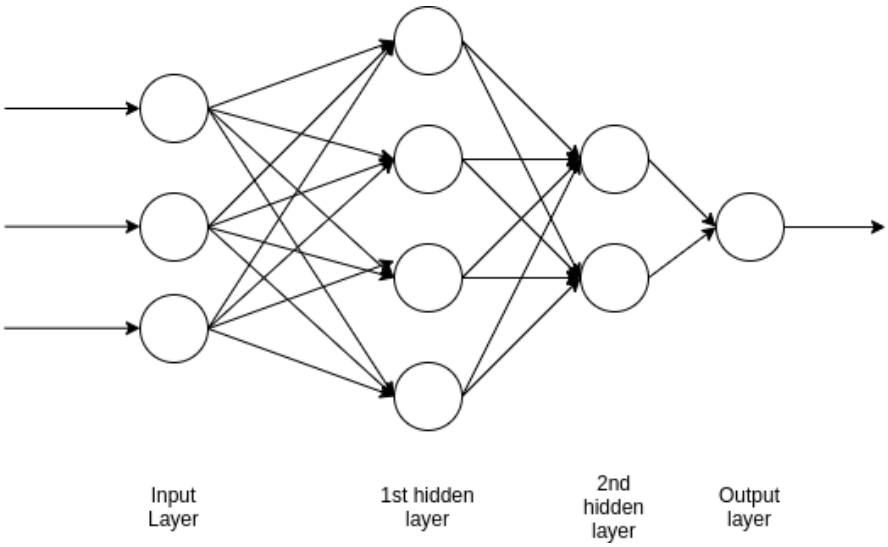
Para problemas de separação não linear, é possível usar uma extensão do método através dos *Kernels* de separação [20], onde o objetivo é aumentar o espaço de features para acomodar um limite não-linear entre as classes.

2.5 Redes Neurais

Quando usamos o espectrograma para analisar o áudio, podemos usar as Redes Neurais Artificiais (*ANN - Artificial Neural Networks*) para classificar cada *bin*, de tempo-frequência no espectrograma, como sendo parte do sinal ou ruído. Portanto, é possível utilizar o algoritmo de tal forma que a saída da rede seja uma máscara binária.

A estrutura de uma rede neural artificial é ilustrada na Figura 2.15. O modelo é composto por várias unidades de processamento (os neurônios) que geralmente são organizados seguindo uma determinada topologia. No caso específico do exemplo apresentado a rede é estruturada em camadas, e corresponde a um tipo particular de rede denominada *Multi Layer Perceptron* (MLP) [22] dividido em camadas - uma camada com as entradas, camadas "escondidas" e a camada de saída, sendo que todas elas estão interconectadas por pesos sinápticos [22].

Figura 2.15: Arquitetura de uma rede neural Multi Layer Perceptron.



O interesse por esse tipo de estrutura reside em sua grande flexibilidade, sendo capaz de aproximar praticamente qualquer mapeamento entrada-saída desejado e assim atuar como um classificador. Para isso, o modelo MLP deve ser treinado, ou seja, ter os pesos sinápticos ajustados de acordo com um critério específico [1]. Em geral utiliza-se um conjunto de dados de treinamento, composto por pares de entradas e saídas desejadas, e os pesos são ajustados visando diminuir o erro entre a saída atual da rede e o valor desejado para sua saída. Esse ajuste é realizado através de algoritmos de otimização, normalmente baseados na informação do gradiente, como o algoritmo de *BackPropagation* e o *Adaptive Moment Estimation* (ADAM)

[1].

2.5.1 Funções de ativação

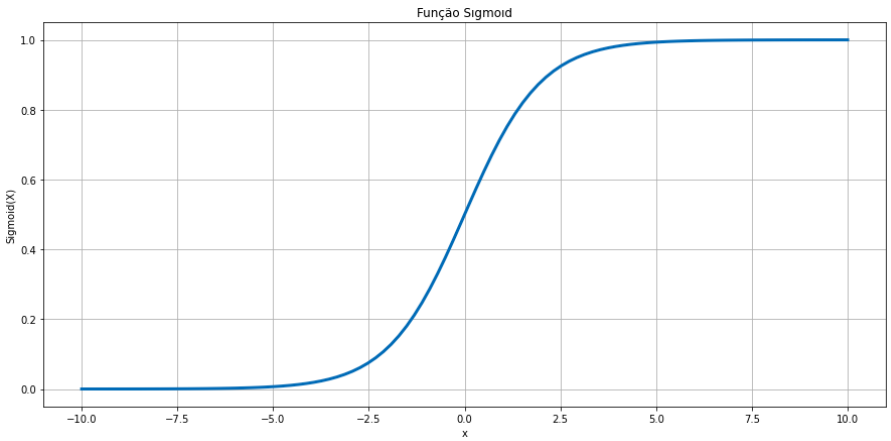
As funções de ativação introduzem um comportamento não-linear aos modelos dos neurônios da rede neural. Matematicamente, a função é aplicada à soma dos pesos sinápticos multiplicados pelas entradas [23], e por conta disso ela influencia no cálculo do gradiente em relação aos pesos através de algum algoritmo de otimização, como *backpropagation* ou o *ADAM* [23]. Há diferentes possibilidades de escolha, e na sequência são apresentados algumas funções usualmente empregadas.

Função Sigmóide

A função de ativação Sigmóide tem seu valor limitado entre 0 e 1 e é uma função contínua. Essa função basicamente mapeia toda a entrada em uma saída pela relação 2.18 [23], muito utilizada em problemas de classificação binária, sendo o seu gráfico representado pela Figura 2.16.

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.18}$$

Figura 2.16: Gráfico da função Sigmóide.



Existem variações da sigmoidal como função de ativação, como por exemplo a *Hard Sigmoid Function* [23], *Sigmoid-Weighted Linear Units* [23], *Derivative of Sigmoid-Weighted Linear Units (dSiLU)* [23]. Essas variantes tem como objetivo contornar alguns pontos negativos que a sigmoid apresenta, como por exemplo convergência demorada no algoritmo de *backpropagation* e atualizações do gradiente em diferentes direções.

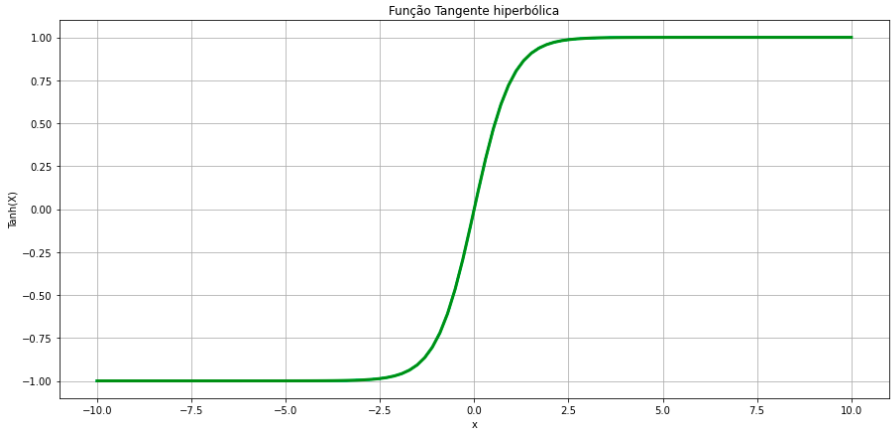
Função Tangente Hiperbólica (Tanh)

A tangente hiperbólica é um outro tipo de função de ativação utilizada em problemas que utilizam redes neurais, definida por [23]:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.19}$$

Possui um intervalo que varia de -1 até 1, e considera valores negativos, como mostra a Figura 2.17.

Figura 2.17: Gráfico da função Tanh.



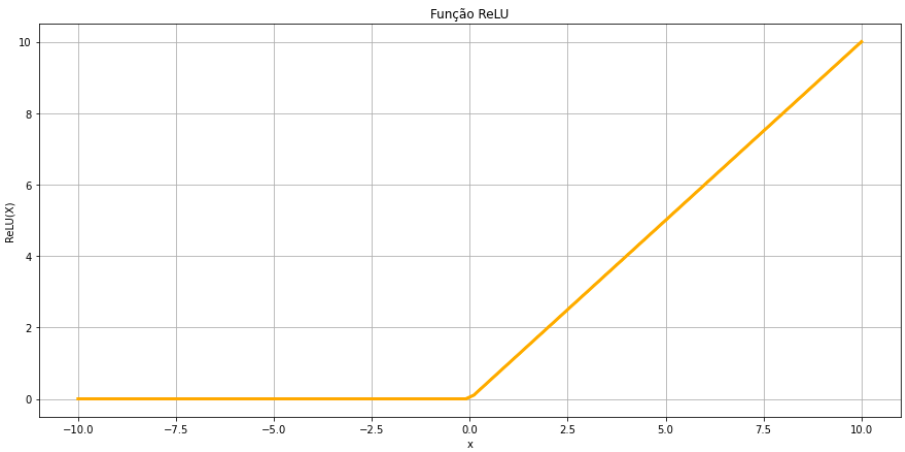
Função de unidade linear retificada (ReLU)

A *ReLU* é uma das funções de ativação mais utilizada em problemas de redes neurais e a que apresenta um dos melhores resultados [24], por ser rápida de convergir, apresentar uma boa generalização, e por apresentar boa eficiência computacional, já que não apresentam exponenciais em sua definição [25], como mostra a equação 2.20.

$$f(x) = max(0,x) = \begin{cases} x_i, \text{ if } x_i \geq 0 \\ 0, \text{ if } x_i < 0 \end{cases} \tag{2.20}$$

Portanto, para valores menores que 0, a saída da função é nula, para valores maiores, o resultado da função é o próprio valor, como mostra a Figura 2.18.

Figura 2.18: Gráfico da função ReLU.



2.5.2 Funções Custo

As funções custo tem como objetivo quantificar a qualidade do mapeamento obtido. Para isso, em geral, as funções custo são baseadas na diferença entre os valores estimados de uma rede neural com seus valores desejados. Por sua vez, o algoritmo de otimização é utilizado para se obter a melhor configuração de pesos sinápticos, i.e., que apresentem o menor valor da função de custo.

Erro quadrático médio

Uma função de custo muito utilizada em problemas de regressão e também para treinar um algoritmo que prevê a *ideal ratio mask*, é o erro quadrático médio, definido como a média

da diferença entre o valor estimado e o valor esperado ao quadrado [1], i.e.:

$$MSE = \frac{1}{N}\sum(y_i - \hat{y}_i)^2 \tag{2.21}$$

onde y_i é o valor esperado, $\hat{y}_i$ é o valor estimado e N o número de dados de entrada.

Entropia cruzada

A função de entropia cruzada avalia os valores preditos pela rede para os dados de treino em comparação com os valores reais daqueles pontos, onde o objetivo é avaliar o quão boa foi a predição realizada pela rede. Sendo assim, quando a rede classifica um *bin* tempo-frequencial, ela vai nos dizer qual a probabilidade daquele *frame* ser sinal de interesse (fala de uma pessoa) ou de ser um ruído, caso essa probabilidade esteja muito longe de seu rótulo real (sinal de fala ou ruído), a função de custo vai ter um valor alto, quando a probabilidade de um ponto for próxima ao seu rótulo real (sinal de fala ou ruído) a função de custo terá um valor baixo. A função de entropia cruzada é definida pela seguinte relação [1]:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} I_{i,c} log(p_{i,c}) \tag{2.22}$$

onde i é o índice do neurônio de saída, $p_{i,c}$ representa a probabilidade predita do i pertencer à classe c, N e C indicam o número de neurônios de saída e o número de classes, respectivamente. E por fim, $I_{i,c}$ é o resultado binário, que é igual a 1 se o valor da classe do neurônio i é c ou 0 caso contrário.

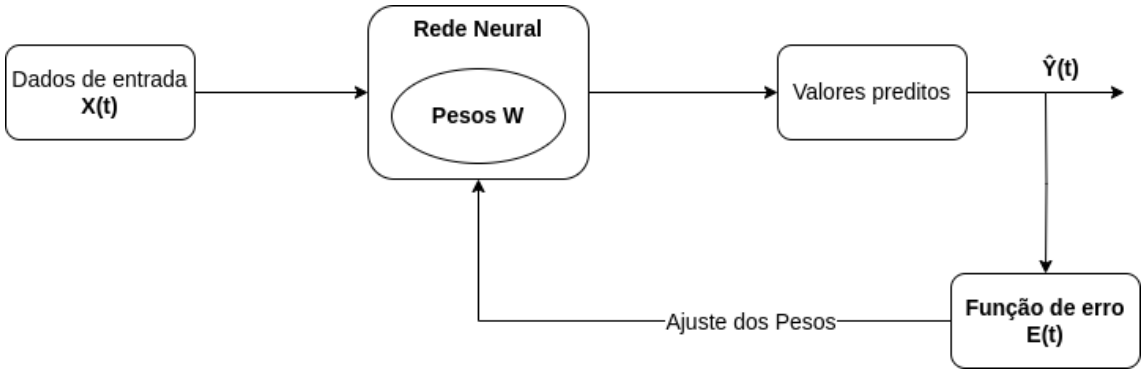
2.5.3 Algoritmos de Treinamento

Algoritmo de *BackPropagation*

Quando uma rede neural é treinada, basicamente estamos dando à ela um vetor contendo os dados de entrada, representado por x que são percorridos pelas camadas presentes na rede e esperamos uma saída $\hat{y}$ na última camada [22]. Ao obtermos o resultado da última camada, podemos comparar o resultado obtido $\hat{y}$ com os valores esperados y usando uma função custo, a fim de que a rede produza saídas que minimizam a função escolhida.

Para ajustar os pesos da rede de maneira a melhorar o desempenho, podemos utilizar o algoritmo de *back-propagation*, onde o principal objetivo é calcular o gradiente da função de erro com relação aos pesos que a rede possui e ajustá-los, a fim de obter as melhores predições possíveis [26], como mostra o fluxograma da Figura 2.19.

Figura 2.19: Fluxograma do algoritmo de *backpropagation*.



Geralmente, os valores iniciais dos pesos W são aleatórios. Depois a saída $\hat{Y}$ e o erro E são calculados, então as derivadas parciais de E em relação à W são obtidas através da expressão 2.23 e pela regra da cadeia [27]. Se ao aumentar os pesos implica em um erro maior, os pesos são ajustados para um valor menor, caso contrário, os pesos são ajustados para um valor maior [26].

$$\frac{\partial E}{\partial W_{i,j}} \tag{2.23}$$

onde $W_{i,j}$ denotam todos os vetores dos pesos sinápticos presentes na rede.

Gradient Descent

O *Gradient Descent* é uma maneira de minimizar a função custo $J(\theta)$ atualizando os parâmetros θ na direção oposta à derivada de $J(\theta)$ em relação aos pesos W. A taxa de aprendizado η determina o tamanho do passo que o algoritmo realiza a cada iteração em direção ao mínimo local [28]. Existem variações do *gradient descent* e geralmente é necessário tomar uma decisão se a taxa de aprendizado será pequena, aumentando a precisão de atualização dos pesos da rede ou se usamos uma taxa maior, com menos precisão, porém com uma velocidade de atualização maior.

Batch Gradient Descent

O *Batch Gradient Descent* calcula a derivada da função custo em relação aos pesos W em todo o conjunto de dados de treino, e então os pesos são atualizados pela seguinte relação 2.24 [28]:

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{2.24}$$

Como para cada época realizamos a derivada de todo o *dataset* para fazer apenas uma atualização, o *batch gradient descent* pode ser lento. Depois que o gradiente é calculado, atualizamos o parâmetro na direção da tangente multiplicando pelo tamanho do passo definido pela taxa de aprendizado [28].

Stochastic Gradient Descent

Ao contrário do *Batch Gradient Descent*, o *Stochastic Gradient Descent (SGD)* atualiza os parâmetros para cada exemplo dos dados de treino, isso é, para o exemplo x^i e para o rótulo y^i, como mostra a relação 2.25:

$$\theta = \theta - \eta \nabla_\theta J(\theta; x^i; y^i) \tag{2.25}$$

O *Batch Gradient Descent* calcula derivadas de forma reduntante, já que para toda época, a derivada é calculada novamente. Com o SGD, conseguimos eliminar essa redundância e por consequência a velocidade de atualização convergência do algoritmo é maior. Porém, o SGD apresenta alta variância [28].

Adaptive Moment Estimation (Adam)

O *Adam* é uma extensão do algoritmo de otimização *sthocastic gradient descent (SGD)* e pode ser utilizado para fazer a atualização dos pesos sinápticos de forma iterativa através dos dados de treino [29]. Porém, ele se mostrou ser mais eficiente e rápido de implementar.

A diferença desse algoritmo para o SGD, é que o *Adam* calcula diferentes taxas de aprendizado adaptativas para os diferentes parâmetros da rede, melhorando a performance do Algoritmo com features esparsas [29]. O algoritmo é baseado em duas variações do SGD, que são o *Adaptive Gradient Algorithm (AdaGrad)* e o *Root Mean Square Propagation (RMSProp* [29] .

A regra de atualização de parâmetros do *Adam* é definida pela equação 2.26 [28].

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \tag{2.26}$$

onde $\hat{v}_t$ e $\hat{m}_t$ são o momento e a variância do gradiente, definidos pelas equações 2.27 e 2.28, respectivamente.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.27}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_{12}^t} \tag{2.28}$$

A vantagem de se utilizar um algortimo de otimização como o Adam, é que não há a necessidade de se preocupar com o valor da taxa de aprendizado, pois ela é definida de forma dinâmica, além de apresentar melhores resultados para dados esparsos [28]

2.6 Redes Neurais Convolucionais

As redes neurais convolucionais (*Convolutional Neural Networks - CNN*) são um tipo de rede neural onde o seu foco é o processamento de dados do tipo matricial, como por exemplo, uma imagem. Para isso, a rede funciona primeiro identificando padrões mais simples, como contornos, linhas e curvas, e depois reconhece padrões mais complexos de uma imagem, como faces e objetos.

Para identificar esses contornos em imagens, a Rede Neural Convolucional combina dois tipos de camadas escondidas, as camadas de convolução e as camadas de *pooling* [20]. As camadas de convolução utilizam banco de filtros para realizar a operação da convolução entre o filtro e a imagem que está sendo analisada, com o objetivo de extrair e aprender características importantes dos dados de entrada. Um exemplo da operação de convolução pode ser representado pela operação entre a matriz 2.29 e o filtro 2.30, resultando em 2.31.

$$\text{Imagem original} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ k & k & l \end{bmatrix} \tag{2.29}$$

$$\text{Filtro de convolução} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \tag{2.30}$$

$$\text{Imagem convoluida} = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix} \tag{2.31}$$

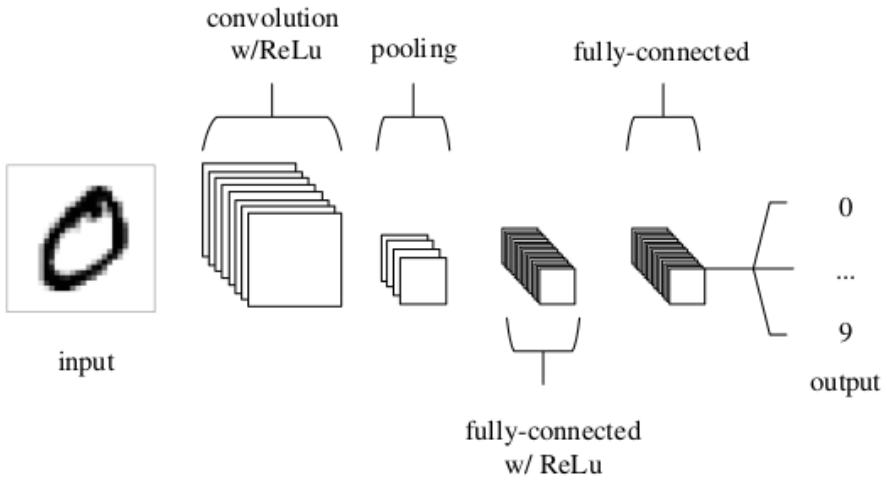
Após a operação de convolução, é comum aplicar uma função de ativação, por exemplo a ReLU, para diminuir a redundância de informação e aumentar a eficiência de processamento, já que a função zera todas as entradas negativas e mantém os valores das entradas positivas 2.20.

Além da camada de convolução, o algoritmo apresenta uma camada de *pooling*, onde o principal objetivo é reduzir a dimensionalidade da imagem e selecionar características mais importantes de porções da matriz. Geralmente, essas camadas são usadas após o uso das camadas de convolução. Uma das formas de realizar o processamento é chamado de *max pooling*, onde o maior valor de seções de uma matriz são selecionados, como ilustra o esquema 2.32, ou seja, cada elemento da matriz resultante é o valor máximo dos elementos em uma janela 2 X 2.

$$\text{Max pool} \begin{bmatrix} 1 & 1 & 4 & 7 \\ 3 & 1 & 4 & 8 \\ 4 & 3 & 5 & 10 \\ 7 & 0 & 6 & 11 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 8 \\ 7 & 11 \end{bmatrix} \tag{2.32}$$

Portanto, a arquitetura da CNN é composta por uma camada convolucional, onde filtros serão aplicados na imagem, uma camada de *pooling* que é responsável por diminuir a dimensionalidade da imagem e o número de *features* e a camada totalmente conectada responsável por ajustar os pesos e realizar a predição da rede, como mostra a Figura 2.20.

Figura 2.20: Arquitetura de uma Rede Neural Convolucional



Fonte: adaptado de [30]

2.7 Redes Neurais Recorrentes

As redes neurais recorrentes (*Recurrent Neural Networks* - RNN) é um tipo de arquitetura das redes neurais que é capaz de processar dados sequenciais e ou de séries temporais, sendo que, geralmente, esses dados apresentam uma certa dependência com amostras passadas e, por isso, o algoritmo é capaz de armazenar e processar essas informações e realizar predições futuras. É uma arquitetura de Rede Neural amplamente utilizada em processamento de documentos ou de textos, séries temporais financeiras, como por exemplo a predição nos valores de ações do mercado financeiro, no processamento de áudio, entre outras áreas da tecnologia [20].

Diferentemente das Redes Neurais tradicionais, os dados de entrada de uma RNN são sequências que podem variar de tamanho e são definidas pelo vetor $X = \{X_1, X_2, X_3, ..., X_L\}$, sendo que a saída Y também pode ser uma sequência [20]. A equação que define a saída das camadas escondidas é dada por 2.33.

$$H_t = \phi(X_T.W_{hh} + H_{t-1}.W_{hh} + b_h) \tag{2.33}$$

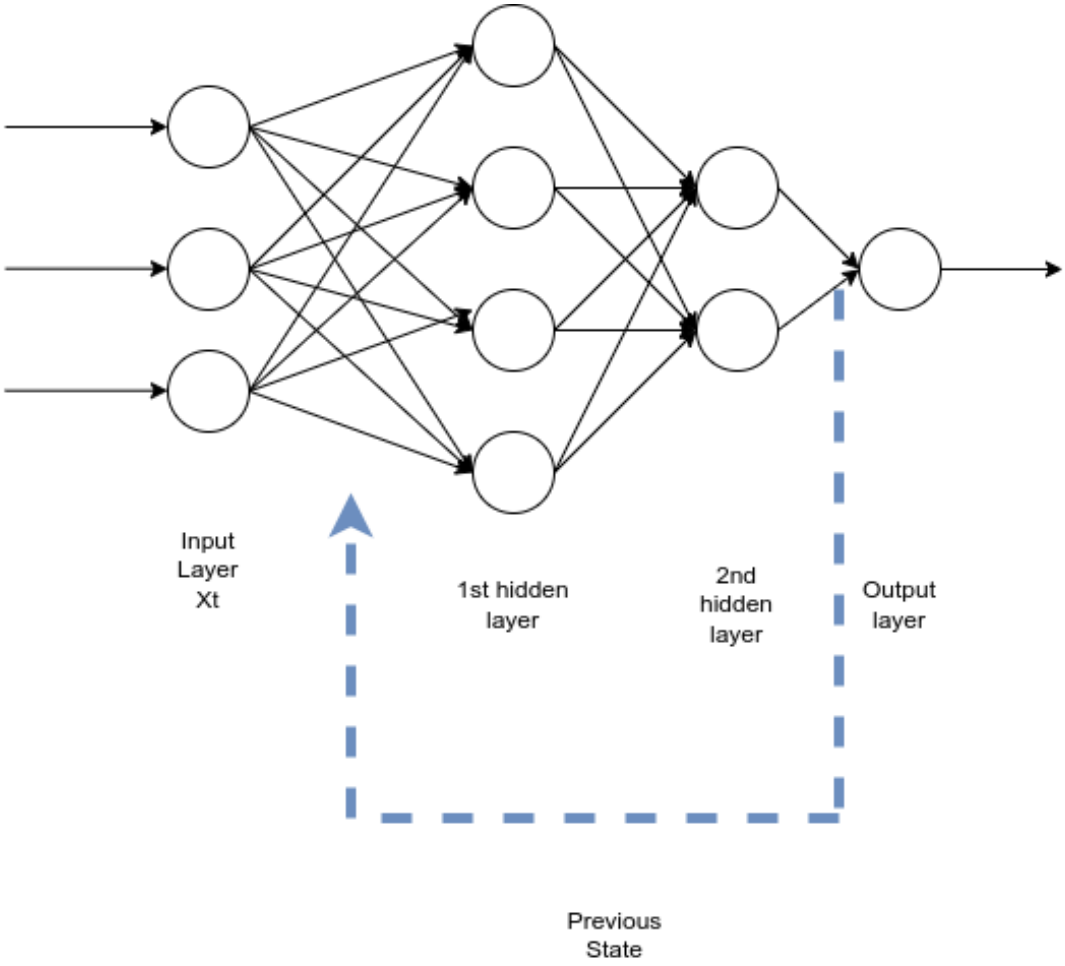
Onde H_t é a saída das interações das camadas escondidas, X_t é a entrada no instante t, H_{t-1} é a saída do estado no instante $t-1$, W_{hh} é a matriz de pesos que são ajustados pelo algoritmo de *backpropagation* e b_h é o bias da rede.

Já a saída da rede é dada pela equação 2.34 sendo O_t o valor predito pela rede na iteração t.

$$O_t = \phi_0(H_t.W_{h0} + b_0) \tag{2.34}$$

O esquemático geral de uma RNN pode ser representado pela Figura 2.21.

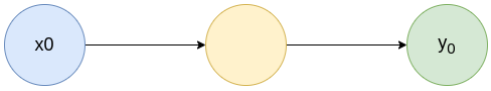
Figura 2.21: Arquitetura de uma Rede Neural Recorrente



Além do esquemático geral de uma RNN ilustado pela Figura 2.21, existem diferentes tipos de arquiteturas que podem ser implementadas:

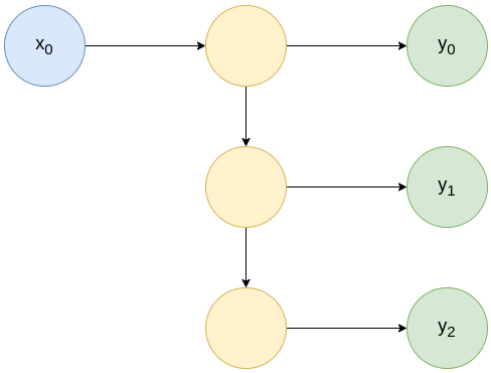
- *One to One*: - Essa é uma arquitetura mais tradicional que mapeia uma entrada $X = \{x_0\}$ em uma saída $Y = \{y_0\}$, como mostra a Figura 2.22.

Figura 2.22: Esquemático RNN one to one.



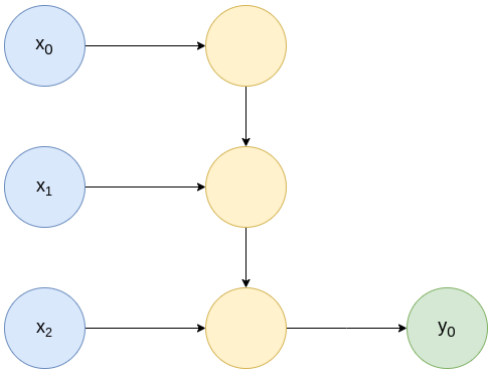
- *One to Many:* - A abordagem um para muitos processa uma entrada da rede $X = \{x_0\}$ e retorna mais de uma saída $Y = \{y_0, y_1, y_2\}$, representado pela Figura 2.23.

Figura 2.23: Esquemático RNN one to many.



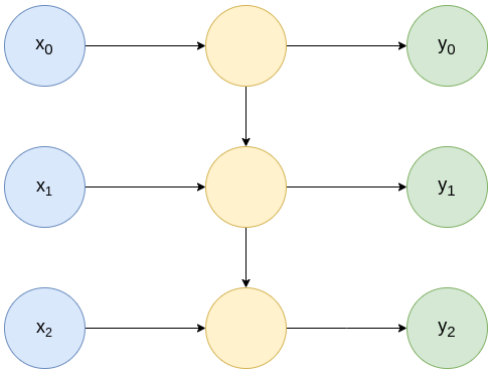
- *Many to One:* - Nessa arquitetura, múltiplas entradas $X = \{x_0, x_1, x_2\}$ são mapeadas em apenas uma saída $Y = \{y_0\}$, como mostra a Figura 2.24.

Figura 2.24: Esquemático RNN many to one.



- *Many to Many:* - Aqui há um mapeamento de várias entradas $X = \{x_0, x_1, x_2\}$ para várias saídas $Y = \{y_0, y_1, y_2\}$, como mostra a Figura 2.25.

Figura 2.25: Esquemático RNN many to many.



3. *Metodologia*

3.1 Base de dados

Dado que o objetivo do projeto é separar sinal de interesse de ruídos externos e avaliar o efeito do mascaramento na extração de um sinal imerso em ruído, foram selecionados dois *datasets*, sendo um composto por falas e outro por ruídos para serem combinados e tornar-se possível realizar máscaras IBM e IRM.

O *dataset* de ruídos selecionado foi o *PNL 100 Nonspeech Sounds* coletado por *Guoning Hu* e *DeLiang Wang* durante seu trabalho de dissertação [31]. Essa base de dados contém 100 ruídos diferentes dos seguintes tipos: Barulho de multidão, ruídos de máquinas, alarmes e sirenes, tráfego e ruído de carros, sons de animais, som de água, barulho de vento, sinos, sons de pessoas tossindo, palmas e etc. A Figura 3.1 representa formato do sinal no tempo discreto de um ruído de multidão e seu respectivo espectrograma. Já a Figura 3.2 representa o ruído de uma pessoa tossindo.

Figura 3.1: Sinal no tempo discreto e espectrograma do ruído de uma multidão.

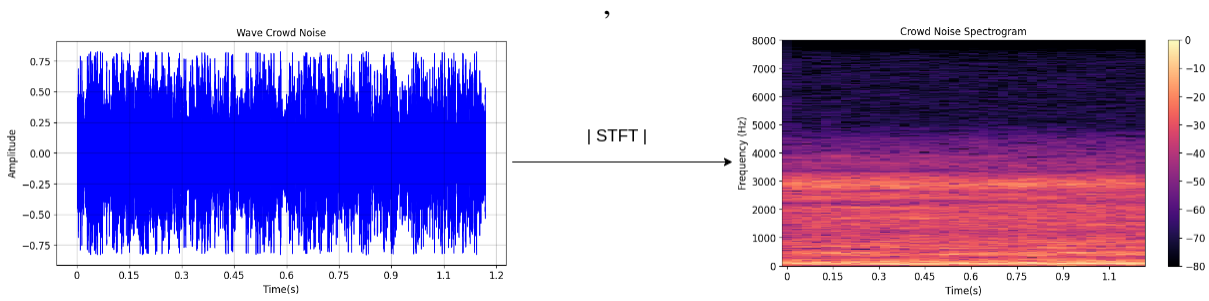
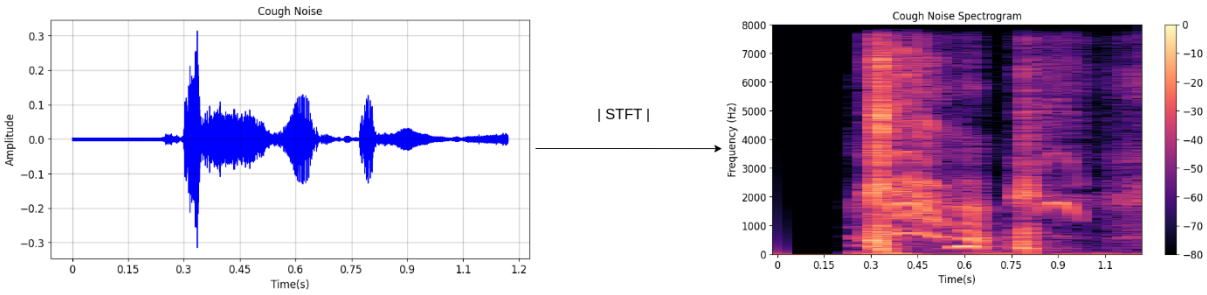
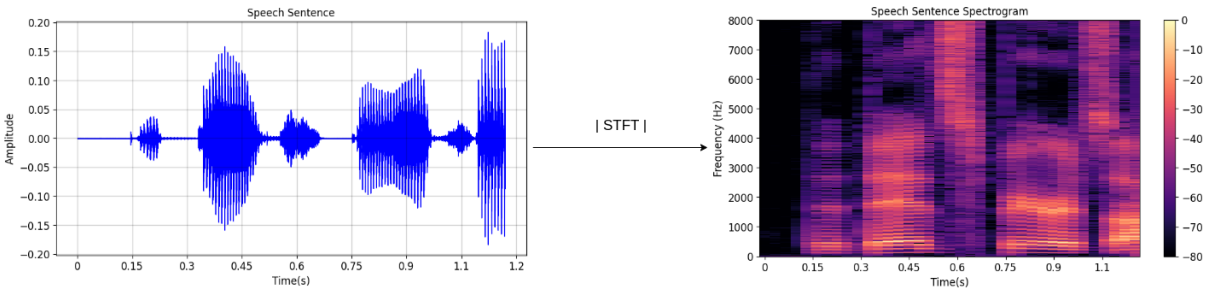


Figura 3.2: Sinal no tempo discreto e espectrograma do ruído de uma pessoa tossindo.



O banco de dados de sinais de voz é o TIMIT [32], proposto inicialmente para estudos acústicos-fonéticos, avaliação de sistemas de reconhecimento automático de fala e foi elaborado pelo Instituto de Tecnologia de Massachusetts, pelo SRI Internacional (SRI) e pela *Texas Instruments, Inc.*(TI). Trata-se de uma base de dados com 6300 sentenças divida em 630 falantes dos oito maiores dialetos do Inglês Americano, portanto cada falante reproduziu 10 diferentes áudios, todos gravados a uma taxa de amostragem igual a 16kHz. A Figura 3.3 representa o formato de um sinal no tempo discreto e o espectograma de uma pessoa falando.

Figura 3.3: Sinal no tempo discreto e espectrograma da fala de uma pessoa.



Para realizar os mascaramentos e avaliar as performances tanto das máscaras quanto da representação tempo-frequencial (espectrograma), foram selecionados de forma aleatória 11 áudios de cada *dataset* e todos foram combinados entre si, totalizando 121 exemplos de misturas. Ademais, para cada conjunto de 121 misturas, foi-se variando os valores de *Signal-to-Noise Ratio* (SNR) entre -20 dB até 30 dB, com passos de 5 dB, com o objetivo de avaliar posteriormente as notas de PESQ e STOI de sinais pós processados pelas máscaras em função do SNR. No total, o conjunto de treinamento é composto por 1331 trechos de áudio, que correspondem a misturas de fala e ruído com diferentes relações sinal ruído.

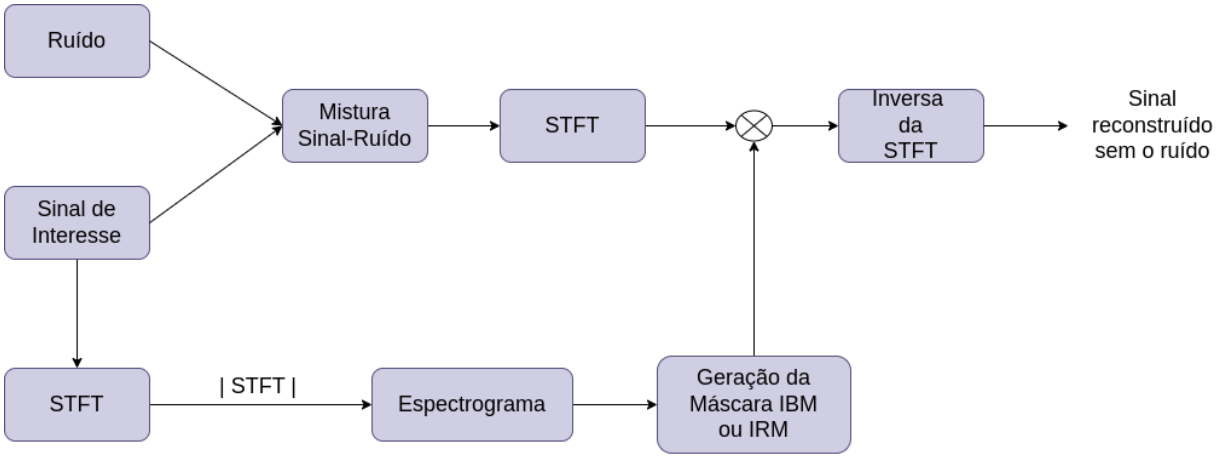
A fim de tornar todo o processo de mascaramento melhor exemplificado, nas próximas

duas seções será ilustrado e discutido toda a sequência de processamento utilizado para a obtenção das máscaras e extração do sinal de interesse, a partir de uma mixagem com sinal-ruído, tanto para o Espectrograma e cogleagrama.

3.2 Mascaramento a partir do Espectrograma

A estrutura geral para realizar o mascaramento em uma mistura sinal-ruído a partir do espectrograma está sendo representada pela Figura 3.4.

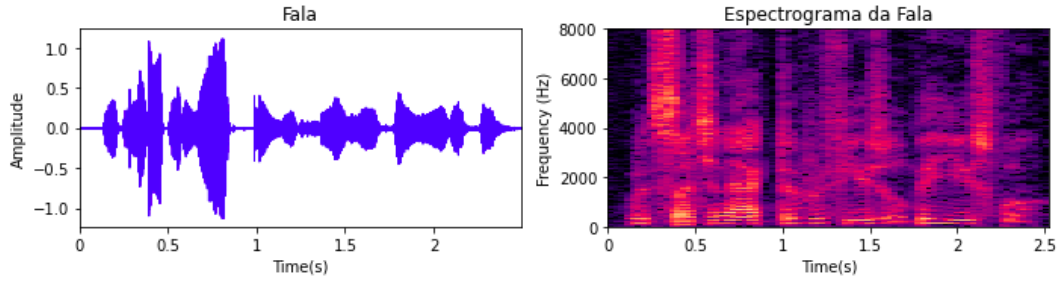
Figura 3.4: Sequência de processamento para mascarar ruídos externos de uma mixagem sinal-ruído a partir do espectrograma.



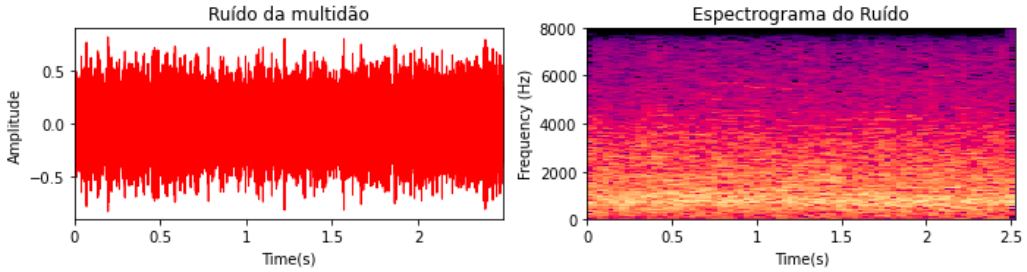
Os trechos processados são compostos por uma mistura do sinal de interesse e um sinal de ruído, ambos a uma taxa de amostragem de 16kHz. Nesse momento, como as misturas foram geradas artificialmente, temos acesso aos sinais originais isoladamente, o que nos permite obter o espectrograma de um sinal "limpo", sem ser contaminado com o ruído. Então, a operação da STFT deve ser aplicada tanto para a fala original (sinal limpo) quanto para o mix obtido (mistura fala-ruído), para isso a biblioteca em Python, Librosa [33], foi utilizada. A Figura 3.5 mostra as formas de ondas do sinais da fala, ruído da multidão, fala + ruído e seus respectivos espectrogramas.

Com o espectrograma da fala, o próximo passo é extrair a máscara Binária ou a *Ratio Mask* a partir do mesmo, como mostra a Figura 3.6. Tendo então a máscara (matriz de valores reais) e a STFT da mixagem em mãos, é possível multiplicar uma pela outra e o resultado será a STFT do mix sinal-ruído com a maioria das unidades tempo-frequenciais sendo composta

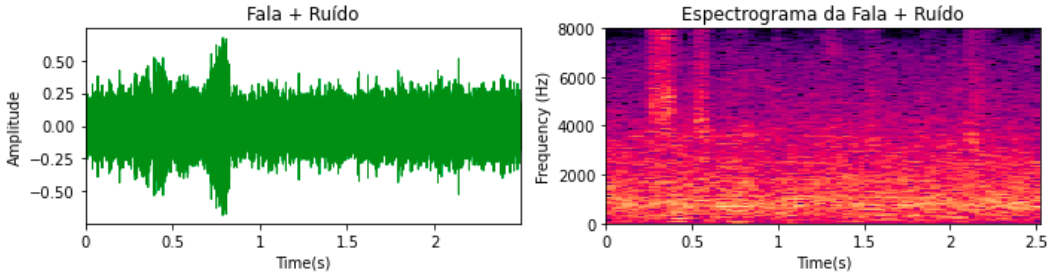
Figura 3.5: Sinais de onda e espectrograma.



(a) Sinal da fala e seu espectrograma.



(b) Sinal do ruído e seu espectrograma



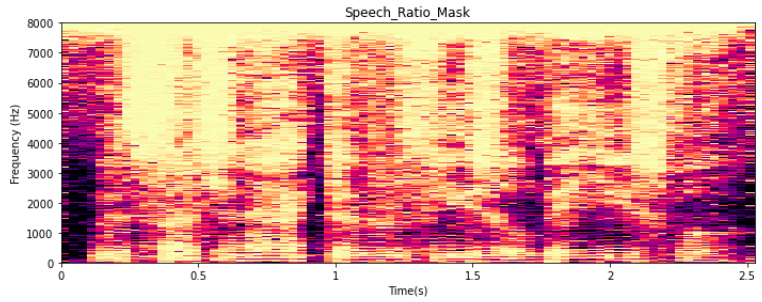
(c) Sinal da mistura e seu espectrograma

pela energia do sinal de interesse. A Figura 3.7 mostra o espectrograma gerado a partir da multiplicação da STFT da fala-ruído com as máscaras da fala.

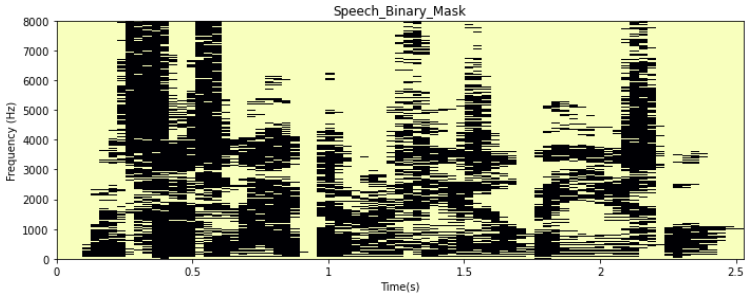
Por fim, é preciso aplicar a operação inversa da STFT para ter o sinal reconstruído no domínio do tempo. A Figura 3.8 compara o sinal da fala original com o sinal da mesma sentença que sofreu processamento do tipo *ratio mask* e máscara binária a partir do espectrograma.

Observa-se na Figura 3.8 que a forma de onda da fala que sofreu mascaramento do tipo *ratio mask* se aproxima muito bem ao da fala original sem ruído. Apesar da máscara binária ter um desempenho inferior à *ratio mask*, ainda é possível perceber uma boa aproximação à fala original sem ruído, indicando uma boa limpeza na mistura sinal-ruído, ilustrada pela Figura 3.8b.

Figura 3.6: *Ratio Mask* e Máscara Binária do espectrograma da fala representado pela Figura 3.5a

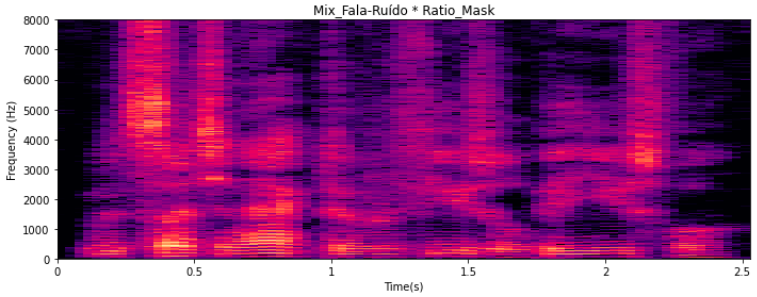


(a) Ratio Mask da fala.

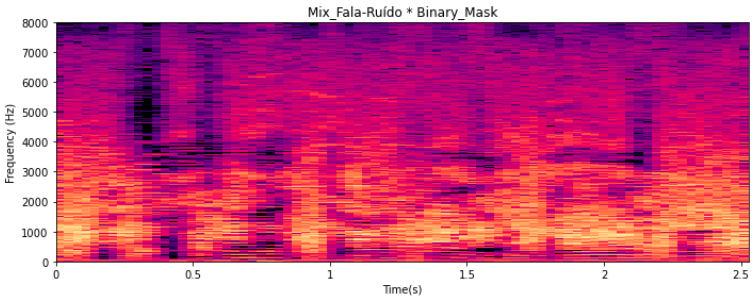


(b) Máscara binária da fala.

Figura 3.7: Espectrogramas da Fala-Ruído pós Mascaramento.



(a) Espectrograma da fala extraída a partir da mistura fala-ruído através da ratio mask.

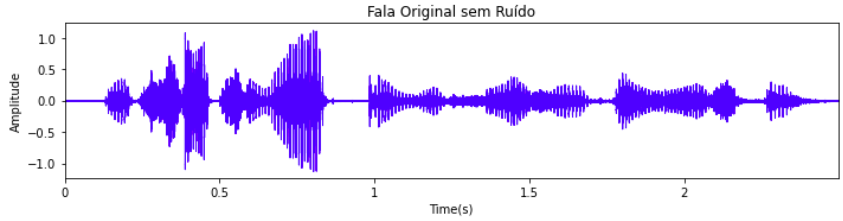


(b) Espectrograma da fala extraída a partir da mistura fala-ruído através da máscara binária.

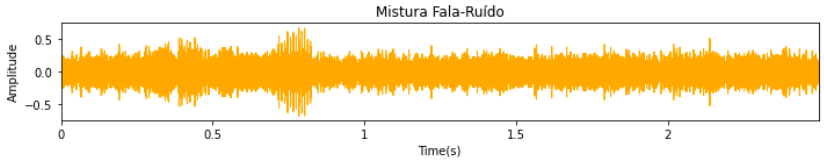
3.3 Mascaramento a partir do Cocleagrama

A estrutura geral para atingir a representação tempo-frequencial do cogleagrama exige uma série de etapas que estão representadas no fluxograma da Figura 3.9.

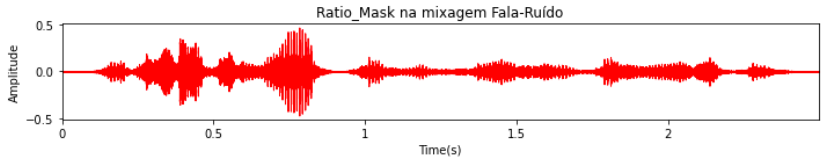
Figura 3.8: Comparação da forma de onda do sinal de fala sem ruído com o sinal de fala reconstruiído pós mascaramento do tipo *ratio* e binário a partir do espectrograma.



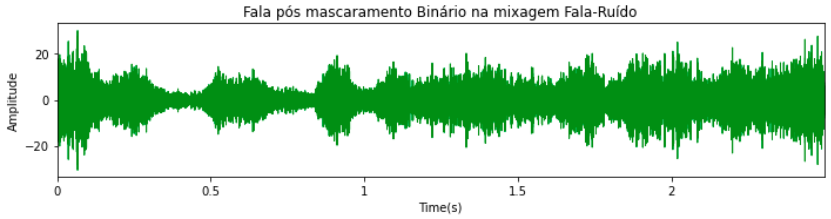
(a) Fala original sem ruído.



(b) Mistura Fala-Ruído.

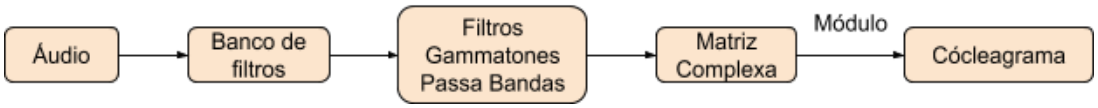


(c) Fala extraída a partir da mistura fala-ruído usando a ratio mask.



(d) Fala extraída a partir da mistura fala-ruído usando a máscara binária.

Figura 3.9: Fluxograma para obtenção do cogleagrama a partir de um sinal.

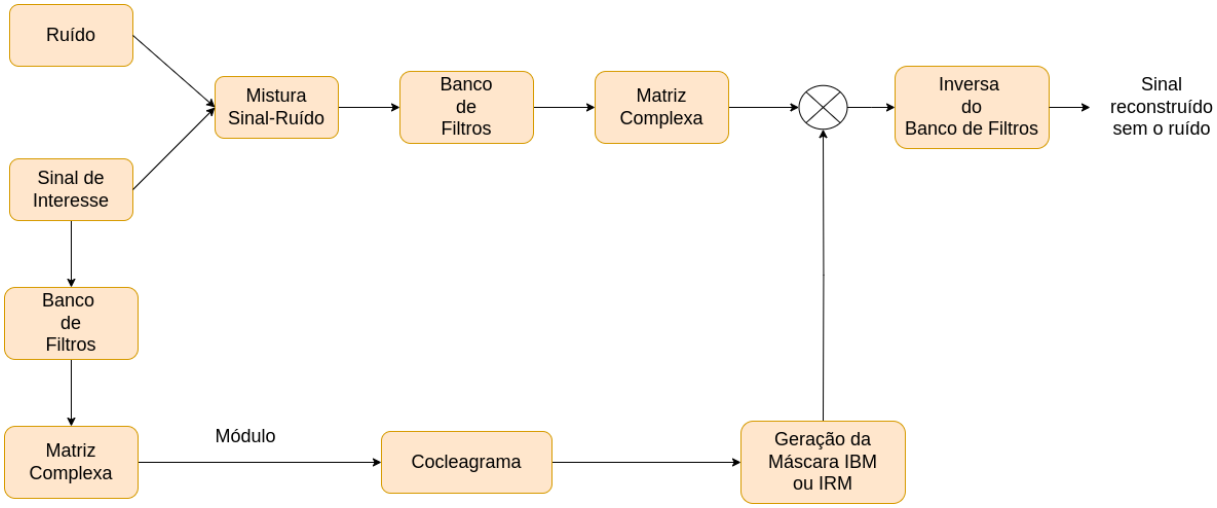


O primeiro passo é definir um banco de filtros que realizará o processamento do sinal. O banco de filtros escolhido é composto por 66 filtros *gammatone* passa bandas, trata-se de um filtro linear onde a resposta ao impulso é o produto de uma distribuição gama e tom senoidal, cuja largura é determinada pela função *Equivalent Rectangular Bandwidth* (ERB) [11]. Esse conjunto de filtros irá dividir o sinal de entrada em bandas espectrais e a saída de cada filtro é chamada de canal, onde todos juntos representam o sinal na forma Tempo-Frequência [12] e trata-se de uma matriz complexa com informações de fase e magnitude do sinal. Posterior-

mente, aplica-se a operação de módulo na matriz, atingindo dessa forma o cocleagrama. Para reconstruir o sinal no domínio do tempo, basta passar a matriz complexa no inverso do banco de filtros.

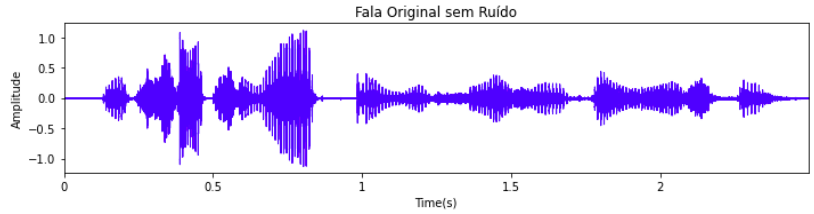
Sabendo como alcançar o cocleagrama de um sinal, os processos de geração das máscaras e a reconstrução do sinal se tornam análogos ao do espectrograma e está representado pela Figura 3.10.

Figura 3.10: Fluxograma para mascarar ruídos externos de uma mixagem sinal-ruído a partir do cogleagrama.

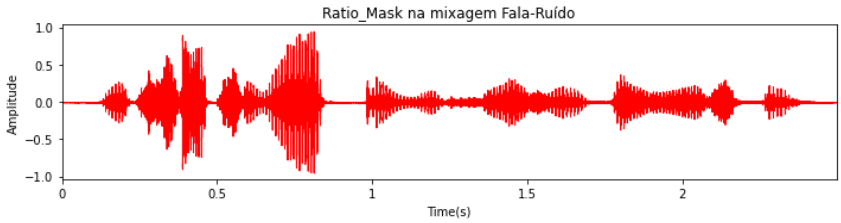


A Figura 3.11 compara o sinal da fala original com o sinal da mesma sentença pós mascaramento tempo-frequencial dos tipos *ratio mask* e máscara binária a partir do cocleagrama, sendo que os cocleagramas pós mascaramento estão representados nas Figuras 3.12a e 3.12b respectivamente.

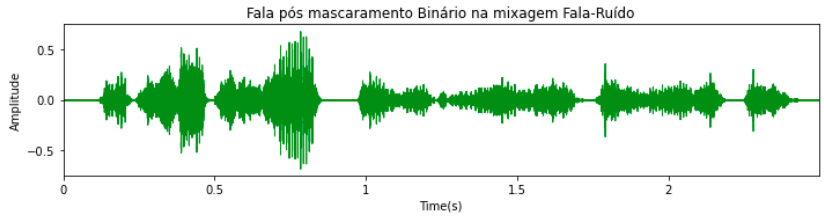
Figura 3.11: Comparação da forma de onda do sinal de fala sem ruído com o sinal de fala reconstruído pós mascaramento do tipo *ratio* e binário a partir do cogleagrama.



(a) Fala original sem ruído.

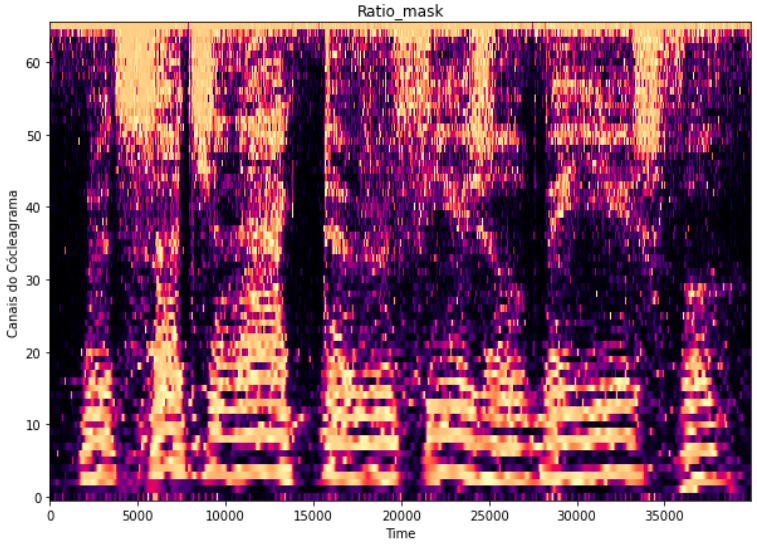


(b) Fala extraída a partir da mistura fala-ruído usando a ratio mask.

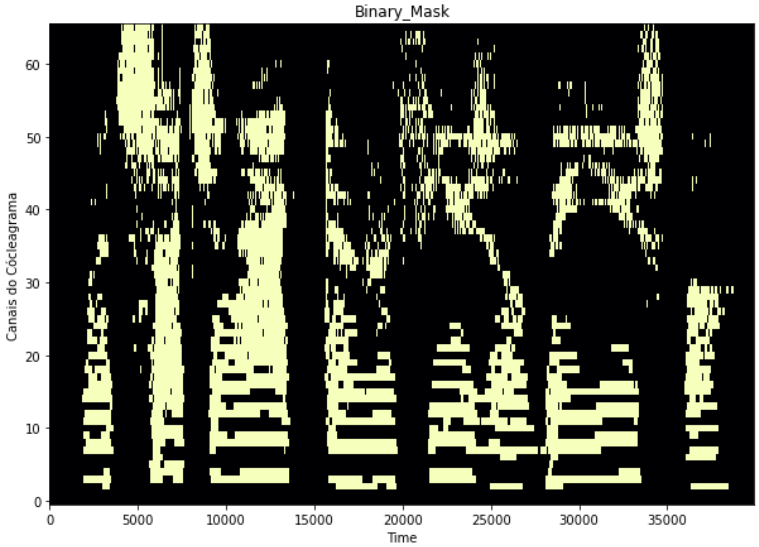


(c) Fala extraída a partir da mistura fala-ruído usando a máscara binária.

Figura 3.12: Máscaras utilizadas para extrair a fala da mistura sinal-ruído.



(a) cogleagrama da *Ratio Mask*.



(b) cogleagrama da Máscara Binária.

Nota-se que tanto a extração do sinal original a partir do mascaramento binário quanto a partir da *ratio mask* apresentam formas de ondas muito parecidas com a original. Ademais, na seção de resultados, comparativos entre Espectrograma e cogleagrama são feitos, considerando métricas de PESQ e STOI a fim de analisar e decidir qual dos dois será uma boa escolha para dar continuidade no projeto.

3.4 Processamento dos dados de áudio

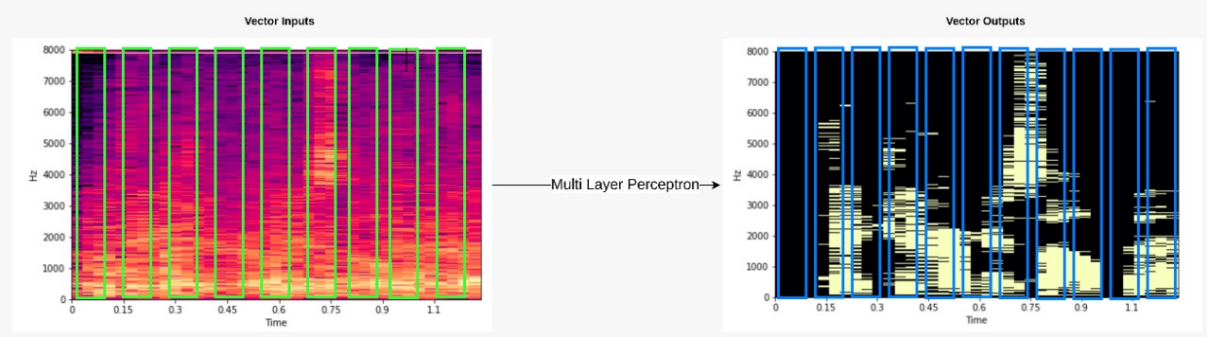
Quando usamos o espectrograma para analisar o áudio, é possível usar algortimos de aprendizado de máquina para classificar cada *bin* de tempo-frequência no espectrograma como sendo parte do sinal de interesse ou ruído. Portanto, é possível definir a saída do algoritmo como uma máscara binária.

Para utilizar os áudios da Base de dados e processá-los de forma que seja possível treinar um algoritmo, 20 áudios de fala e 81 sons de ruído foram selecionados e mixados aleatoriamente, produzindo 1620 áudios no total.

Então, para cada áudio, a representação tempo-frequência é obtida por meio da *STFT*, considerando janelas de 2.048 amostras, com *hop length* igual a 512. Assim, cada áudio foi transformado em uma matriz com tamanho 1025×37, representando o espectrograma, como mostra a Figura 3.13. Também é necessário obter o espectrograma da fala sem ruído, a partir da qual a *IBM* é obtida aplicando-se um limiar aos valores do espectrograma, também ilustrado pela Figura 3.13. Além disso, os coeficientes *MFCCs* são obtidos a partir da Transformada Discreta de Cosseno, resultando em 13 recursos adicionais para cada índice de tempo do espectrograma.

Então, podemos aproveitar as matrizes geradas a partir dos espectrogramas e realizar a classificação de cada *bin* de tempo-frequência, onde os vetores de entradas representam a mistura fala e ruído, representado na Figura 3.13 pelos retângulos verdes, e os de saída, as máscaras binárias, representado na Figura 3.13 pelos retângulos azuis. Portanto, o vetor de entrada é composto do conteúdo de frequência para um determinado índice temporal m e 13 coeficientes MFCCs associados ao mesmo índice temporal, definindo assim um vetor de tamanho 1038. O vetor de saída corresponde à máscara binária ideal associada ao mesmo índice de temporal m.

Figura 3.13: Vetores de entrada e saída da rede *MLP*.



4. *Resultados e discussão dos resultados*

4.1 Análise de desempenho da rede Multilayer Perceptron

Neste projeto, foi implementado uma rede neural do tipo Multilayer Perceptron cuja arquitetura para realizar o mapeamento entre a entrada e a saída possui quatro camadas, uma de entrada, duas camadas ocultas e uma camada de saída, caracterizando uma Rede Neural Profunda. A primeira camada possui 1038 neurônios que representam o tamanho do vetor de entrada, as duas camadas ocultas têm seu número de neurônios variando entre 25 e 100 (essa variação foi utilizada para obtermos resultados de performance de cada combinação), cada uma com tangente hiperbólica como função de ativação e a última possui 1025 neurônios correspondentes ao tamanho de um vetor de máscara binária, com a função de ativação sigmóide.

A rede foi treinada usando a função custo entropia cruzada onde 1 representa um *bin* de interesse, fala e 0 o ruído.

O treinamento considerou 1620 amostras de áudio, resultando em 59.940 dados de entrada, sendo eles vetores de dimensão 1038×1 (representando a combinação do áudio com os coeficientes MFCCs) e 59.940 vetores de saída, com dimensões igual a 1025×1 (representando a máscara binária ideal), com 2 camadas intermediárias e tendo a tangente hiperbólica como função de ativação. O treinamento foi realizado por meio do algoritmo de otimização *Adam*, com taxa de aprendizado $\eta = 0,0007$, $beta_1 = 0,9$ e $beta_2 = 0,999$ (obtido após simulações preliminares), além 600 épocas de treinamento.

Os dados de testes foram compostos por 300 áudios, que foram utilizados apenas para avaliar a performance da rede neural. Para atingir essa quantidade de dados, selecionamos de forma aleatório 89 áudios de fala da base de dados do TIMIT, e 8 áudios de ruído da base de dados *PNL Nonspeech Sound*, diferentes daqueles utilizados para treinar a rede. Então, todos foram misturados entre si, produzindo no total 712 exemplos. E portanto, 300 foram selecionados para avaliar a performance da rede neural.

Na Figura 4.1, temos o *STOI* obtido com a máscara binária estimada pela rede, considerando diferentes combinações na quantidade de neurônios das camadas intermediárias (variando de 25 até 100). De acordo com os resultados obtidos, o número de neurônios apresenta pouco impacto na métrica avaliada. Além disso, o gráfico da Figura 4.1 mostra que a rede neural conseguiu atingir bons resultados em todas as combinações. Por exemplo, se avaliarmos a configuração com 50 neurônios em cada camada, percebemos que a média de inteligibilidade é próxima a 0.7, o valor mínimo é próximo a 0.5, e o valor máximo próximo a 0.9, além de que 50% dos dados têm *STOI* entre 0.6 e 0.72, o que representa um bom resultado de inteligibilidade, pois a *IBM* consegue atingir valores de *STOI* entre 0.7 e 0.94, com média próxima a 0.83, como mostra a Figura 4.2.

Figura 4.1: Box-plot of STOI in function of the number of neurons.

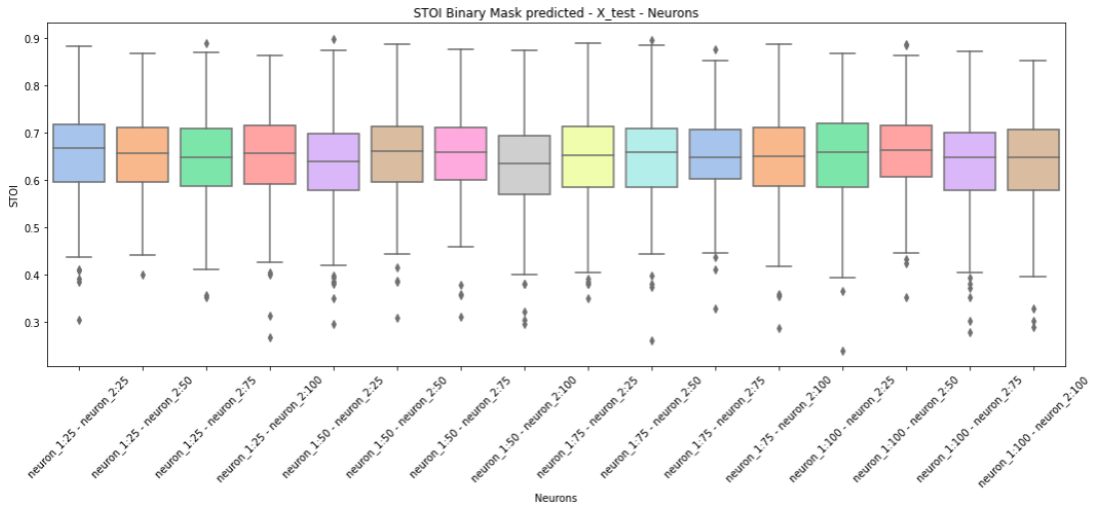
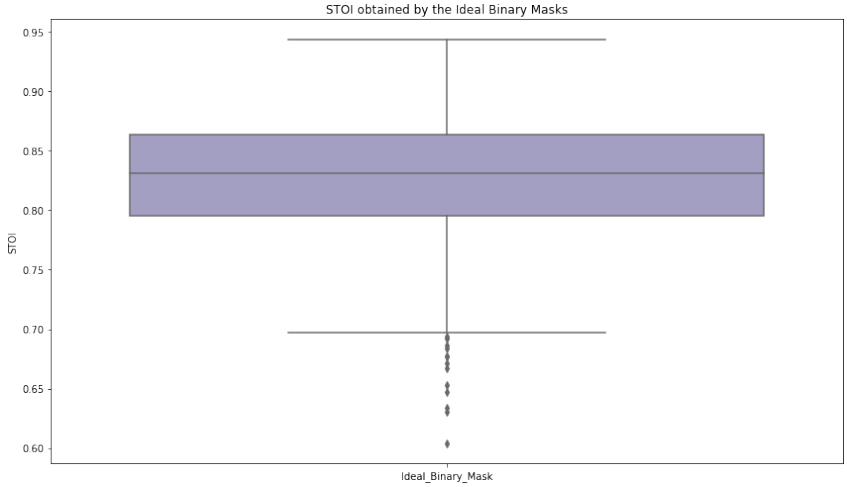


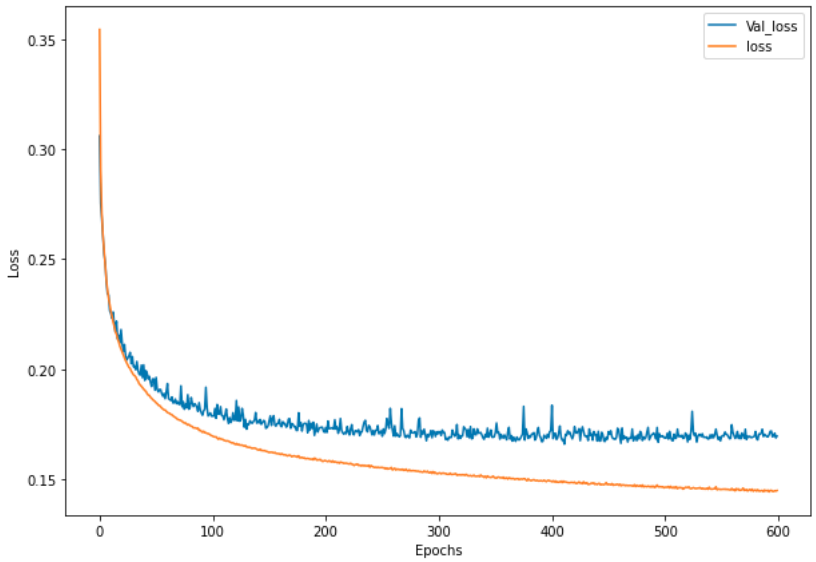
Figura 4.2: Box-plot of the Ideal Binary Masks with 50 neurons in each intermediate layers.



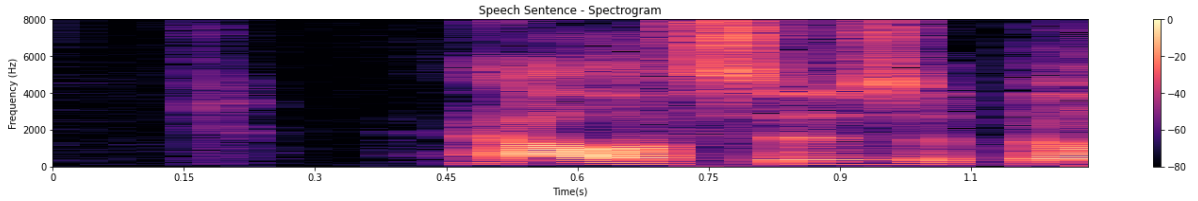
A Figura 4.3 mostra o comportamento da função custo para a configuração com 50

neurônios em cada camada intermediária. Podemos notar que a função custo tem uma queda a medida que o número de épocas aumenta, atingindo o menor valor na época 600. Podemos ainda notar, que ao plotarmos a função custo dos dados de validação, o mesmo apresenta também uma queda, portanto não ocorre *overfitting*, ou seja, a rede não está sendo enviesada pelos dados de treino e consegue generalizar para dados novos.

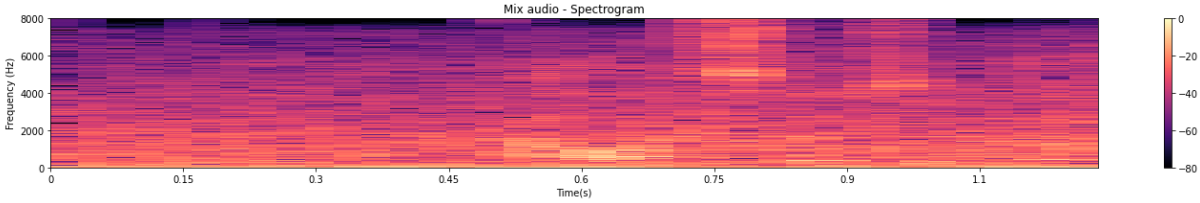
Figura 4.3: loss and validation loss function for the Neural Networks.



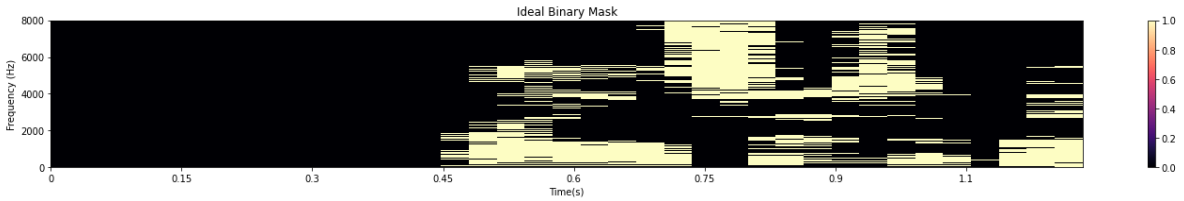
A Figura 4.4 representa um exemplo de uma máscara binária predita pela rede neural. Enquanto que a Figura 4.4a representa o espectrograma de um sinal de fala da base de dados, a Figura 4.4b mostra a sua versão com ruído, sua respectiva máscara binária ideal está sendo representada pela Figura 4.4c, e a máscara predita está sendo ilustrada pela Figura 4.4d. A máscara estimada é próxima a sua versão ideal e consegue capturar porções da fala do espectrograma (em 0.20s) que a IBM simplesmente ignorou. Essa mesma característica também foi observada em outros áudios.



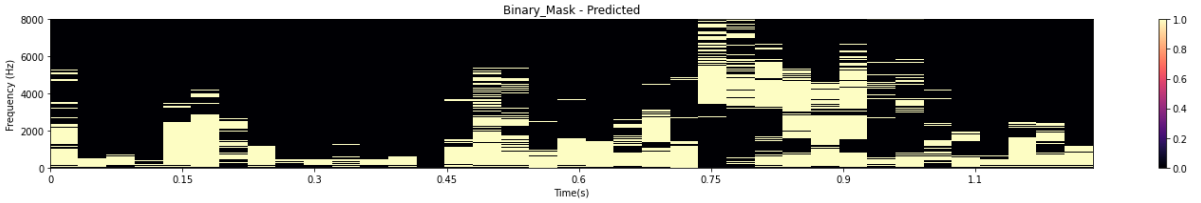
(a) Speech spectrogram.



(b) Speech spectrogram with noise.



(c) Ideal Binary mask for the speech



(d) Binary Mask predicted by Neural Networks.

Figura 4.4: Máscara binária ideal e máscara gerada pela rede MLP.

4.2 Análise de desempenho das Florestas Aleatórias

Com o objetivo de realizar comparações entre algoritmos de *Deep Learning* e de aprendizado de máquina mais clássicos, foi implementado o algoritmo de floresta aleatória no mesmo banco de dados que a rede *Multilayer Perceptron* para a predição de máscaras binárias. Para isso, foi utilizado o pacote *Scikit-Learn* do python [34]. Para a implementação, foi considerado a profundidade das árvores como sendo igual a 2 nós, para evitar overfitting e aumentar a velocidade de treinamento do algoritmo e foram utilizadas 100 árvores de decisão na floresta aleatória.

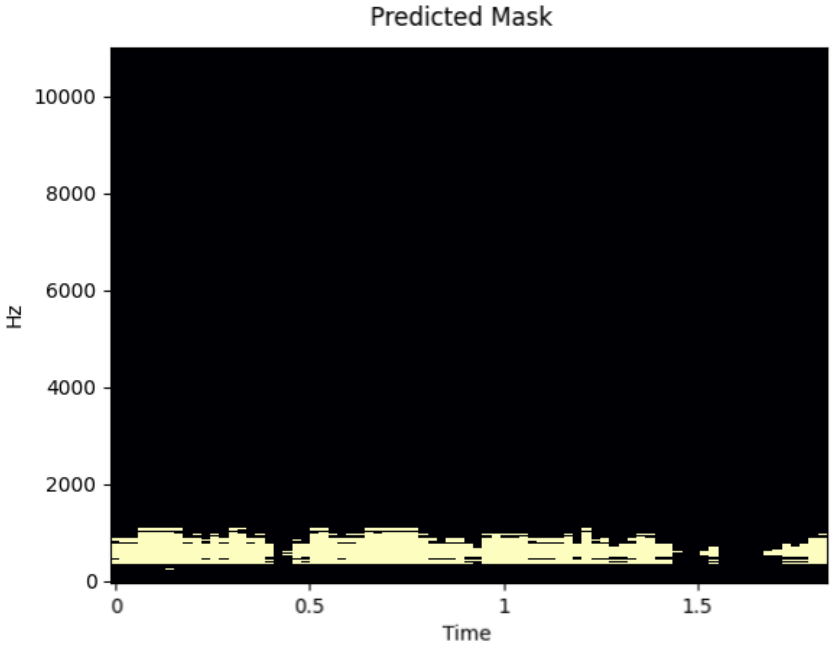
Para facilitar a comparação entre os diferentes algoritmos de classificação analisados nas próximas subseções, a Figura 4.9 apresenta os resultados obtidos no formato de *box-plot*.

Além disso, a variação escolhida para a MLP foi a configuração de 50 neurônios em cada camada escondida, conforme apresentado na Figura 4.1.

Como mostra a Figura 4.9, o desempenho da floresta aleatória em termos de STOI foi inferior ao da Multilayer Perceptron, pois a média do box-plot está próximo de 0.56 e o valor mínimo está perto de 0.35 o que indica um baixo desempenho do algoritmo e que resulta em baixas notas de inteligibilidade.

Um exemplo de uma máscara binária gerada pelo algoritmo da Floresta Aleatória está representada pela Figura 4.5. Observamos que a máscara binária gerada pelo algoritmo exclui estruturas importantes do espectrograma, preservando apenas componentes de baixa frequência.

Figura 4.5: Máscara binária gerada pela Floresta Aleatória



4.3 Análise de desempenho Máquinas de Vetores de Suporte

Um dos algortimos que também foi treinado com o propósito de fazer o comparativo com a MLP é o SVM, sendo esse um algoritmo capaz de processar dados e gerar predições tanto de problemas de regressão quanto de classifação. Na implementação, foi considerado o parâmetro de regularização $C = 0.1$, que controla o quão flexível é a margem do algoritmo, quanto menor esse valor, maiores são os erros de classificação, porém, maior será a capacidade de generalização e de predição de máscaras binárias. O parâmetro *random state* foi fixado com o valor 0, a fim de mantermos a reprodutibilidade dos resultados do modelo. Além disso, foi atribuído ao parâmetro *max iter* o valor de 2000, sendo esse um parâmetro que determina o número máximo de interações, atuando como um limitador no processo de convergência do algoritmo, para que ele não fique executando indefinidamente. Também foi aplicado a abordagem de classificação multirrótulo, dado que os dados a serem preditos pelo modelos são espectrogramas que possuem formato de matrizes.

Assim como na *MLP* e no algoritmo da Floresta Aleatória, os dados de treinamento consistem em ”pilhas”de espectrogramas de 1620 amostras de áudios no formato $(37, 1025)$, como explicado na seção 3.4, gerando no total 59.940 dados de entrada de dimensão 1025×1 e 59.940 vetores de saída, com dimensões iguais a 1025×1 (representando a máscara binária ideal).

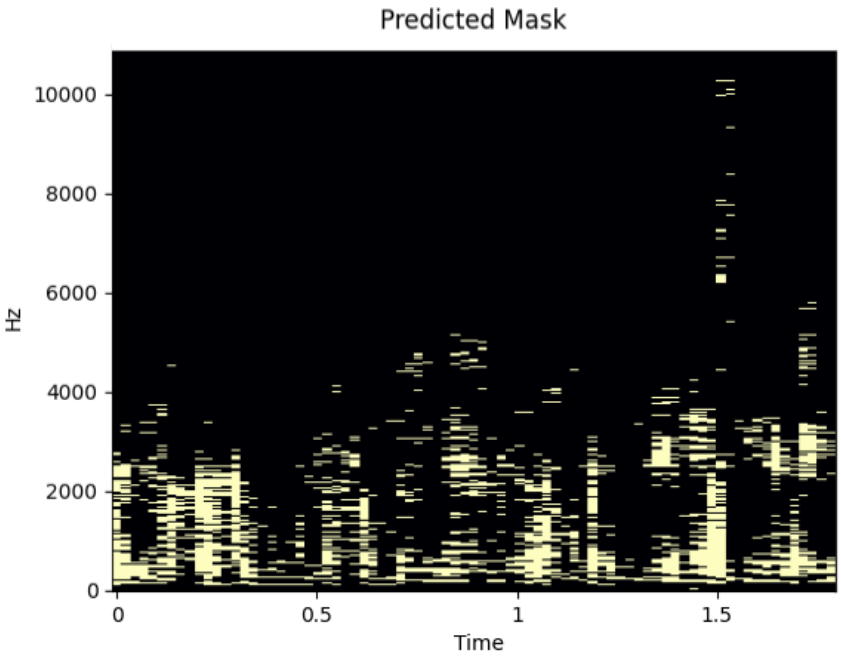
Pela Figura 4.9, podemos observar que a mediana do *boxplot* se encontra em torno de 0.62, o que evidencia um resultado inferior a *MLP*, porém, ao comparar o resultado com a Floresta Aleatória, as notas de STOI do SVM foram superiores.

A Figura 4.6 ilustra uma máscara gerada pelo SVM em uma mistura entre ruído e discurso de uma pessoa, sendo esse um áudio totalmente novo para o algoritmo. Em comparação com a máscara da Floresta Aleatória, representada pela Figura 4.5, o algoritmo do SVM mantém estruturas importantes do espectrograma, tirando de fundo o ruído do áudio e refletindo as notas de STOI superiores ao Floresta Aleatória.

Como comentado na seção 2.4.2, o SVM apresenta uma extensão para problema de separação não linear, que é através de Kernels. Por isso, como uma forma de explorar diferentes abordagem para a geração de máscaras binárias, foi implementado uma versão do SVM que utiliza do conceito de Kernel, sendo que a função utilizada foi a *Radial Basis Function* (RBF).

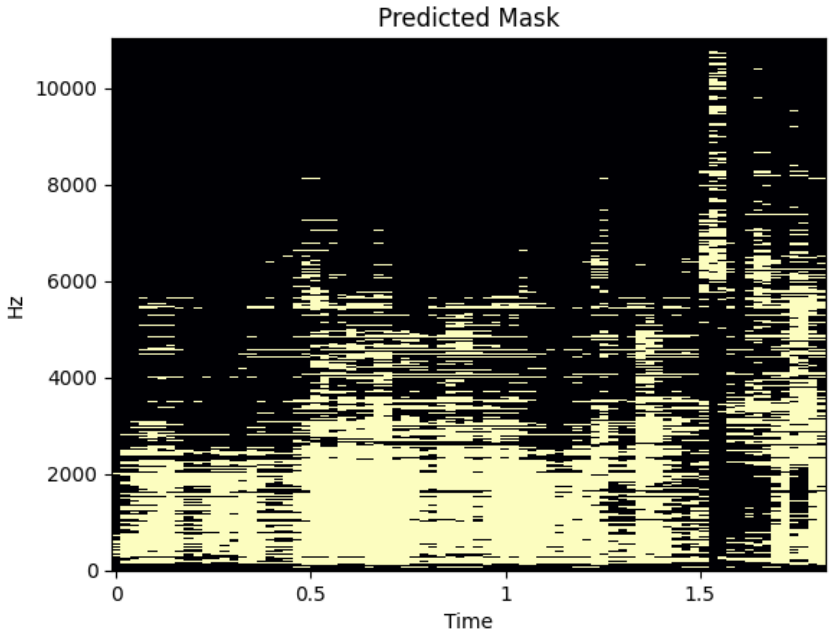
A Figura 4.9 também exemplifica notas de inteligibilidade em termos de STOI que

Figura 4.6: Máscara binária gerada pelo SVM.



o SVM, utilizando Kernel, foi capaz de atingir nos dados de teste. Como podemos observar, os resultados foram melhores do que a versão do SVM linear, onde a mediana do *boxplot* em termos de STOI ficou acima de 0.70, sendo que o valor mínimo do primeiro quartil está acima de 0.65. Porém, como mostra a Figura 4.7, o algoritmo preservou bastante componentes frequenciais ao longo do áudio, isso se reflete na inteligibilidade que, apesar de estar melhor, o barulho de ruído é mais perceptível se comparado ao SVM do caso linear.

Figura 4.7: Máscara binária gerada pelo SVM usando Kernel.



4.4 Análise de desempenho Redes Neurais Recorrentes

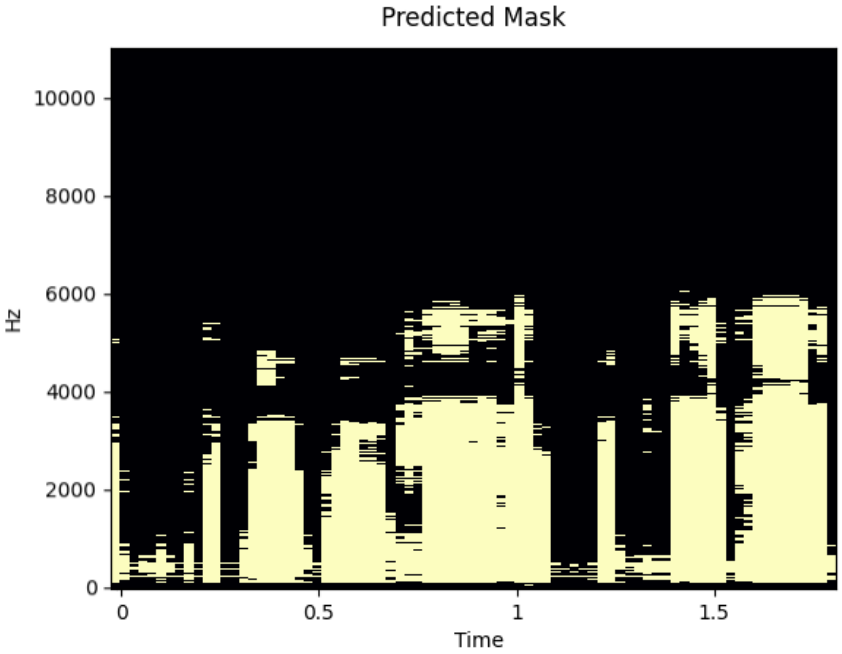
A Rede Neural Recorrente é uma versão de arquitetura de *deep learning* que tem o foco de processar dados sequenciais. Por isso, a rede consegue fazer o trabalho de predizer máscaras binárias dado a natureza do espectrograma, que possui uma componente temporal em sua estrutura. A arquitetura implementada para realizar o mapeamento entre os *bins* tempo-frequencial dos espectrogramas de entrada e *bins* das máscaras binárias ideais possui a primeira camada com 50 neurônios com função de ativação é a tangente hiperbólica e entrada de forma $(1, 1038)$, depois tem-se uma camada de *dropout* para evitar o *overfitting*. A segunda camada também é composta por 50 neurônios de ativação usando a tangent hiperbólica, e por fim, uma camada totalmente conectada com a função de ativação sendo a sigmóide com 1025 neurônios, que corresponde ao tamanho do vetor da máscara binária.

O número de amostras utilizado para o treinamento da rede foi o mesmo que para os demais algoritmos, ou seja, 1602 amostras de áudio para o treino do modelo e 300 áudios para validação da performance da rede. A Figura 4.9 ilustra a performance da rede, em termos de STOI. Evidentemente, a RNN apresenta um resultado melhor se comparado aos algoritmos mais clássicos de aprendizado de máquina, exceto pela versão *Kernel* do SVM. Sendo que, a mediana do *boxplot* fica em torno de 0.66, muito condizente com os resultados obtidos pela rede *Multilayer Perceptron*, como é apresentado na seção 4.1.

Ademais, se compararmos com as notas de *STOI* das máscaras binárias ideais, como mostra a Figura 4.2, observamos que a rede apresentou boas notas de STOI, já que temos como valor máximo um STOI próximo a 0.90, enquanto que as máscaras ideais apresentaram valores próximos a 0.95. Já o terceiro quartil da RNN ultrapassa 0.7, um pouco pior que as máscaras binárias, onde esse valor é aproximadamente 0.86.

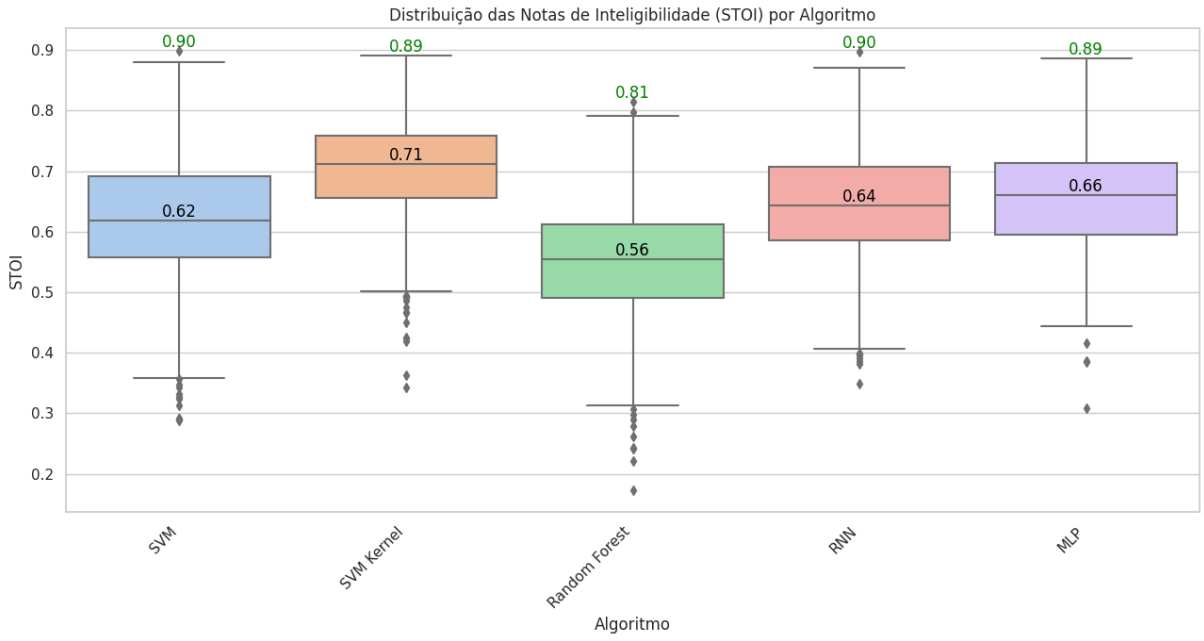
O bom resultado nas notas de inteligibildade da Rede Neural Recorrente se reflete na máscara binária que ela é capaz de gerar. A Figura 4.8 é um exemplo de uma máscara binária gerada a partir de um áudio combinando ruído e voz, sendo esse um som totalmente novo para a rede. É possível observar que a rede é capaz de manter estruturas importantes do espectrograma e diminuir o ruído do áudio.

Figura 4.8: Máscara binária gerada pela RNN



Em resumo, a Figura 4.9 apresenta os resultados obtidos em termos de notas de inteligibilidade (STOI) para cada um dos algoritmos discutidos nesta seção.

Figura 4.9: Notas de STOI para cada um dos algoritmos de classificação



5. *Conclusão*

No presente trabalho, foi realizado um estudo sobre o problema de extração de sinais utilizando a abordagem baseada em mascaramento tempo-frequencial, criando um sistema automático que consiga realizar a tarefa de ressaltar apenas o sinal de interesse.

Para determinar a máscara a ser utilizada no sistema, foram empregadas redes neurais artificiais, que foram ajustadas a fim de estimar a máscara a partir da informação contida no próprio sinal. O treinamento da rede considerou como entrada amostras de áudio fala-ruído e como saída suas respectivas máscaras binárias ideais, que foram obtidas a partir dos sinais de fala sem ruído, já conhecidos. Além de possuir duas camadas intermediárias, tendo a tangente hiperbólica como função de ativação e o algoritmo de otimização Adam. Para realizar o treinamento, 600 épocas foram suficientes, pois o algoritmo não apresentou melhorias significativas com um número superior de épocas.

Também foi possível realizar uma comparação entre a performance obtida pelas redes neurais e o uso de algoritmos mais clássicos, como as florestas aleatórias. Como foi evidenciado pela seção de resultados e discussão, as redes neurais apresentaram uma performance superior, devido à sua flexibilidade em encontrar padrões nos espectrogramas.

Os resultados do presente trabalho foram muito consistentes com a literatura, reforçando a flexibilidade das redes neurais para aprender e predizer máscaras binárias a partir de uma mistura fala-ruído, conseguindo extrair a fala-alvo da mistura e obtendo bons resultados de inteligibilidade. Ainda, existem grandes possibilidades de implementação para melhorar o desempenho do algoritmo, como o aproveitamento de mais dados, já que utilizamos uma pequena quantidade dos dados diponíveis, devido à falta de poder computacional. Além de que outras estruturas de mascaramento podem ser promissoras para atingir o mesmo objetivo, como o uso da *ratio mask* [1], que se mostrou ser mais eficiente do que a máscara binária, porém exige um poder computacional maior, e por fim o uso do cócleagrama [11] para treinar a rede neural, que também exige um grande poder computacional.

Referências Bibliográficas

[1] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and language processing*, vol. 26, pp. 1702–1726, 2018.

[2] A. Hyvarinen and J. Karhunen, *Independent Component Analysis*. Wiley-Interscience, 2001.

[3] C. J. Pierre Comon, *Handbook of Blind Source Separation: Independent Component Analysis and Applications*. Academic Press, 2010.

[4] Y. Wang and D. Wang, "Towards scaling up classification-based speech separation," *IEEE/ACM Transactions on Audio, Speech, and language processing*, vol. 21, pp. 1381–1390, 2013.

[5] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. PEARSON, 2009.

[6] J. Allen and L. Rabiner, "A unified approach to short-time fourier analysis and synthesis," *PROCEEDINGS OF THE IEEE*, vol. 65, pp. 1558–1564, 1977.

[7] A. Oppenheim, "Speech spectrograms using the fast fourier transform," *IEEE Spectrum*, vol. 7, pp. 57–62, 1970.

[8] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING*, vol. 32, 1984.

[9] Z. Khodzhaev, "Spectrogram - practical guide," *PROCEEDINGS OF THE IEEE*, 2020.

[10] D. Wang and G. Brown, "Computational auditory scene analysis: Principles, algorithms and applications.," *Wiley-IEEE Press*, 2006.

[11] R. Sharan and T. Moir, "Cochleagram image feature for improved robustness in sound recognition," *IEEE International Conference on Digital Signal Processing*, 2015.

[12] C. Taal and R. Hendriks, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and language processing*, vol. 19, 2011.

[13] B. Logan, "Mel frequency cepstral coefficients for music modeling," *Proc. 1st Int. Symposium Music Information Retrieval*, 11 2000.

[14] G. Hu and D. L. Wang, "Speech segregation based on pitch tracking and amplitude modulation," *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, p. 79–82, 2001.

[15] T. S. C. Hummersone and T. Brooks, "On the ideal ratio mask as the goal of computational auditory scene analysis," *Blind Source Separation, G. R. Naik and W. Wang, Eds. Berlin, Germany: Springer*, p. 349–368, 2014.

[16] A. Rix and J. Beerends, "Perceptual evaluation of speech quality (pesq) – a new method for speech quality assessment of telephone networks and codecs," *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001.

[17] ITU-T Rec. P.10 (2006), *Vocabulary for performance and quality of service.*

[18] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING*, vol. 14, p. 1462–1469, 2006.

[19] T. M. Mitchell, *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[20] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in Python*. Springer, 2003.

[21] J. Nayak, B. Naik, and H. S. Behera, "A comprehensive survey on support vector machine in data mining tasks: Applications challenges," *International Journal of Database Theory and Application*, vol. 14, p. 1462–1469, 2006.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] C. E. Nwankpa and et al, "Activation functions: Comparison of trends in practice and research for deep learning," 2020.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," *Department of Computer Science, University of Toronto, Toronto, ON M5S 2G4, Canada*.

[25] M. Zeiler and et al, "On rectified linear units for speech processing," *New York University, USA. Google Inc., USA. University of Toronto, Canada*.

[26] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, 1990.

[27] D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS, "Learning internal representations by error propagation,"

[28] S. Ruder, "An overview of gradient descent optimization algorithms," *Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin*.

[29] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization,"

[30] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv e-prints*, 11 2015.

[31] G. Hu and D. Wang, "A tandem algorithm for pitch estimation and voiced speech segregation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, pp. 2067–2079, 2010.

[32] J. Garofolo and et al, "Darpa timit acoustic-phonetic continuous speech corpus," *National Inst. of Standards and Technol.*, 1993.

[33] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python.," *In Proceedings of the 14th python in science conference*, pp. 18–25, 2015.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.