UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GERMANO CABERLON SMANIOTTO

SUPER-RESOLUÇÃO DE ÁUDIO UTILIZANDO REDES NEURAIS ARTIFICIAIS

Porto Alegre
2023

GERMANO CABERLON SMANIOTTO

SUPER-RESOLUÇÃO DE ÁUDIO UTILIZANDO REDES NEURAIS ARTIFICIAIS

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Engenheiro Eletricista.

ORIENTADOR: Prof. Dr. Tiago Oliveira Weber

Porto Alegre
2023

GERMANO CABERLON SMANIOTTO

SUPER-RESOLUÇÃO DE ÁUDIO UTILIZANDO REDES NEURAIS ARTIFICIAIS

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Engenheiro Eletricista.

Orientador: _____
Prof. Dr. Tiago Oliveira Weber, UFRGS
Doutor pela Universidade de São Paulo – São Paulo, Brasil

Banca Examinadora:

Prof. Dr. Tiago Oliveira Weber, USP
Doutor pela Universidade de São Paulo

Prof. Dr. Ivan Müler, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul

Prof. Dr. Raphael Martins Brum, UM2
Doutor pela Université Montpellier 2

Chefe do DELET: _____
Roberto Petry Homrich

Porto Alegre, setembro de 2023.

AGRADECIMENTOS

Agradeço, em primeiro lugar, ao meu orientador, Tiago, por todo o apoio e paciência.

Agradeço a todos que fizeram parte da minha vida, que me ensinaram a ser.

Agradeço meu grande amigo, Maurício, que me ensinou a pertencer.

Agradeço minha namorada, Olivia, que me ensinou a amar.

Agradeço minha irmã, Melissa, que me ensinou a me divertir.

Agradeço meus pais, Adair e Luciane, que me ensinaram a andar, correr, falar, ler, escrever, sorrir, chorar. Agradeço a vocês dois, que me ensinaram a perder, a vencer, a viver. Sou extremamente sortudo de ser seu filho. Amo vocês.

RESUMO

A super-resolução de áudio é um problema da teoria de processamento de sinais cujo objetivo é aumentar o número de pontos no conjunto de amostras de um sinal de forma inteligente, resultando em uma aproximação de uma frequência de amostragem superior e uma melhor qualidade percebida. Recentemente, técnicas de redes neurais artificiais têm recebido muita atenção para a resolução de inúmeras diferentes categorias de problemas. O presente trabalho busca, portanto, estudar as metodologias de super-resolução de sinais de voz com foco no uso de redes neurais artificiais, que demonstram um grande potencial nesta área. Para isso, são investigadas duas diferentes arquiteturas de redes neurais (um *autoencoder* e uma rede convolucional baseada em *autoencoders*) para 3 diferentes taxas de ampliação de frequência de amostragem (2, 4 e 6), bem como diferentes técnicas de treinamento para avaliar o seu desempenho no problema proposto. Além disso, são realizadas comparações com abordagens tradicionais de processamento de sinais, utilizando as métricas de desempenho SNR (relação sinal-ruído), LSD (distância logarítmica do espectro) e o tempo de inferência. Por fim, é avaliada a possibilidade de se empregar uma rede neural de super-resolução para a conversão de sinais de áudio em tempo real. Foi concluído que, para todas as taxas $r = 2$ e $r = 4$ de ampliação de frequência de amostragem, a rede convolucional possui o melhor desempenho: SNR de 27,2 e 22,6, enquanto que para $r = 6$ a rede mais performática foi um *autoencoder* de pequeno porte e rápido tempo de inferência. A *baseline* considerada foi a interpolação *spline*, cuja maior vantagem é o tempo de inferência extremamente rápido, apesar de não resultar em boas métricas de desempenho quando comparada às redes neurais.

Palavras-chave: Super-resolução de áudio; redes neurais artificiais; aprendizado de máquina; processamento de sinais; ampliação de largura de banda.

ABSTRACT

Audio super-resolution is a problem of signal processing theory whose goal is to intelligently increase the number of points of a signal sample set, resulting in an approximation of a higher sampling frequency and improved perceived quality. Recently, artificial neural network techniques have been used to solve different kinds of problems. The present work aims to study existing techniques of voice signal super-resolution focusing on artificial neural networks, which demonstrate great potential in this area. To this end, two different neural network architectures (an autoencoder and a convolutional network based on autoencoders) are investigated for 3 different upsampling rates (2,4 and 6), as well as different training techniques to evaluate their performance on said problem. In addition, comparisons with traditional signal processing techniques are made, using SNR (signal-to-noise ratio), LSD (log-spectral distance), and inference time as performance metrics. Finally, the possibility of using a super-resolution neural network for real-time audio signal conversion is assessed. It was concluded that, for all the amplification rates of $r = 2$ and $r = 4$, the convolutional network has the best performance: SNR of 27.2 and 22.6, while for $r = 6$ the most performant network was a small and fast-inference autoencoder. The considered baseline was the spline interpolation, whose greatest advantage is the extremely fast inference time, despite not resulting in good performance metrics when compared to neural networks.

Keywords: audio super-resolution; artificial neural networks; machine learning; signal processing; bandwidth expansion.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ADC	*Analog-Digital Converter*
AE	*Autoencoder*
API	*Application Programming Interface*
$BIBO$	*Bounded Input, Bounded Output*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
DAC	*Digital-Analog Converter*
CD	*Compact Disc*
$CSTR$	*Centre for Speech Technology*
DFT	*Discrete Fourier Transform*
$DTFT$	*Discrete Time Fourier Transform*
DVD	*Digital Video Disc*
ELU	*Exponential Linear Unit*
FFT	*Fast Fourier Transform*
FIR	*Finite Impulse Response*
FT	*Fourier Transform*
GAN	*Generative Adversarial Network*
GB	*GigaByte*
$GELU$	*Gaussian Error Linear Unit*
CPU	*Central Processing Unit*
$HDF5$	*Hierarchical Data Format version 5*
IIR	*Infinite Impulse Response*
IPA	*International Phonetic Alphabet*
$ISTFT$	*Inverse Short Time Fourier Transform*
KNN	*K-Nearest Neighbors*
LSD	*Log-Spectral Distance*

LTI	*Linear Time Invariant*
MAE	*Mean Absolute Error*
$MFCC$	*Mel-Frequency Cepstrum Coefficients*
MLP	*Multilayer Perceptron*
MSE	*Mean Squared Error*
PCA	*Principal Component Analysis*
$ReLU$	*Rectifier Linear Unit*
RMS	*Root Mean Square*
SGD	*Stochastic Gradient Descent*
SNR	*Signal to Noise Ratio*
$STFT$	*Short Time Fourier Transform*
SVM	*Support Vector Machine*
$TFLOPS$	*Teraflops*
$TFNet$	*Time-Frequency Network*
$VCTK$	*Voice Cloning Toolkit*

LISTA DE SÍMBOLOS

c	*Velocidade da onda sonora*
u	*Amplitude de Onda*
t	*Velocidade da onda sonora*
∂	*Operador diferencial parcial*
∇	*Operador diferencial vetorial*
k	*Número de onda*
λ	*Comprimento de onda*
π	*Razão entre circunferência e raio do círculo*
f	*Frequência*
t	*Tempo*
ϕ	*Fase*
dB	*Decibel*
Hz	*Hertz*
W	*Watts*
m	*Metro*
m^2	*Metro Quadrado*
I	*Intensidade sonora*
f_s	*Taxa de amostragem*
I	*Intensidade sonora*
$\mathbb{Z}$	*Conjunto dos números inteiros*
$\mathbb{C}$	*Conjunto dos números complexos*
$\mathbb{N}$	*Conjunto dos números naturais*
$\mathbb{R}$	*Conjunto dos números reais*
$\in$	*Pertence*
T	*Período*
n	*Tempo discretizado*

f_c	*Frequência de Nyquist*
f_m	*Frequência máxima de um sinal*
$x(t)$	*Sinal no tempo contínuo*
$X(\omega)$	*Sinal na frequência*
$x[n]$	*Sinal no tempo discreto*
$X(e^{jw})$	*Sinal na frequência da DTFT*
e	*Número de Euler*
d	*Operador diferencial*
N	*Número de pontos*
∞	*Número infinito*
$\int$	*Operador integrador*
$\sum$	*Operador somatório*
τ	*Variável alternativa para o tempo*
m	*Variável alternativa para o tempo discreto*
σ	*Desvio-padrão*
μ	*Média*
T	*Transformação*
S	*Sinal*
$\lvert \cdot \rvert$	*Operador de normalização*
$\delta[n]$	*Impulso unitário*
$h[n]$	*Resposta ao impulso*
$H(\omega)$	*Resposta ao impulso na frequência*
G	*Ganho*
ϵ	*Erro*
θ	*Parâmetros*
ω	*Pesos*
Ω	*Regularizador*
θ_{MLE}	*Estimador de Máxima Verossimilhança*
Θ	*Conjunto dos parâmetros*
$\ln$	*Logaritmo natural*
$\log$	*Logaritmo base 10*
b	*Viés*
$Var(x)$	*Matriz de covariância*
Λ	*Matriz de autovalores*

g	*Gradiente estimado*
L	*Função de perda unitária*
E	*Operador de média aritmética*
δ^l	*Erro propagado por uma camada*
$\max$	*Operador de máximo*
$\sigma(x)$	*Função sigmoide*
$\tanh(x)$	*Função tangente hiperbólico*
$I(x)$	*Função identidade*
Φ	*Função de distribuição cumulativa*
L^2	*Espaço das funções de norma quadrática finita*
L^1	*Espaço das funções absolutamente integráveis*
L	*Fator de ampliação de largura de banda*
r	*Fator de ampliação de largura de banda*
f_{LR}	*Frequência de amostragem de baixa resolução*
f_{HR}	*Frequência de amostragem de alta resolução*
$\sqrt{\cdot}$	*Operador de raiz quadrada*
$X(\ell, k)$	*Potência logarítmica espectral*
t_{med}	*Tempo de inferência mediano*

SUMÁRIO

1 INTRODUÇÃO

Na última década, os avanços no campo de inteligência artificial e, mais notavelmente, em aprendizagem de máquina, vêm demonstrando que atividades difíceis com poucas ou nenhuma solução analítica podem ser atacadas com um novo conjunto de ferramentas. Anteriormente, pensava-se impossível desenvolver um conjunto de algoritmos capazes de simular capacidades artísticas e intuitivas. Hoje, tarefas como geração de pinturas similares às de grandes artistas e superar os humanos no jogo chinês Go (considerado como um dos jogos de tabuleiro que mais exigem intuição e com um número gigantesco de jogadas possíveis) já foram alcançadas. Esses e outros exemplos indicam que cada vez mais tarefas que hoje exigem intervenção humana serão resolvidas com a utilização de inteligência artificial.

Uma dessas tarefas é a super-resolução de imagens - ou, ainda, super-zoom. Seu objetivo é: dado uma imagem-alvo, busca-se ampliar a sua qualidade percebida por meio da geração de pixels intermediários, distorcendo-a o mínimo possível. Esse problema tem aplicações em inúmeros campos como fotografia, compressão de arquivos, telescopia e microscopia.

A super-resolução de sinais arbitrários tem se mostrado cada vez mais presente nos últimos anos em diversos estudos. No passado, essa tarefa era concluída através de métodos de filtragem analógicos e digitais que, apesar da eficácia, não são tão capazes de restaurar as informações de altas frequências. Disso surge o interesse de se utilizar técnicas mais sofisticadas. O presente trabalho, portanto, abordará um problema de análise de sinais: o de super-resolução de arquivos de áudio (também denominado extensão de largura de banda) utilizando redes neurais artificiais.

O problema da super-resolução de áudio para fins de regeneração de sinais de voz foi amplamente estudado durante a década de noventa e início dos anos 2000. O objetivo das

pesquisas nessa época era reconstruir as altas frequências perdidas durante a transmissão de áudio, principalmente em vias telefônicas. (ABE; YOSHIDA, 1995; YASUKAWA, 1996)

Esses estudos utilizaram, em sua maioria, técnicas preditivas lineares (FUEMMELER; HARDIE; GARDNER, 2001) que dividiam o problema de reconstrução em dois: formar um sinal de erro residual de banda larga e recriar um conjunto de coeficientes lineares preditivos de banda larga capazes de filtrar tais sinais de erro, resultando num sinal de voz regenerado de banda larga.

Técnicas mais recentes utilizam métodos mais avançados de pós-processamento de sinais filtrados por algoritmos de redução de ruído tradicionais (DING; SOON; YEO, 2010). Apesar de não atuarem diretamente na ampliação de largura de banda, essas técnicas desempenham um papel semelhante ao de melhorar a qualidade percebida de sinais de voz.

A utilização de redes neurais artificiais tornou-se alvo de grande interesse nos últimos dez anos, incluindo para a super-resolução de imagens. Diversos tipos de redes neurais foram utilizados para resolver este problema, em especial as redes neurais convolucionais (DONG *et al.*, 2015) e as redes generativas antagônicas (ou GANs) (LEDIG *et al.*, 2016). Graças ao crescente número de bancos de dados abertos, essas e outras pesquisas foram capazes de solucionar o problema com poucos artefatos visuais, e alguns estudos foram capazes inclusive de aplicar métodos semelhantes para super-resolução de vídeos em tempo real. (SHI *et al.*, 2016)

Nos últimos cinco anos, algumas pesquisas promissoras têm surgido no campo de super-resolução de áudio. A principal arquitetura de rede neural utilizada é a convolucional, sendo utilizada para reconstrução de altas frequências (MIRON; DAVIES, 2018) com uma implementação relativamente simples (KULESHOV; ENAM; ERMON, 2017). Outras frentes desenvolveram as chamadas TFNet (ou redes de tempo-frequência) (LIM *et al.*, 2018; DONG *et al.*, 2020), que propõem redes neurais atuando tanto no domínio do tempo quanto no domínio da frequência simultaneamente a fim de evitar problemas gerados pela Transformada de Fourier de Curto-Tempo (STFT).

1.1 Justificativa

A resolução do problema de super-resolução de áudio é fundamental em diversas esferas da tecnologia e da comunicação. Em setores como telecomunicações, a capacidade de aprimorar a qualidade do áudio contribui para comunicações mais claras e nítidas, sendo especialmente valiosa em videoconferências e chamadas telefônicas. Na compressão de dados, a super-resolução desempenha um papel crucial ao preservar a fidelidade do áudio mesmo em configurações de taxa de bits reduzida, resultando em economia de espaço de armazenamento e melhor eficiência na transmissão de dados.

Além disso, a super-resolução de áudio tem implicações significativas na análise forense, onde auxilia na identificação de detalhes cruciais em gravações de áudio, desempenhando um papel essencial na resolução de casos criminais. Por outro lado, em aplicações de síntese de áudio e conversão de texto para fala, a super-resolução melhora a qualidade e a naturalidade da voz gerada, enriquecendo a experiência do usuário em assistentes de voz e sistemas de resposta automática. Não menos importante, essa abordagem possibilita economias substanciais em hardware, tornando acessível a captura de áudio de alta qualidade em dispositivos de baixo custo, democratizando, assim, a excelência sonora em uma ampla gama de cenários.

1.2 Objetivos

1.2.1 Objetivos Gerais

O objetivo primário do presente projeto é estudar e buscar aprimorar métodos existentes de ampliação de largura de banda de sinais de voz utilizando redes neurais artificiais. As métricas que serão utilizadas para analisar os resultados obtidos são uma comparação direta das medidas SNR (Relação sinal-ruído) e LSD (Distância logarítmica do espectro) entre um sinal de áudio e sua reconstrução gerada pela rede neural. Esses resultados serão comparados com os mesmos para uma super-resolução interpolativa (por *splines*).

1.2.2 Objetivos Específicos

Cabe, ainda, separar alguns objetivos específicos adequados que demonstrem o progresso e andamento do projeto. São eles:

- Estudar e analisar as técnicas existentes de super-resolução;

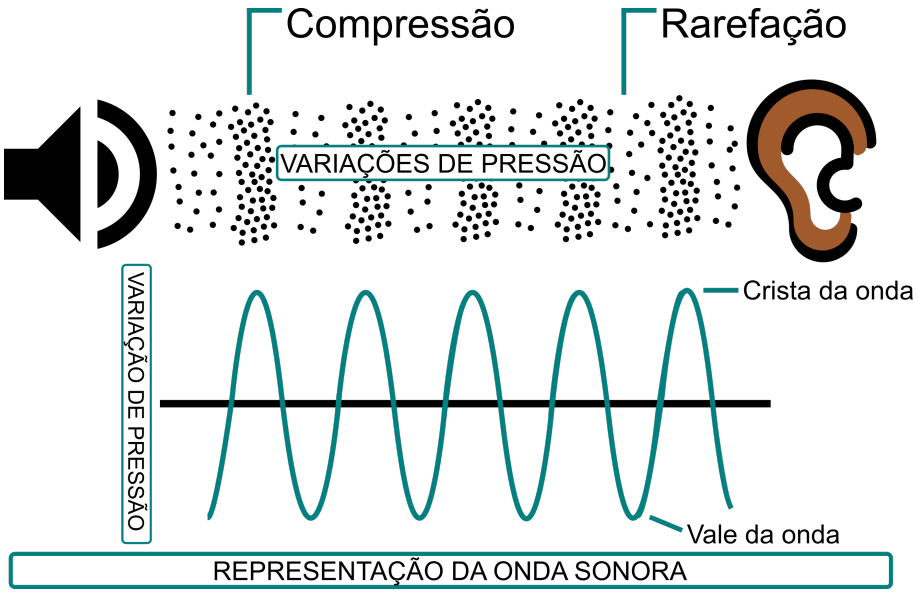
- Escolher duas técnicas de redes neurais para a resolução do problema;
- Escolher, analisar e processar uma base de dados de sinais de voz;
- Implementar um algoritmo de treinamento das redes neurais;
- Comparar o desempenho das técnicas implementadas com o método de interpolação *spline*;

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Som e formas de onda

O som é definido como uma oscilação ou superposição de oscilações na pressão de um meio com forças internas (elásticas ou viscosas), e suas grandezas fundamentais são a pressão e o tempo. Essas oscilações, ou vibrações, se propagam através de ondas longitudinais em meios como ar, água e sólidos, e podem se propagar também como ondas transversais em sólidos. Elas são geradas conforme a fonte sonora vibra, se propagando no meio na velocidade do som c a ele correspondente (no ar, à pressão ambiente, a velocidade do som é de 343 m/s) e transmitindo energia. A figura 1 apresenta uma visualização do som em formato de onda.

Figura 1 – Visualização de uma onda sonora.



Fonte: WIKIPEDIA (2023a).

As ondas sonoras obedecem a equação de onda

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \tag{1}$$

e carregam diversas informações. A solução mais simples para a versão unidimensional dessa equação é $u(x,t) = \sin(kx - \omega t)$, onde ω é a frequência angular expressa em rad/s e k é o número de onda expresso em m^{-1}, obedecendo a relação $c = \omega/k$. O comprimento de onda em m é obtido de k através da fórmula $\lambda = 2\pi/k$, e a frequência em Hertz é obtida através da fórmula $f = 2\pi\omega$. Se é adicionado um termo ϕ (fase) ao argumento da senoide, a equação de onda continua sendo obedecida.

Por conta do teorema da superposição, a equação de onda também é satisfeita por uma superposição de diferentes ondas sinusoidais. Isso permite que as ondas sonoras possuam componentes de diferentes amplitudes e de diferentes frequências, podendo ser arbitrariamente complexas. De forma a caracterizar e comparar ondas sonoras, utiliza-se características como o tom e o timbre.

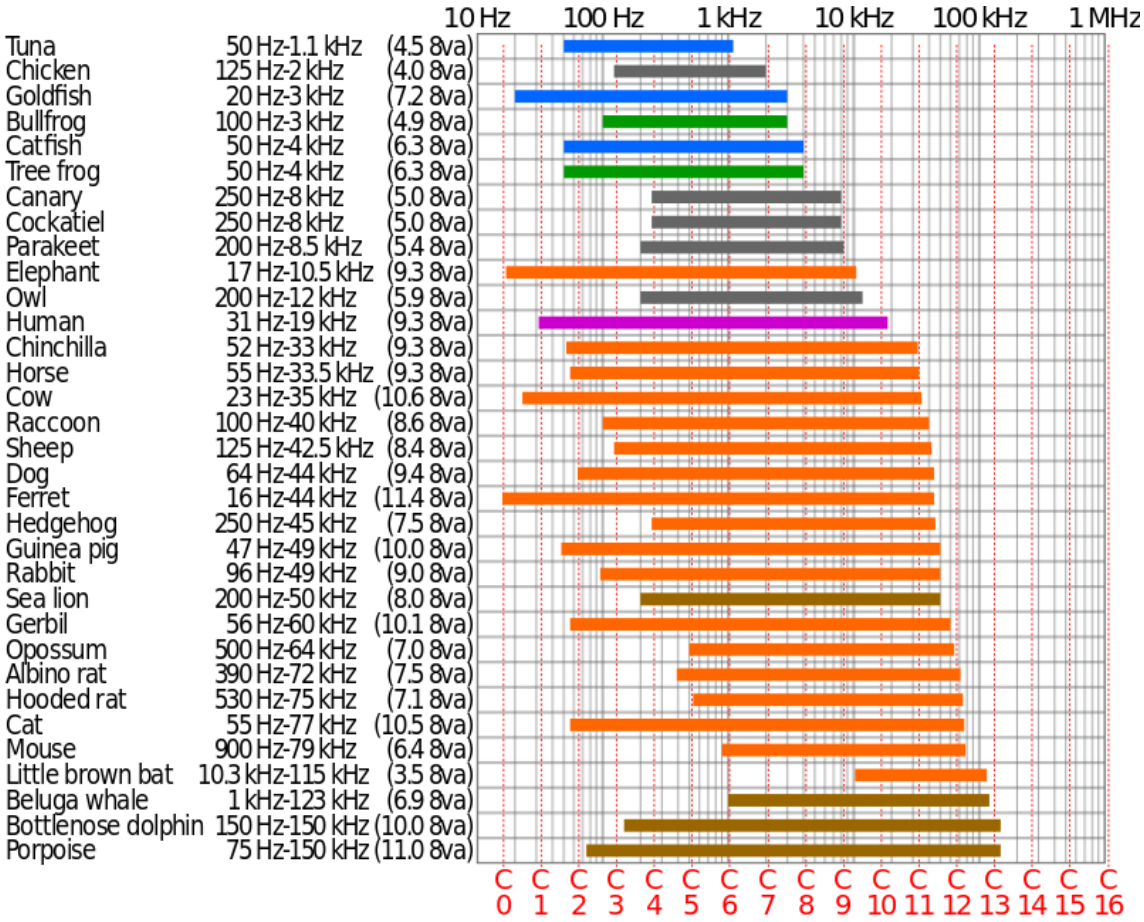
O tom é a percepção logarítmica da frequência. O ouvido humano é capaz de captar ondas sonoras de frequências entre 30 Hz e 19 kHz, e interpreta como similares duas ondas com frequências diferentes se elas diferem de uma potência de 2. Por exemplo, a frequência da nota Dó, base do sistema tonal, vale $130,813 \cdot 2^m$ Hz para valores inteiros de m.

Já o timbre é percebido como a qualidade de sonora e representa a alocação pré-consciente de uma identidade a um som. É o timbre que permite diferenciar um instrumento de outro, mesmo que sejam tocados na mesma frequência fundamental e com mesma intensidade. Isso se deve ao fato de que, apesar de possuírem a mesma frequência fundamental, a amplitude das frequências harmônicas (múltiplos inteiros da frequência fundamental) difere, resultando em formatos de onda diferentes. Além disso, compõe o timbre também o envelope sonoro, que é composto por quatro diferentes momentos:

- Ataque: início de cada nota musical;
- Decaimento: período de diminuição de intensidade sonora após o ataque, anterior à estabilização da onda;
- Sustentação: tempo de duração da nota musical, tipicamente é controlável;

- Relaxamento: final da nota, quando a intensidade sonora diminui até desaparecer completamente.

Figura 2 – Intervalo de frequência audível para diferentes animais.



Fonte: WIKIPEDIA (2023b).

Como as ondas sonoras transmitem energia, a cada uma delas é atribuída também a potência sonora, que é a taxa pela qual a energia é transferida por tempo em todas as direções, medida em Watts. Geralmente, fontes sonoras emitem uma baixa potência. Um trovão possui uma potência sonora similar à uma orquestra sinfônica, cuja ordem de grandeza é 1W.

Já a intensidade sonora se refere à densidade superficial da potência sonora, que é medida em W/m². É a intensidade sonora que é captada pelo ouvido humano, permitindo a distinção entre sons altos e sons baixos. Os seres humanos são capazes de ouvir intensidades sonoras entre 10^{-12} W/m² e 10 W/m² (ponto no qual o humano passa a sentir dor). Como a frequência, a intensidade sonora também é percebida de forma logarítmica, sendo medida também em decibéis (dB) pela fórmula de conversão $I_{dB} = 10\log(I_{W/m}/10^{-12})$.

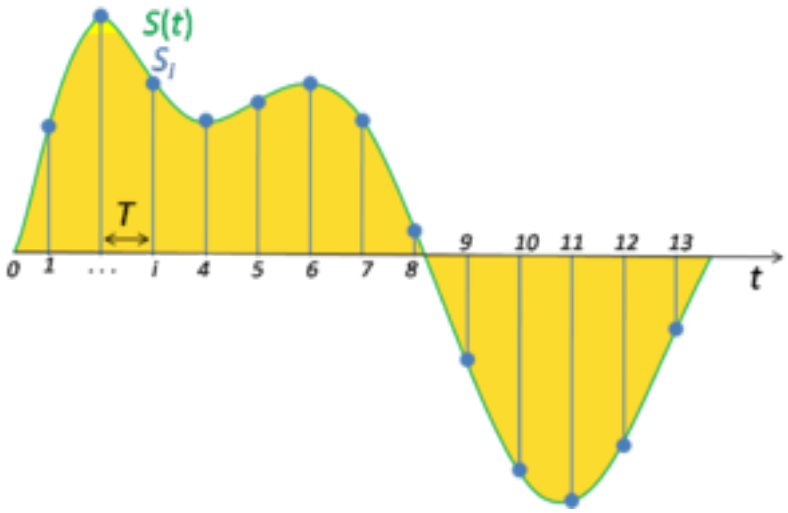
Dessa forma, o intervalo da audição humana é entre 0 dB e 130 dB. A partir de 160 dB, o tímpano é instantaneamente perfurado.

2.1.1 Sinais de áudio

Um sinal de áudio é qualquer representação de um som que contenha todas as informações necessárias para reproduzi-lo. Sinais de áudio podem ser contínuos (tensão elétrica em circuitos analógicos) ou discretos (*bits* em circuitos digitais). Quando digitalizado, por meio de um ADC (conversor analógico-digital), o sinal de áudio contínuo é amostrado e quantizado.

A amostragem, conforme a Figura 3, é o procedimento que discretiza o eixo temporal de um sinal com uma taxa de amostragem f_s, a qual indica o número de amostras do sinal por unidade de tempo. Para compreender todo o espectro da audição humana, a taxa de amostragem mínima necessária é de aproximadamente 40 kHz. É usual, em aplicações comerciais, a utilização de áudios em 48 kHz em virtude do seu alto número de divisores.

Figura 3 – Visualização da amostragem de um sinal contínuo.

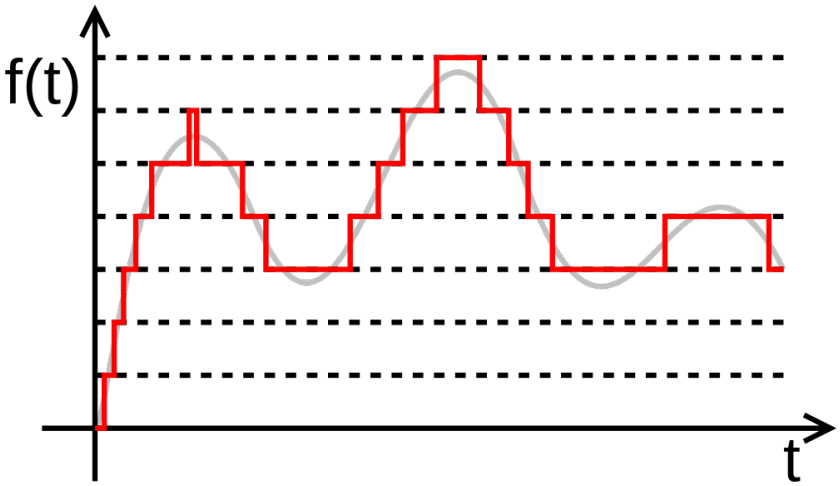


Fonte: WIKIPEDIA (2022).

Já a quantização discretiza o eixo da tensão elétrica correspondente à pressão do meio em que o áudio foi captado. Essa quantização é feita armazenando-se o valor da amplitude para uma dada amostra em um número fixo de bits. Por conta disso, a resolução da amplitude é chamada de profundidade de bits do áudio digitalizado, e quanto maior a profundidade de bits, mais preciso é a representação do sinal digital. Por exemplo, CDs utilizam uma profundidade de bits de 16, o que permite que a amplitude possa assumir

65536 valores diferentes. Em outras aplicações, como DVDs e computadores, é comum a utilização de uma profundidade de bits de 24 ou mais, cuja amplitude pode assumir 16,78 milhões de valores diferentes.

Figura 4 – Visualização da quantização de um sinal contínuo.



Fonte: WIKIPEDIA (2006).

Para a gravação de áudio (conversão da onda mecânica para sinal digital), é explorado o caráter oscilatório do som. Quando uma onda sonora atinge um microfone, o diafragma nele contido oscila e gera um sinal elétrico analógico, que é posteriormente filtrado, amostrado e convertido em um sinal digital por um ADC (quantização). Para a reprodução de áudio, esse processo é revertido: um DAC (conversor digital-analógico) converte o sinal digital em um sinal analógico, que é filtrado, amplificado e transmitido a uma membrana dos auto-falantes que oscila conforme o sinal analógico, produzindo uma onda mecânica.

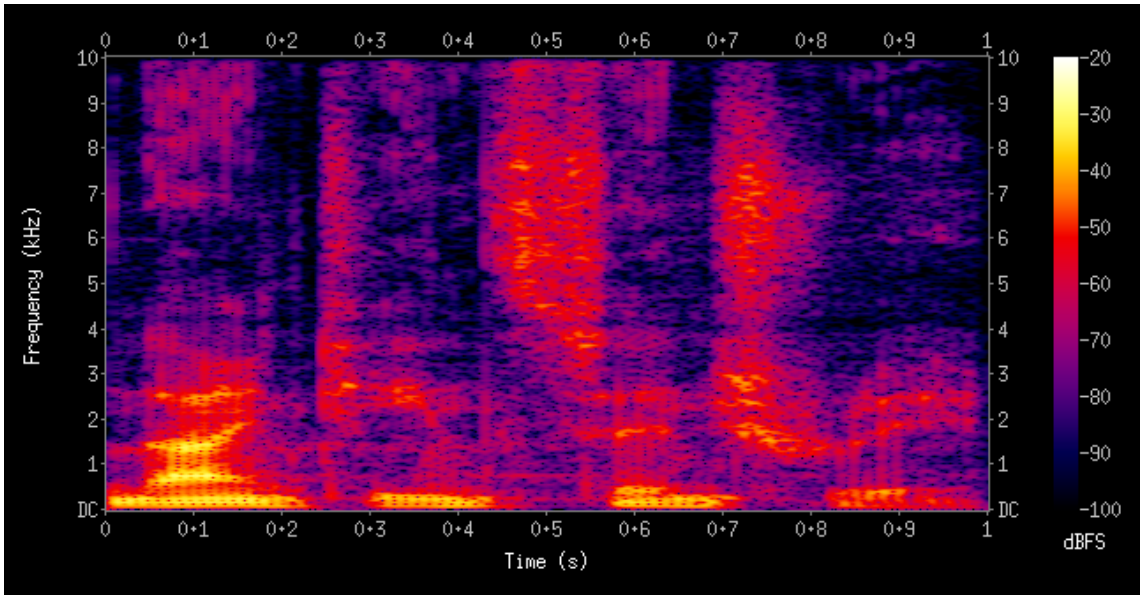
2.1.2 Outras propriedades e características de áudios

Além das propriedades fundamentais do som descritas anteriormente, sinais de áudio possuem diversas características diferentes. Essas características podem ser quantitativas ou qualitativas, e permitem uma descrição completa do áudio para diferentes níveis de abstração.

A mais importante característica de um sinal sonoro é a distribuição de suas frequências, que pode variar ao longo do tempo. Para representar essa distribuição ao longo do tempo, são utilizados os chamados espectrogramas, conforme a Figura 5.

Em virtude da representação em espectrograma, sinais de áudio podem ser caracteri-

Figura 5 – Espectrograma da gravação de voz para as palavras "*nineteenth century*".



Fonte: WIKIPEDIA (2008).

zados tanto no domínio do tempo como no domínio da frequência.

Características do domínio do tempo revelam recortes instantâneos de um sinal, e tipicamente são relacionadas à amplitude do sinal. Como exemplo, o envelope da amplitude no tempo permite identificar os intervalos de tempo com maior ou menor intensidade sonora. É útil também na detecção de *onsets* (tempo em que a nota foi tocada) e classificação de gênero musical. Outras medidas importantes são a energia RMS (*Root mean square*), que atua como uma medida central móvel, e a taxa de cruzamento de zeros, muito utilizada em reconhecimento de voz.

Já as características no domínio da frequência revelam informações mais abstratas do sinal de áudio. Obtidas em janelas de tamanho fixo, essas características permitem analisar separadamente cada componente de frequência de um sinal. Acordes, melodias, ritmos e energia costumam ser identificados no domínio da frequência.

Algumas medidas importantes são a frequência principal (o tom de um sinal) e o centroide espectral, que é uma medida de tendência central análoga ao centro de massa para sólidos. Essas medidas permitem a identificação das tonalidades mais relevantes em um sinal de áudio.

Além do espectro usual da frequência, há também o cepstro de frequências de Mel, que busca representar sinais sonoros conforme o ouvido humano os interpreta. Em particular, esse espectro costuma utilizar um número pequeno de coeficientes (em geral, de 10

a 20) chamados MFCC, que atuam como descritores do timbre e da distribuição de energia na frequência de um sinal. Os MFCC são muito utilizados em tarefas de reconhecimento de fala.

Por fim, é possível classificar as características de áudio por nível de abstração:

- Alto nível de abstração: Instrumentos utilizados, acordes e melodias presentes, ritmos, palavras, gênero musical e energia musical.
- Médio nível de abstração: Tom, descritores relacionados à batida, *onsets*, padrões de flutuação de MFCC.
- Baixo nível de abstração: Envelope de amplitude, energia do sinal, centroide espectral, fluxo espectral e taxa de cruzamento de zeros.

2.2 Teoria de processamento de sinais

Em teoria de processamento de sinais, define-se como sinal tudo aquilo que carrega informação de alguma maneira. Os sinais costumam indicar o estado ou comportamento de um sistema, e geralmente são sintetizados com o objetivo de permitir a comunicação entre dois interlocutores, sejam eles humanos ou máquinas.

Matematicamente, os sinais são representados como funções de uma ou mais variáveis independentes. A eletricidade, por exemplo, pode ser modelada como um mapeamento do tempo no conjunto dos números complexos. Já as imagens digitais podem ser modeladas como funções que levam duas variáveis espaciais discretas ao conjunto dos pixels possíveis de serem exibidos nas telas.

Costuma-se estudar duas classes especiais de sinais: sinais contínuos no tempo, definidos como funções do conjunto dos números reais, e sinais discretos no tempo, definidos como funções no conjunto dos números inteiros (isto é, sequências). Como será visto posteriormente, os dois estão intimamente relacionados, de forma que para a grande maioria dos sinais há representações tanto contínuas como discretas – e, sendo cumpridas algumas condições, as duas representações são equivalentes.

Os sinais discretos são particularmente utilizados em uma variedade de tecnologias contemporâneas devido à flexibilidade e facilidade de manipulação que eles oferecem em comparação aos sinais contínuos. Em particular, destacam-se os sinais digitais, definidos

como sequências discretas cuja imagem também é discretizada, além de poderem assumir apenas um número limitado de valores. Estes sinais são responsáveis pela grande parte das aplicações computadorizadas, servindo como aproximações suficientes dos sinais contínuos que representam.

Todo sinal contínuo pode ser transformado em sinal discreto a partir de um procedimento chamado de amostragem, demonstrado anteriormente na Figura 3. Na amostragem, o sinal contínuo é dividido em múltiplos intervalos de mesmo comprimento (chamado de período e comumente denotado pela letra T) de forma que o sinal discreto é definido como o valor do sinal contínuo em cada fronteira destes intervalos.

2.2.1 Sinais discretos no tempo e teorema de amostragem de Nyquist-Shannon

Os sinais discretos no tempo são representados como sequências numéricas na forma

$$x = x[n], n \in \mathbb{Z}, \tag{2}$$

tal que $x : \mathbb{Z} \rightarrow \mathbb{C}$, embora normalmente a imagem pertença a $\mathbb{R}$. No caso de sinais digitais, cuja amplitude é quantizada, a imagem é um conjunto contável que pode assumir 2^b valores, onde b é a quantidade de bits disponíveis para codificar a amplitude do sinal. Essa quantidade é também chamada de profundidade de bit, e os valores mais utilizados são 16-bit, 24-bit e 32-bit.

Os sinais discretos podem também ser obtidos de um sinal contínuo $x_c(t)$ pela equação:

$$x[n] = x_c(nT), T \in \mathbb{R}, \tag{3}$$

que define a amostragem, sendo T o período de amostragem em segundos. Seu recíproco, $f_s = \frac{1}{T}$, é a frequência de amostragem, uma das propriedades mais relevantes de sinais discretos.

A partir da definição da frequência de amostragem, é natural questionar qual a mínima frequência de amostragem necessária para representar todas as informações de um sinal. Desse questionamento, surge um dos resultados mais importantes da teoria de processamento de sinais: o teorema de amostragem de Nyquist-Shannon (SHANNON, 1949). Esse teorema afirma que um sinal contínuo pode ser completamente determinado por um sinal discreto se a taxa de amostragem for pelo menos duas vezes maior que a maior frequência presente no sinal. Matematicamente, isso se expressa como

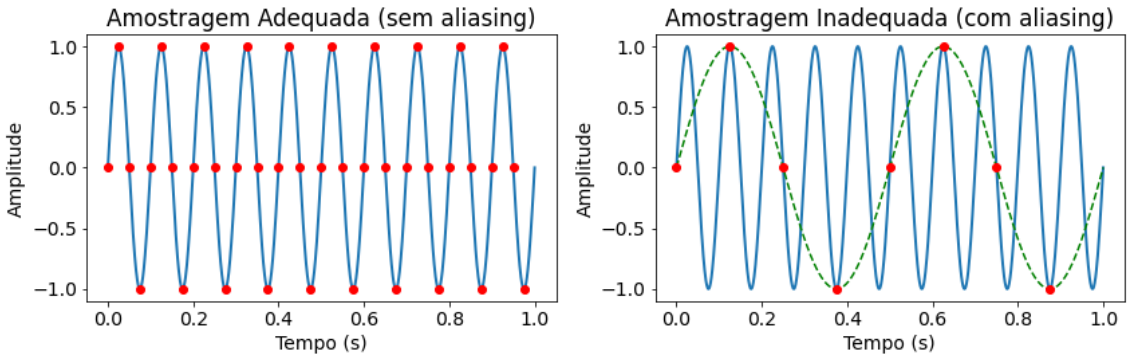
$$f_s \geq 2f_m, \tag{4}$$

onde f_m representa a maior frequência do sinal analógico.

Essa frequência crítica $f_c = 2f_m$ é conhecida como a frequência de Nyquist e representa a menor taxa de amostragem que pode ser empregada para capturar todas as informações do sinal contínuo. Se uma amostragem for realizada abaixo da frequência de Nyquist, ocorrerá um fenômeno conhecido como *aliasing* (serrilhamento), onde as frequências do sinal que excedem a frequência de Nyquist aparecerão falsamente como frequências mais baixas. Como resultado, o sinal digital resultante será distorcido e não representará fielmente o sinal analógico original.

A Figura 6 ilustra o fenômeno de *aliasing*. Na esquerda, o sinal é amostrado com frequência de amostragem $f_s = 40$ Hz, enquanto que na direita a frequência de amostragem é $f_s = 8$ Hz. Nota-se que, em consequência do teorema de Nyquist-Shannon, o sinal da direita é incorretamente reconstruído como $\sin(4\pi t)$.

Figura 6 – Demonstração do fenômeno de *aliasing* no sinal $\sin(20\pi t)$.



Fonte: Autoria própria.

A importância do teorema de Nyquist-Shannon reside em sua capacidade de estabelecer a quantidade mínima de informação necessária para reconstruir fielmente um sinal contínuo. Esse resultado demonstra que arquivos de áudio armazenados acima de 44 kHz possuem todo o espectro de frequência audível pelo ser humano. Por conta disso, é comum arquivos de áudio conhecidos como *lossless* (sem perdas) serem armazenados a uma frequência de 48 kHz.

2.2.2 Representação de sinais na frequência

É muito comum, nos estudos da teoria de processamento de sinais, a presença de sinais periódicos. Como esses sinais podem ser compostos por diferentes frequências, é útil decompô-los em seus diferentes componentes para permitir análises mais apropriadas.

O espectro de frequência de um sinal é a representação de seu conteúdo de frequência - quais frequências estão presentes no sinal e quão fortes elas são. Para um sinal discreto, a representação no domínio da frequência fornece informações importantes sobre a composição do sinal em termos de suas componentes sinusoidais fundamentais.

A ferramenta matemática que permite mapear um sinal do domínio do tempo para o domínio da frequência é a Transformada de Fourier (FOURIER, 1807), definida como:

$$X(\omega) = \int_{n=-\infty}^{\infty} x(t)e^{-j\omega t}dt, \tag{5}$$

que resulta em uma função contínua de ω. Essa é uma das fórmulas mais importantes na engenharia elétrica, porque permite resolver diversos problemas no domínio da frequência. Para os sinais discretos é utilizada uma versão específica chamada Transformada de Fourier de Tempo Discreto (DTFT), cuja definição é:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \tag{6}$$

A DTFT resulta sempre em uma função periódica de período 2π. A magnitude de $X(e^{j\omega})$ determina a amplitude da componente de frequência correspondente no sinal, e a fase (o argumento de $X(e^{j\omega})$) determina a fase dessa componente de frequência. Essa equação torna possível que diversas sequências sejam representadas por meio de uma integral de Fourier da forma:

$$x[n] = \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega \tag{7}$$

A DTFT de uma sequência é uma função contínua de ω (ou, ainda, de $e^{j\omega}$). Se a sequência original for absolutamente somável (isto é, se a soma dos valores absolutos da sequência for finita), a DTFT convergirá e existirá para cada valor de ω.

Na prática, porém, a DTFT não é computacionalmente acessível, já que requer uma soma infinita e uma função contínua de frequência ω. Para contornar esse problema, é comum a utilização da Transformada de Fourier Discreta (DFT) (GAUSS, 1866), que amostra a DTFT do sinal em um número finito de pontos. A DFT de um sinal discreto $x[n]$ de N pontos é dada por:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \tag{8}$$

A DFT resulta em uma representação de frequência discreta de $X[k]$, o que a torna computacionalmente útil e amplamente usada em muitos algoritmos de processamento

de sinais digitais. Para torná-la ainda mais rápida, foi desenvolvido o algoritmo FFT (Transformada de Fourier Rápida), criado com o intuito de calcular a DFT de um sinal de maneira eficaz e com baixa complexidade computacional (a complexidade da FFT é de $O(n \log n)$, enquanto que a da DFT é $O(n^2)$). A FFT é uma das ferramentas mais utilizadas no processamento digital de sinais.

Essas representações são muito úteis para analisar um sinal em sua totalidade. Entretanto, por vezes deseja-se entender também como as diferentes frequências variam ao longo do tempo, principalmente para sinais que não são periódicos. Para isso, basta incluir uma soma de convolução nas definições da Transformada de Fourier:

$$X(\tau, \omega) = \int_{n=-\infty}^{\infty} x(t) w(t-\tau) e^{-j\omega t} dt, \tag{9}$$

$$X(m, e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] w[n-m] e^{-j\omega n}, \tag{10}$$

onde $w(t-\tau)$ e $w[n-m]$ é uma função de janela (que vale zero fora de um intervalo determinado) e τ e m são as variáveis temporais. Em geral, opta-se por uma janela de Hann ($w[n] = \sin^2(\frac{\pi n}{N})$) ou uma janela gaussiana ($w[n] = \exp(-\frac{1}{2}(\frac{n-N/2}{\sigma N/2})^2)$), com $0 \leq n \leq N$ e $\sigma \leq N$. A STFT é a responsável pela geração de espectrogramas como o da Figura 5. Para sinais de áudio de frequência de amostragem de 48 kHz, é comum escolher janelas de tamanho 2048 (equivalente a 42,6 ms).

2.2.3 Sistemas de tempo discreto

Em teoria de processamento de sinais, muito se estudam as transformações (ou operadores) de sinais. Tais transformações são chamadas de sistemas, cuja definição é: seja S o conjunto de todos os sinais, então um sistema T é qualquer mapeamento da forma $T : S \to S$. Na engenharia, é comum interpretar um sistema como uma "caixa preta", podendo possuir múltiplos sinais de entrada e saída.

Uma classe de sistemas de grande relevância na teoria de processamento de sinais são os sistemas LTI (Sistema Linear Invariante no Tempo), centrais na engenharia elétrica e na engenharia de controle e automação. Esses sistemas são definidos pelas seguintes propriedades:

- Linearidade: a relação entre a entrada $x[n]$ e a saída $y[n]$ é linear. Isto é, se $T(x_i) = y_i$ então $T(\alpha x_i + \beta x_j) = \alpha y_i + \beta y_j$ para quaisquer valores de α e β.

- Invariância no tempo: esta propriedade indica que o sistema não muda ao longo do tempo, de forma que a única diferença entre aplicar o sistema num sinal x no tempo 0 e no tempo τ é o atraso τ. Ou seja, $T(x_i[nT - \tau]) = y_i[nT - \tau]$.

Essas propriedades permitem o desenvolvimento de técnicas muito poderosas de análise e transformação de sinais como, por exemplo, a Transformada $\mathcal{Z}$ e a Transformada de Fourier. Entretanto, é importante destacar que a linearidade, apesar de facilitar o desenvolvimento teórico, é uma propriedade muito restritiva para um sistema qualquer. Grande parte da teoria de processamento de sinais é baseada em sistemas LTI, e mesmo a parcela que se dedica a estudar sistemas não lineares utiliza métodos de resolução que tentam aproximar tais sistemas em sistemas LTI. Na prática, os sistemas encontrados nos complexos problemas da realidade são não-lineares e, por vezes, caóticos.

Outra propriedade importante dos sistemas é a estabilidade. Um sistema é estável no sentido BIBO (Bounded-input, Bounded-output) se e somente se toda entrada limitada produz uma saída limitada. Em outras palavras, para todo valor de n existem os valores B_x e B_y tais que $|x[n]| < B_x < \infty$ e $|y[n]| < B_y < \infty$. Estes sistemas são particularmente relevantes em aplicações práticas, dado que uma saída ilimitada pode implicar no mau funcionamento ou na danificação de equipamentos valiosos.

Por fim, existe também a propriedade da causalidade. Um sistema é causal se e somente se, para todo valor n_0, a saída $y[n_0]$ depende apenas das entradas $x[n]$ quando $n \leq n_0$. Quase todos os sistemas utilizados em aplicações práticas são causais, dado que a não-causalidade implicaria em ter conhecimento de valores futuros de um sinal.

Para descrever um sistema T, é suficiente determinar sua resposta ao impulso. A resposta ao impulso é definida como a saída do sistema dado uma entrada $\delta[n]$, a função delta de Dirac (que pode ser interpretada como um impacto súbito de intensidade 1 para $n = 0$). Essa função vale 1 para $n = 0$ e 0 para todos os outros valores de n.

Para sistemas Lineares Invariantes no Tempo (LTI), a resposta ao impulso é extremamente importante por uma razão: qualquer entrada ao sistema pode ser descrita como uma combinação de impulsos deslocados e escalados no tempo. Graças à propriedade de invariância no tempo, é conhecido como o sistema responde a um impulso em qualquer ponto no tempo. Qualquer sinal $x[n]$ pode ser expresso como uma soma de impulsos da

seguinte forma:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \tag{11}$$

Devido à linearidade do sistema, se é conhecida a resposta do sistema a $\delta[n]$, é possível determinar sua resposta para a entrada $x[n]$ simplesmente somando as respostas a cada impulso individual $x[k]\delta[n-k]$. Para isso, utiliza-se a soma de convolução, que relaciona a entrada $x[n]$, a saída $y[n]$ e a resposta ao impulso $h[n]$ em um sistema LTI:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{12}$$

Esta operação é essencialmente uma média móvel ponderada de $x[n]$ onde os pesos são fornecidos por $h[n]$. Uma propriedade muito importante da soma de convolução é que, no domínio da frequência, ela se torna uma multiplicação simples. Isto é, se $X(e^{j\omega})$, $H(e^{j\omega})$ e $Y(e^{j\omega})$ são as Transformadas de Fourier de $x[n]$, $h[n]$ e $y[n]$ respectivamente, então:

$$Y(\omega) = X(\omega)H(\omega) \tag{13}$$

Isso significa que a resposta em frequência do sistema (a Transformada de Fourier da resposta ao impulso) $H(e^{j\omega})$ atua como um molde que modifica o espectro da entrada $X(\omega)$ para produzir o espectro de saída $Y(\omega)$. A conclusão disso é que é possível desenvolver filtros que atenuam frequências indesejadas.

Por exemplo, um filtro passa-baixas ideal é aquele que preserva totalmente as frequências abaixo de uma frequência de corte especificada e atenua totalmente as frequências acima dessa frequência de corte. A resposta ao impulso que corresponde a essa resposta em frequência é $h[n] = \text{sinc}[n] = \frac{\sin[n]}{n}$.

Entretanto, a função sinc se estende por toda a reta do tempo - seu suporte é infinito. A consequência disso é que filtros ideais não são causais, o que os torna irrealizáveis.

2.2.4 Filtros digitais

Anteriormente, foi demonstrado que a amostragem em taxas abaixo da frequência de Nyquist de um sinal implica em distorções indesejadas. Uma maneira de evitar essas distorções ao amostrar um sinal com uma frequência de amostragem abaixo da frequência de Nyquist é utilizar filtros digitais.

Filtros digitais são uma classe de sistemas LTI que são especificamente projetados para modificar um sinal de entrada atenuando ou amplificando faixas de frequência. Eles

são úteis, também, para remover ruídos/interferências e extrair/separar componentes do sinal. Existem vários tipos de filtros que são comumente usados, tais como:

1. Filtros passa-baixas (low-pass filters): Estes filtros permitem a passagem de frequências abaixo de uma certa frequência de corte, enquanto atenuam as frequências acima dela.

2. Filtros passa-altas (high-pass filters): Eles permitem a passagem de frequências acima de uma certa frequência de corte, enquanto atenuam as frequências abaixo dela.

3. Filtros passa-faixa (band-pass filters): Eles permitem a passagem de frequências dentro de uma certa faixa de frequências, enquanto atenuam as frequências fora dessa faixa.

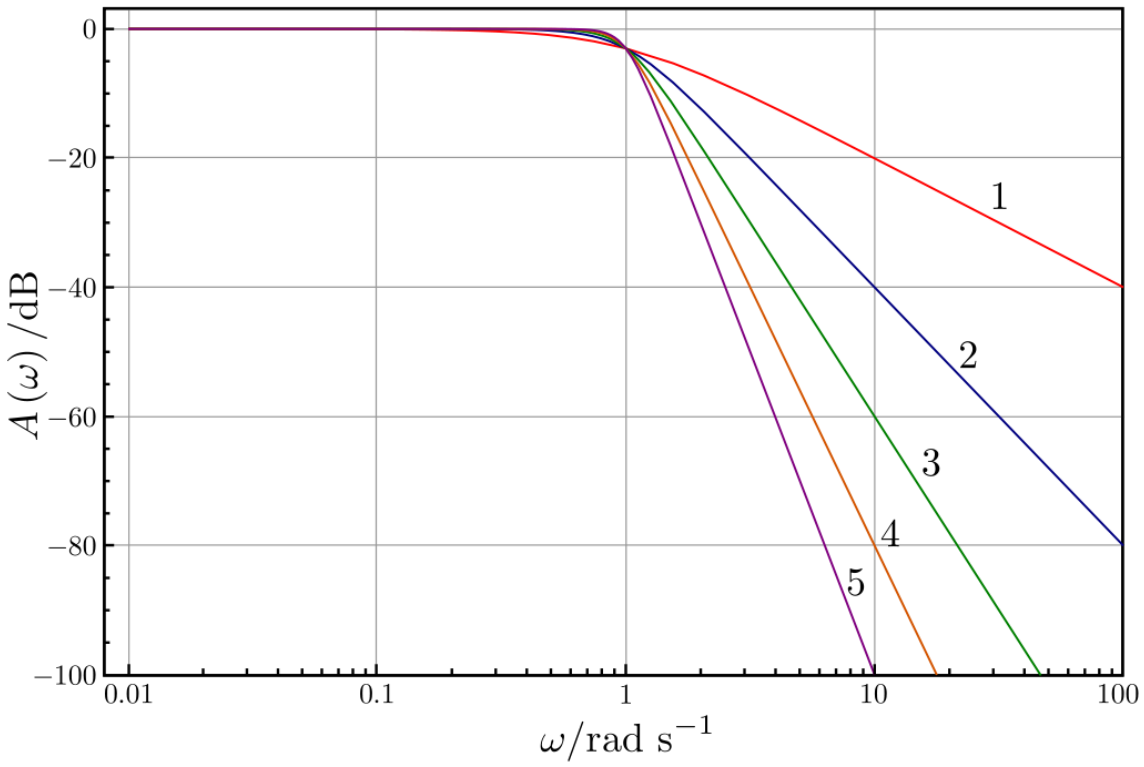
4. Filtros rejeita-faixa (band-stop filters, ou notch filters): Estes filtros atenuam as frequências dentro de uma certa faixa de frequências, permitindo a passagem de frequências fora dessa faixa.

Duas arquiteturas de filtros muito utilizadas na prática são o filtro de Butterworth e o filtro de Chebyshev de tipo I. O filtro de Butterworth de ordem n é definido pela função de transfência $G_n(\omega) = (1 + (\omega/\omega_0)^{2n})^{-1/2}$, enquanto que o filtro de Chebyshev de tipo I de ordem n é definido pela função de transferência $G_n(\omega/\omega_0) = (1 + \epsilon^2 T_n^2(\omega/\omega_0))^{-1/2}$. As figuras 7 e 8 apresentam as respectivas funções de transferência.

Apesar de apresentarem um decaimento acelerado, os filtros de Chebyshev tipo I distorcem levemente as frequências de passagem. Por outro lado, os filtros de Butterworth não decaem tão rapidamente, mas também não distorcem tanto as frequências mais baixas.

Finalmente, um filtro digital pode ser implementado de duas maneiras principais: implementação de resposta ao impulso finito (FIR), onde a saída é uma soma ponderada de entradas passadas; e implementação de resposta ao impulso infinito (IIR), onde a saída é uma soma ponderada de entradas e saídas passadas.

Figura 7 – Diagrama do ganho de filtros Butterworth de diferentes ordens.



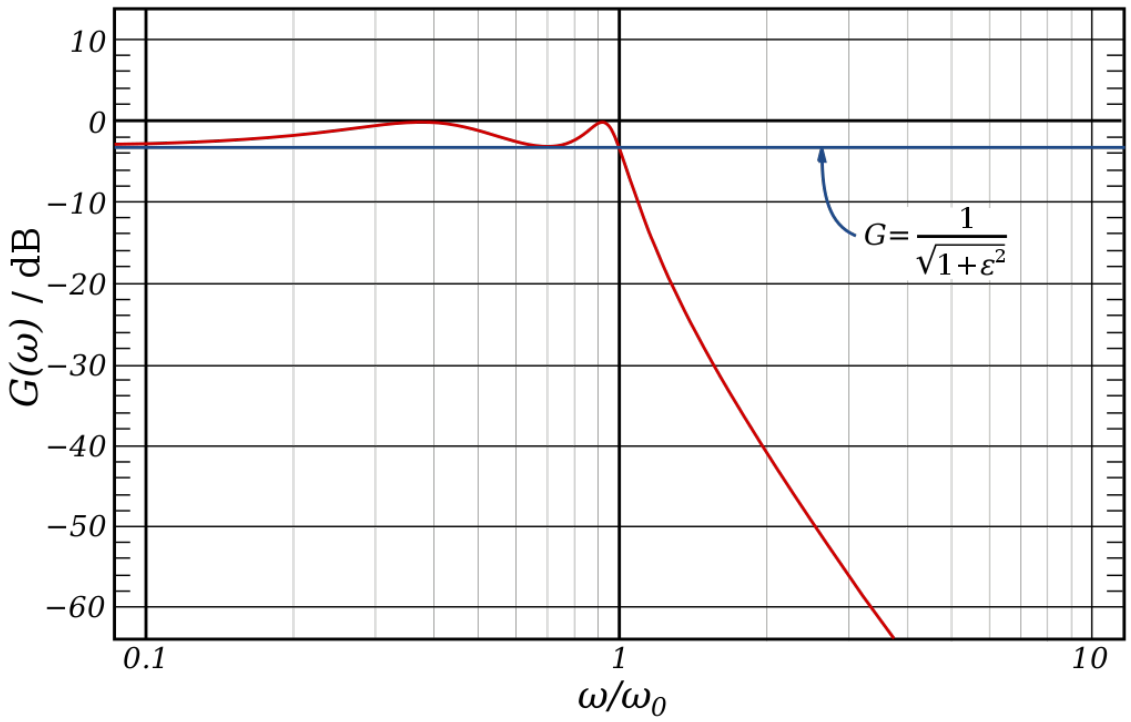
Fonte: WIKIPEDIA (2007).

2.3 Aprendizagem de Máquina

Antes de adentrar diretamente na teoria das redes neurais e aprendizagem profunda, é importante definir e fundamentar a teoria do aprendizado de máquina (*Machine Learning*). Entende-se por algoritmo de aprendizagem de máquina todo algoritmo que aprende a partir dos dados que lhe são fornecidos. Em outras palavras: "Um programa de computador é dito aprender a partir da experiência E com respeito a alguma classe de tarefas T e medida de performance P, se a sua performance nas tarefas em T, conforme medida por P, melhora com a experiência E" (MITCHELL, 1997). Um exemplo disso é um robô que aprende a somar: a tarefa é acertar o resultado da soma de dois números, a experiência é o robô observar diversas operações de soma e a medida de performance é o quão próximo o robô chegou ao resultado correto.

A aprendizagem de máquina possibilita a resolução de tarefas que são complexas para serem solucionadas com programas rígidos, projetados por humanos. Dentro desse conjunto, enquadram-se problemas como, por exemplo, o reconhecimento de carros em uma imagem ou a transcrição de fala a partir de arquivos de áudio. Para os seres humanos,

Figura 8 – Diagrama do ganho de filtros Chebyshev tipo I de ordem 1 e $\epsilon = 1$.



Fonte: WIKIPEDIA (2009).

essas são tarefas simples e cotidianas, contudo, é importante ressaltar que elas requerem uma base de conhecimento do mundo muito extensa e são mais facilmente descritas de forma intuitiva. Em outras palavras, a aprendizagem de máquina viabiliza a execução de tarefas intuitivas que são inviáveis por meio de soluções algorítmicas convencionais.

Tais tarefas são comumente descritas em termos de como um sistema de aprendizagem de máquina deve processar um exemplo. Entende-se por exemplo uma coleção de características de um objeto ou evento que se deseja que o sistema de aprendizagem de máquina aprenda. Um exemplo é representado como um vetor $x \in R^n$, no qual cada elemento x_i do vetor é uma característica. Por exemplo, as características de uma imagem podem ser os valores das cores e brilho de seus pixels.

Diversas classes de tarefas podem ser executadas com a utilização de aprendizagem de máquina. Entre as mais comuns estão: classificação, regressão, transcrição, tradução, detecção de anomalias, síntese, amostragem e remoção de ruído. Com o passar do tempo, o número de tarefas solucionadas por meio de aprendizagem de máquina tem apresentado crescimento acelerado, evidenciando a fertilidade deste campo de estudo.

Já a medida de performance P depende da classe da tarefa a ser resolvida. Por exem-

plo, é comum medir-se a acurácia (ou taxa de acerto) de um modelo para tarefas como classificação e transcrição. A acurácia é uma medida de performance estatística que pode assumir valores no intervalo $[0, 1] \in \mathbb{R}$ e indica a probabilidade de, dada uma entrada válida, o modelo classificá-la ou transcrevê-la corretamente.

Em geral, busca-se determinar quão bem o modelo de aprendizagem de máquina resolve o problema com um exemplo que não foi anteriormente visto por ele. Portanto, avaliam-se essas medidas de performance em dados pertencentes ao conjunto conhecido como conjunto de teste, que contém exemplos nunca antes vistos pelo modelo. Este conjunto e a acurácia do modelo a ele associado permitem estimar a acurácia do modelo em aplicações práticas da vida real que possuam as mesmas características dos exemplos do conjunto de teste. A prática de utilizar um conjunto de teste é muito relevante para diferentes técnicas de aprendizado de máquina.

A maior parte dos algoritmos de aprendizado de máquina pode ser entendida como capaz de observar um grande conjunto de dados, cujos exemplos são chamados de pontos de dados. O ato de observar um conjunto de dados também é conhecido como treinamento de um modelo, e o conjunto de dados associado é chamado de conjunto de treinamento. Existem infinitas maneiras diferentes de realizar o processo de treinamento, e busca-se comumente a metodologia que resulta no modelo mais performático e treinável em quantidades razoáveis de tempo.

Por fim, a experiência de um sistema de aprendizado de máquina o caracteriza como supervisionado ou não supervisionado. Os algoritmos de aprendizado não supervisionado observam um conjunto de dados que contém diversas características e aprendem propriedades úteis da estrutura desse conjunto de dados. No contexto de aprendizado profundo, que será abordado posteriormente, o objetivo geralmente é aprender toda a distribuição de probabilidade que gerou o conjunto de dados, seja de forma implícita ou explícita.

Já os algoritmos de aprendizado supervisionado observam um conjunto de dados que, além de conter diversas características, também possui associado a cada exemplo um rótulo ou alvo. Esses rótulos indicam qual seria a saída desejada do modelo caso os pontos de dados a eles associados sejam fornecidos como entrada. Por exemplo, numa tarefa de classificação de imagens de gatos e cachorros, associa-se a cada imagem o rótulo "gato"ou "cachorro"para que o modelo aprenda a distinguir gatos de cachorros.

O termo aprendizado supervisionado origina-se da visão de um instrutor ou professor

fornecendo ao sistema de aprendizado de máquina a resposta de antemão, enquanto que no aprendizado não supervisionado a máquina deve aprender as características dos dados por conta própria.

Em termos práticos, aprendizado não supervisionado envolve a observação de diversos exemplos de um vetor aleatório x e a derivação de sua distribuição de probabilidade $p(x)$ ou propriedades interessantes dela, enquanto que aprendizado supervisionado envolve a observação de diversos exemplos de um vetor aleatório x e um valor associado y com o objetivo de aprender a prever y a partir de x, normalmente estimando a distribuição $p(y|x)$. (GOODFELLOW; BENGIO; COURVILLE, 2016)

Outras variantes de paradigmas de aprendizado também são possíveis. Entre as mais relevantes estão os algoritmos de aprendizado de reforço. Estes algoritmos interagem com o ambiente no qual estão inseridos, havendo um ciclo de retroalimentação entre o sistema de aprendizado e suas experiências. Este paradigma é muito útil no campo da robótica e em jogos/simulações digitais.

2.3.1 Capacidade, sobre-ajuste e subajuste

O problema central da otimização de algoritmos de aprendizado de máquina é que se deseja que tais algoritmos apresentem a melhor performance possível em pontos de dados nunca antes vistos, isto é, deseja-se maximizar o poder de generalização do modelo.

No desenvolvimento de algoritmos de aprendizado de máquina, o primeiro passo é definir a base de dados a ser utilizada. Esta base de dados é dividida em dois subconjuntos: o conjunto de treinamento, que contém a maior parte dos dados, e o seu complementar, chamado de conjunto de testes. Além desses, existe ainda o conjunto de validação, que costuma ser um subconjunto do conjunto de treinamento. O conjunto de validação serve para otimizar os hiper-parâmetros (todos os parâmetros que o modelo não aprende) antes de treinar o algoritmo em todo o conjunto de treinamento.

O conjunto de treinamento possui a ele associada uma taxa de erro, calculada a partir de alguma medida de erro estabelecida para cada iteração do processo de aprendizado do modelo. Essa taxa é chamada de taxa de erro de treinamento, e almeja-se minimizá-la durante o estágio de treinamento de forma que o modelo aprenda as características fundamentais dos dados utilizados.

Na prática, o que realmente importa é a taxa de erro de generalização (também cha-

mada de taxa de erro de teste), definida como a taxa de erro do modelo quando se utilizam dados que não pertençam ao conjunto de treinamento. Essa taxa de erro é definida como o valor esperado do erro em novas entradas, e para calculá-la é necessário aplicar o modelo em um conjunto de dados cuja distribuição é a mais próxima possível da que se espera que o modelo encontre na prática. Por definição, esse conjunto é o conjunto de testes.

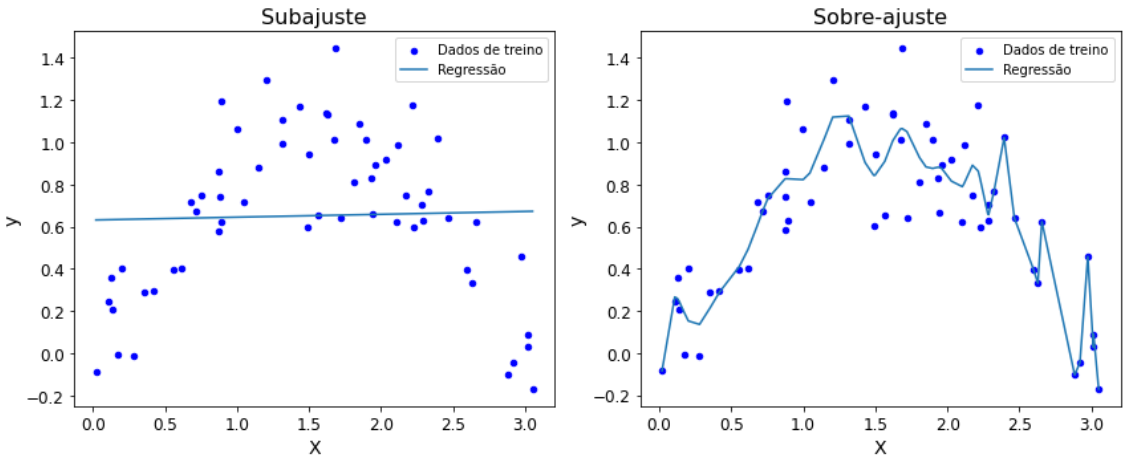
É importante, porém, ter cuidado ao dividir a base de dados nos conjuntos de treinamento e de teste. Se os dados forem alocados arbitrariamente, haveria uma grande dificuldade de otimizar a taxa de generalização. Graças ao campo da teoria do aprendizado estatístico, é possível alocar os dados de forma a estabelecer uma relação entre a taxa de erro de treinamento e a taxa de generalização. Tal procedimento estabelece as suposições de que os exemplos em cada conjunto são independentes e identicamente distribuídos. Assim, as distribuições de probabilidade que descrevem ambos os conjuntos são idênticas.

Essa distribuição de probabilidade é denominada distribuição geradora de dados, denotada por p_{dados}. Uma conclusão imediata é que a taxa de erro esperada para ambos os conjuntos é a mesma.

Porém, isso só seria possível se já se soubessem de antemão os parâmetros do modelo. Na prática, é necessário primeiro amostrar o conjunto de treinamento para depois serem obtidos os parâmetros que minimizam a taxa de erro de treinamento, para finalmente ser amostrado o conjunto de testes. Por conta deste processo, o valor esperado da taxa de erro de generalização é maior ou igual ao valor esperado da taxa de erro de treinamento. Portanto, os fatores determinantes da performance de um modelo de aprendizado de máquina são a sua habilidade de minimizar a taxa de erro de treinamento e a sua habilidade de minimizar a diferença entre a taxa de erro de treinamento e a taxa de erro de generalização.

Estes dois fatores correspondem a dois desafios centrais do aprendizado de máquina: subajuste (*underfitting*) e sobre-ajuste (*overfitting*). O subajuste ocorre quando o modelo é incapaz de obter uma taxa de erro de treinamento baixa, e o sobre-ajuste ocorre quando a diferença entre a taxa de erro de treinamento e a taxa de generalização é muito grande.

Figura 9 – Visualização do subajuste e do sobre-ajuste.



Fonte: Autoria própria.

Controla-se se um modelo irá se subajustar ou sobre-ajustar alterando sua capacidade. A capacidade de um modelo é definida como sua habilidade de se ajustar a uma grande variedade de funções. Modelos com baixa capacidade são subajustados, e modelos com alta capacidade são sobre-ajustados.

Para controlar a capacidade, escolhe-se o espaço de hipóteses do modelo, que é o conjunto de funções possíveis de serem selecionadas como solução pelo modelo. Como um exemplo, para um algoritmo de regressão linear, o espaço de hipóteses é o conjunto de todas as funções lineares. Se, em vez disso, se generalizar esse espaço para incluir o conjunto de todas as funções polinomiais, aumenta-se a capacidade do modelo.

Algoritmos de aprendizado de máquina apresentam a melhor performance quando sua capacidade é apropriada para a verdadeira complexidade da tarefa a ser executada e para a quantidade de dados de treinamento fornecida. Modelos com capacidade insuficiente não são capazes de resolver tarefas complexas, e modelos com capacidade excessiva sobre-ajustam e se tornam inadequados para resolver a tarefa desejada. Por exemplo, ao se efetuar uma regressão num conjunto de n pontos de dados, pode-se escolher um polinômio de grau arbitrário. Quando o grau do polinômio é maior que n, existem infinitas soluções para o problema de regressão, e o modelo terá uma probabilidade nula de escolher a função de regressão mais adequada.

A capacidade não é determinada apenas pela escolha do modelo. O modelo especifica a família de funções que o algoritmo de aprendizado pode escolher quando seus parâmetros são variados a fim de reduzir um objetivo de treinamento. Isso é chamado de

capacidade representacional. Contudo, encontrar a função que melhor se ajusta à tarefa dentre as funções dessa família é um problema de otimização difícil. Na prática, por conta de limitações adicionais, a capacidade efetiva de um modelo de aprendizado é menor que a capacidade representacional da família de funções do modelo.

A teoria do aprendizado estatístico fornece algumas maneiras de quantificar a capacidade de diferentes modelos. O exemplo mais conhecido é a dimensão de Vapnik-Chernovenkis, que mede a capacidade de um classificador binário. Ela é definida como o maior valor de m tal que exista um conjunto de treinamento de m pontos que o classificador pode rotular arbitrariamente.

Quantificar a capacidade dos modelos permite que a teoria do aprendizado estatístico realize previsões quantitativas. Os resultados mais importantes dessa teoria demonstram que a diferença entre a taxa de erro de treinamento e a taxa de erro de generalização é limitada por uma quantidade que cresce conforme a capacidade do modelo aumenta e diminui conforme o número de exemplos de treinamento aumenta. Esses limites fornecem uma justificativa formal de que algoritmos de aprendizado de máquina podem funcionar. Infelizmente, para algumas famílias de modelos, como métodos de aprendizagem profunda (redes neurais), existe uma grande dificuldade de quantificar as suas capacidades devido ao pequeno conhecimento teórico dos problemas de otimização não convexos envolvidos.

Esses resultados indicam que é necessário ter cautela ao escolher um modelo e considerar sua capacidade. Tipicamente, a taxa de erro generalização como função da capacidade tem um formato parabólico (formato de U), possuindo um valor mínimo que se deseja obter durante a modelagem do algoritmo de aprendizado de máquina.

Para alcançar os casos mais extremos de capacidade, podem-se conceituar modelos não-paramétricos. Enquanto que modelos paramétricos aprendem uma função descrita por um vetor-parâmetro cujo tamanho é finito e fixado antes de qualquer dado ser observado, os modelos não-paramétricos não têm tal limitação. Normalmente esses modelos são abstrações teóricas, mas pode-se projetar modelos não-paramétricos fazendo a complexidade ser uma função do tamanho do conjunto de treinamento. Um exemplo é a regressão do vizinho mais próximo. Enquanto que a regressão linear armazena os pesos de um vetor de tamanho fixo, o modelo de regressão do vizinho mais próximo simplesmente armazena as entradas $\mathbf{X}$ e o vetor de rótulos $\mathbf{y}$ do conjunto de treino. Ao se querer classificar um ponto de teste $\mathbf{x}$, o modelo simplesmente procura a entrada do conjunto

de treinamento mais próxima e retorna o rótulo associado. Aqui, proximidade pode ser qualquer métrica de distância, que pode inclusive ser aprendida.

É possível também criar um algoritmo de aprendizado não-paramétrico ao envolver um algoritmo paramétrico dentro de outro algoritmo que aumenta o número de parâmetros conforme necessário. Pode-se imaginar um laço externo de aprendizado que modifica o grau de um polinômio aprendido por um algoritmo de regressão linear, por exemplo.

Por fim, é importante mencionar que até mesmo o modelo de aprendizado de máquina ideal pode conter algum erro em diversos problemas. Isso ocorre porque o modelo ideal é simplesmente um oráculo que conhece a verdadeira distribuição de probabilidade que gerou os dados, e mesmo essa distribuição pode conter algum ruído. A taxa de erro correspondente às previsões desse oráculo que divergem da verdadeira distribuição $p(\mathbf{x}, y)$ é chamada de erro de Bayes.

2.3.2 O Teorema da Inexistência do Almoço Grátis e Regularização

Um resultado crucial da teoria do aprendizado de máquina é o Teorema da Inexistência do Almoço Grátis (WOLPERT, 1996). Este teorema demonstra que, quando a performance de todos os métodos de otimização é avaliada em todos os problemas concebíveis quando lhes é fornecido dados e tempo de treinamento infinitos, todos eles apresentam o mesmo desempenho. Em outras palavras, não existe um algoritmo de otimização ideal - consequentemente, não há um método de aprendizado de máquina perfeito para tarefas de modelagem preditiva, como classificação e regressão.

Esse resultado sugere que não há um atalho para resolver os problemas. Ou seja, se um modelo for escolhido arbitrariamente para tentar resolver todos os problemas existentes, a performance média do modelo será a mesma que determinar o resultado aleatoriamente.

Felizmente, essa afirmação só é verdadeira quando se realiza a média sobre todas as distribuições geradoras de dados possíveis. Ao se fazer algumas suposições sobre os tipos de distribuição de probabilidade encontrados em aplicações práticas, torna-se possível projetar algoritmos de aprendizado que apresentam um bom desempenho nestas distribuições.

A conclusão desse teorema é que o objetivo da pesquisa de algoritmos de aprendizado de máquina não é a busca pelo algoritmo de aprendizado universal porque tal modelo não existe. Em vez disso, busca-se entender que tipos de distribuições são relevantes para um

agente de inteligência artificial e que tipos de algoritmos de aprendizado performam bem nos dados obtidos dessas distribuições geradoras de dados relevantes.

Esses resultados indicam a necessidade de se projetar um algoritmo de aprendizado de máquina eficiente para a tarefa em questão. Para isso, uma série de restrições e preferências são aplicadas ao algoritmo. É possível priorizar certas famílias de soluções em relação a outras no espaço de hipóteses, de maneira que uma família de funções de baixa prioridade seria selecionada apenas se apresentar um ajuste consideravelmente melhor que as famílias preferenciais.

Esse processo é chamado de regularização. Regularizar um modelo que aprende uma função $f(x, \theta)$ pode ser feito adicionando-se uma penalidade chamada regularizador à função de custo durante o processo de treinamento. Um exemplo é o decaimento de pesos, cujo regularizador é $\Omega(w) = \lambda w^T w$ no caso de regressão linear. Aqui, w é o vetor dos pesos e λ é uma constante de decaimento de pesos. Quando $\lambda = 0$, nenhuma preferência é imposta, e quanto maior o seu valor, menores serão os coeficientes polinomiais, resultando em uma curva mais suave e linear.

A utilização de regularizadores representa uma forma mais geral de controlar a capacidade de um modelo. A regularização é, portanto, qualquer modificação feita em um algoritmo de aprendizado cujo objetivo é reduzir a sua taxa de generalização, mas não a sua taxa de erro de treinamento. Regularização e otimização são as preocupações mais importantes no campo de aprendizado de máquina. Uma vasta gama de tarefas pode ser resolvida efetivamente utilizando formas de regularização de propósito geral.

2.3.3 Hiper-parâmetros e Conjuntos de Validação

Os hiper-parâmetros correspondem às configurações utilizadas para controlar o comportamento de um modelo. Representam valores e estruturas que o modelo não pode modificar, embora seja viável um algoritmo de aprendizado de máquina otimizar os hiper-parâmetros de um determinado modelo. No caso da regressão linear, o grau do polinômio é o único hiper-parâmetro, e no caso dos regularizadores abordados anteriormente, λ também se configura como um hiper-parâmetro.

Frequentemente, uma configuração deve ser adotada como hiper-parâmetro pois não é apropriado que o modelo aprenda tal configuração. Essa situação é aplicável a todos os hiper-parâmetros que controlam a capacidade dos algoritmos de aprendizado. Se tais

hiper-parâmetros fossem aprendidos durante a fase de aprendizado, sempre seriam escolhidos valores que maximizam a capacidade do modelo, resultando em um sobre-ajuste indesejado.

Para resolver esse problema, faz-se necessário a utilização de um conjunto de validação de exemplos que o algoritmo de treinamento não observe. Este conjunto é construído a partir dos dados de treinamento, com o objetivo de auxiliar na estimação da taxa de erro de generalização durante o processo de treinamento, permitindo a atualização dos hiper-parâmetros de maneira apropriada. Normalmente, destina-se uma parte pequena (entre 1% a 20%, dependendo do tamanho da base de dados) do conjunto de treinamento para a validação.

Um possível problema que pode surgir está relacionado ao tamanho do conjunto de treinamento. Quanto menor o conjunto de testes, maior a incerteza estatística na estimativa do valor esperado da taxa de generalização, dificultando a comparação entre diferentes algoritmos no mesmo conjunto de dados. Nestas situações, uma possível solução é repetir o processo de aprendizado para diferentes subconjuntos de treinamento e de testes da base de dados disponível, o que implica em um custo computacional mais elevado.

2.3.4 Algoritmos de Aprendizado Supervisionado

Os algoritmos de aprendizado supervisionado correspondem aos que aprendem a associar uma entrada a um rótulo dos exemplos da base de dados (ALBON, 2018). Tais rótulos podem ser coletados manualmente (com inserção manual dos rótulos por humanos) ou de maneira automatizada.

Uma classe significativa de aprendizado supervisionado é o aprendizado probabilístico. Esta estratégia é baseada na estimativa da distribuição de probabilidade $p(y|x)$ ao utilizar o método de estimativa da máxima verossimilhança para encontrar o vetor de parâmetros θ mais adequado para uma família de distribuições $p(y|x, \theta)$. Em estatística, para uma determinada família de funções $f(x|\theta)$, o estimador de máxima verossimilhança é definido como:

$$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} \sum_{i=1}^{n} \ln f(x_i|\theta). \tag{14}$$

O aprendizado probabilístico é responsável pela generalização do algoritmo de regressão linear para um algoritmo de classificação chamado regressão logística. A regressão logística não possui uma solução analítica para seus parâmetros ótimos, mas pode ser vi-

abilizada ao maximizar a verossimilhança. Esse caso é aplicável para a regressão de qualquer família de funções, e a estratégia de aprendizado probabilístico pode ser utilizada para solucionar a maioria dos problemas de aprendizado supervisionado ao se selecionar uma família de parametrização adequada.

Um algoritmo importante de aprendizado supervisionado são as máquinas de vetores de suporte (SVM). Elas se assemelham à regressão logística, compostas por uma função linear $w^T x + b$. O objetivo é a classificação binária, e o processo de treinamento busca encontrar o hiperplano entre dados de duas classes distintas que maximiza a distância entre os pontos mais próximos do hiperplano. Assim, para um exemplo do conjunto de testes, avalia-se sua posição relativa ao hiperplano para classificá-lo.

Uma importante inovação originada das pesquisas de SVM's é o truque do núcleo (*kernel trick*). Basicamente, o truque do núcleo é um método que permite introduzir não-linearidades no espaço de características. Aplica-se uma função de característica $\phi(x)$ no produto escalar $k(x, x^{(i)}) = \phi(x) \cdot \phi(x^{(i)})$ conhecido como núcleo, que possibilita a transformação do espaço dos exemplos em outro mais facilmente analisável pelo modelo. Um exemplo clássico é determinar se um ponto bidimensional está dentro ou fora de um disco de raio 1 fixado na origem. Observar apenas uma das coordenadas do ponto (x, y) não é suficiente para resolver essa tarefa, mas ao se utilizar o núcleo $\phi((x, y)) = (x, y, x^2 + y^2)$, o problema torna-se trivial, bastando apenas comparar a terceira coordenada a 1. Este truque possibilita que algoritmos de aprendizado de máquina aprendam funções ou fronteiras de decisões não-lineares.

Por fim, outros tipos de modelos supervisionados, como árvores de decisão e k-vizinhos mais próximos, são relevantes para diversas aplicações práticas. Normalmente, essas aplicações correspondem a tarefas de classificação, distinção e identificação, problemas muito relevantes no mundo moderno.

2.3.5 Algoritmos de Aprendizado Não-Supervisionado

De forma geral, o aprendizado não-supervisionado refere-se à extração de informações de uma distribuição que não requer trabalho humano para anotar e rotular exemplos. Este termo é frequentemente associado à estimação de densidade, ao aprendizado para gerar amostras de uma distribuição, à aprendizagem para remover o ruído de uma distribuição ou ao agrupamento de dados em conjuntos de exemplos relacionados entre si.

Uma tarefa não-supervisionada clássica é a busca pela melhor representação dos dados. Ou seja, busca-se uma representação que preserve a máxima informação de x possível, obedecendo uma certa penalidade ou restrição que torna a representação mais simples ou mais acessível que x.

Existem várias maneiras de se definir uma representação mais simples. Três das mais comuns são as representações de menor dimensão (que visam comprimir a máxima informação de x possível), as representações esparsas (que transformam o conjunto de dados numa representação cujos valores são majoritariamente nulos) e as representações independentes (que visam desenrolar as fontes de variação contidas na distribuição de dados de tal maneira que cada dimensão da representação seja estatisticamente independente).

O conceito de representação é um dos temas centrais do aprendizado profundo, que será apresentado posteriormente. A Análise de Componentes Principais (PCA) é uma das principais metodologias que utilizam este conceito e pode ser considerada como um algoritmo não-supervisionado que aprende uma representação com menor dimensionalidade que a entrada, cujas dimensões não possuem correlações lineares entre si. Entretanto, elas ainda não são completamente independentes, devido às correlações não-lineares entre as variáveis.

A PCA aprende uma transformação linear e ortogonal que projeta um vetor x para outro vetor z em uma base (representação) diferente. Essa transformação linear é fornecida pela matriz W que torna diagonal a matriz de covariância de X, definida por:

$$Var(x) = \frac{X^T X}{m-1}, \tag{15}$$

onde m é a dimensão de X. A matriz W, que permite diagonalizar a matriz de covariância, é única e pode ser calculada utilizando a matriz diagonal de autovalores Λ pela equação:

$$X^T X = W \Lambda W^T. \tag{16}$$

A PCA é amplamente utilizada em análises exploratórias e para construir modelos preditivos. Suas aplicações práticas incluem genética populacional, estudos de microbiomas e ciência atmosférica.

Por fim, outro algoritmo de aprendizado de máquina não-supervisionado relevante é o agrupamento em k-médias. A meta deste algoritmo é dividir um conjunto de dados em k agrupamentos de dados similares entre si, resultando em um diagrama de Voronoi. Ele

funciona inicializando k centroides diferentes e alternando entre dois passos até a convergência: no primeiro, atribui-se um grupo a cada exemplo de treinamento; no segundo, atualiza-se a posição de cada centroide para a média de todos os exemplos de treinamento atribuídos a ele.

Um dos desafios deste algoritmo é interpretar e entender as propriedades de cada agrupamento resultante e como essas propriedades se traduzem nas aplicações práticas. Além disso, existem múltiplas configurações de agrupamentos válidos de tamanho k que podem surgir a partir do algoritmo, todas igualmente válidas. Por exemplo, na tentativa de separar em dois grupos imagens que podem ser bananas, maçãs, carros vermelhos e caminhões amarelos, um resultado possível seriam agrupamentos baseados em cores, enquanto outro seria agrupamentos baseados em frutas ou automóveis.

Existem várias variações do algoritmo de k-médias, a maioria delas obtidas ao se usar uma medida de tendência central diferente ou uma medida de distância diferente. Por exemplo, a mediana ou a média geométrica podem ser utilizadas no lugar da média aritmética.

2.3.6 Gradiente Descendente Estocástico

Antes de se avançar para o estudo de redes neurais, é crucial compreender um algoritmo de suma importância: o gradiente descendente estocástico (SGD). Este método visa minimizar ou maximizar uma função-objetivo, também conhecida como critério. Quando essa função é minimizada, pode-se referir a ela como função de custo, função de perda ou função de erro.

A premissa do gradiente descendente é reduzir $f(x)$ deslocando x de maneira iterativa em pequenos passos na direção oposta do seu gradiente, $\nabla f(x)$, buscando a convergência. Idealmente, espera-se que o algoritmo encontre o mínimo global de $f(x)$. Contudo, por vezes, o resultado obtido é um mínimo local. O método é descrito pela equação:

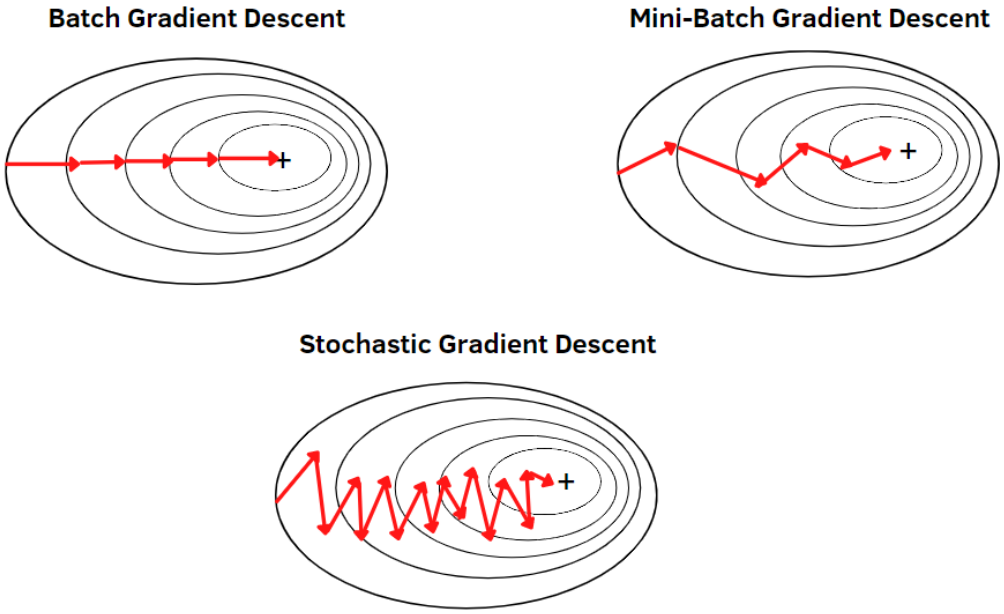
$$x' = x - \epsilon \nabla_x f(x) \tag{17}$$

, onde ϵ é uma constante que determina a magnitude de cada passo entre as iterações. Esta constante é também conhecida como taxa de aprendizado e é um dos hiper-parâmetros mais importantes dos algoritmos de aprendizado de máquina que utilizam a otimização por gradiente descendente.

O método do gradiente descendente estocástico é uma extensão do algoritmo acima

descrito, que procura minimizar o número de computações necessárias do algoritmo. Um dos problemas recorrentes em aprendizado de máquina é que, embora conjuntos de dados grandes sejam necessários para uma boa generalização, o seu tamanho também implica um maior poder computacional. Comumente, a função de custo de um algoritmo de aprendizado de máquina se decompõe como uma soma sobre os exemplos de treinamento de alguma função de perda associada a cada exemplo. Para essas funções de custo aditivas, o gradiente descendente requer o cálculo, para cada exemplo, do gradiente da função de perda em função dos parâmetros a serem obtidos. À medida que o tamanho da base de dados aumenta, o tempo para cada passo de gradiente torna-se proibitivamente longo.

Figura 10 – Diferentes técnicas de descida do gradiente.



Fonte: WAGH (2022).

O SGD resolve esse problema de uma maneira astuta. Utiliza-se o fato de que o gradiente é uma medida de esperança para aproximá-lo por meio de algumas amostras retiradas de forma uniforme do conjunto de treinamento. O tamanho m' desses lotes de amostras é escolhido de maneira a ser relativamente pequeno em relação à base de dados, tipicamente entre apenas uma amostra e algumas centenas. Assim, o gradiente estimado é calculado através da equação:

$$g = \frac{1}{m'} \nabla_\theta \sum_{i=1}^{m'} L(x^{(i)}, y^{(i)}, \theta), \tag{18}$$

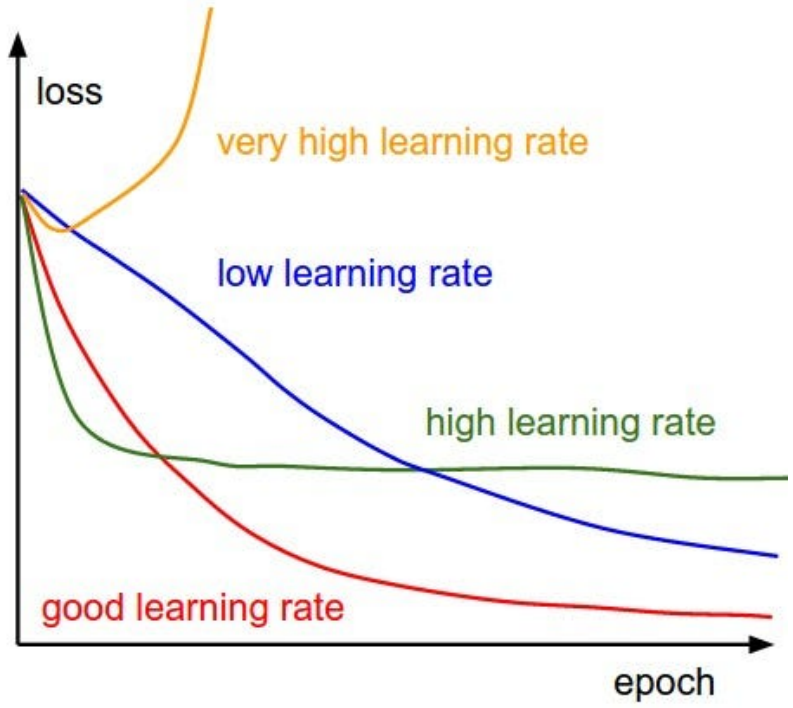
onde $L(x^{(i)}, y^{(i)}, \theta)$ é a função de perda utilizada e m' é o número de amostras utilizadas. Dessa forma, o algoritmo SGD segue a aproximação do gradiente estimado pela iteração de

$$\theta \leftarrow \theta - \epsilon g. \tag{19}$$

Na prática, a taxa de aprendizado utilizada não precisa ser uma constante, mas um valor que decresce a cada iteração e é denotado por ϵ_k. Isso ocorre porque o estimador do gradiente introduz uma fonte de ruído (a seleção aleatória de m' amostras) que não desaparece mesmo quando se atinge um mínimo. Uma condição suficiente para assegurar a convergência do SGD é que as somas de todos os ϵ_k e de ϵ_k^2 seja limitada. É comum modelar o algoritmo de aprendizado de forma que a taxa de aprendizado decaia linearmente a cada iteração até atingir uma constante que costuma ser aproximadamente $\epsilon_0/100$.

A escolha da taxa de aprendizado pode ser realizada por tentativa e erro, porém é mais apropriado fazê-lo monitorando as curvas das função de perda de treinamento e validação. Não existe um método ideal para otimizar a escolha do valor de ϵ. Conforme visualizado na Figura 11, um taxa de aprendizado muito grande faz o modelo divergir durante o treinamento, enquanto que uma taxa de aprendizado muito pequena faz o modelo convergir muito lentamente.

Figura 11 – Efeitos de diferentes taxas de aprendizado no treinamento de um modelo.



Fonte: RAKHECHA (2019).

A maior vantagem do SGD é que o número de cálculos por iteração não cresce com o aumento do número de pontos de dados do conjunto de treinamento. Isso permite a convergência mesmo quando o número de exemplos é muito grande.

2.4 Aprendizado Profundo

O aprendizado profundo é um dos subconjuntos mais importantes do aprendizado de máquina atualmente. Para um algoritmo de aprendizado de máquina ser caracterizado como aprendizado profundo, é necessário que ele pertença à classe de redes neurais artificiais que possuem três ou mais camadas. Esses algoritmos são inspirados no comportamento do cérebro humano de uma forma muito simplificada para resolver problemas a partir de grandes quantidades de dados. Essa classe de algoritmo é capaz de assumir arquiteturas muito complexas, permitindo-a ser utilizada na resolução de inúmeros problemas diferentes. Devido à sua composição, métodos de aprendizado profundo são adequados em tarefas que podem ser simplificadas como o mapeamento de um vetor em outro. Para as outras tarefas ainda mais complexas, o aprendizado profundo torna-se insuficiente.

Uma das grandes diferenças entre o aprendizado profundo e outras técnicas de aprendizado de máquina são os dados utilizados para treinar o algoritmo. Enquanto que para algoritmos de aprendizado de máquina as características específicas dos dados são definidas de antemão, os algoritmos de aprendizado profundo reduzem o pré-processamento necessário. Esses algoritmos são capazes de aprender a partir de dados não estruturados como texto e imagens, além de automatizarem a extração de características - removendo, assim, a dependência de especialistas de domínio na modelagem do algoritmo. Além disso, redes neurais artificiais são capazes de aproximar funções não-lineares arbitrárias, uma característica extremamente poderosa dessa classe de algoritmos.

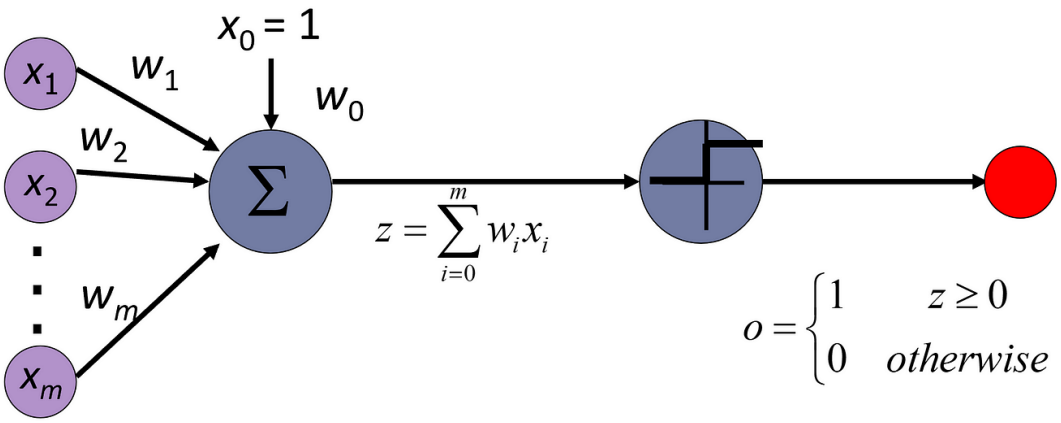
O exemplo fundamental do aprendizado profundo é o *perceptron* multicamadas (Multilayer Perceptron, ou MLP), também conhecido como rede neural pró-alimentada. Por ser a base da grande maioria das redes neurais modernas, entender os princípios e funcionamento dessa estrutura é de grande importância para o entendimento e a aplicação dos diversos métodos de redes neurais existentes.

2.4.1 O *perceptron* e redes neurais artificiais

O *perceptron*, visualizado na Figura 12, é uma modelagem de um neurônio biológico desenvolvida nos anos 50 (ROSENBLATT, 1958) que pode ser descrito como uma função que mapeia um vetor de tamanho arbitrário em uma saída unidimensional.

Quando em uma rede neural artificial de diversas camadas, um *perceptron* h_i da camada i retorna o valor $h_i = g(\mathbf{W} \cdot \mathbf{X} + c_i)$, onde $g(\cdot)$ é uma função não-linear conhecida como função de ativação, $\mathbf{W}$ é o vetor de pesos (ponderações) da entrada, X é o vetor das saídas dos perceptrons da camada anterior e c_i é chamado de parâmetro de viés, devido ao fato de que, na ausência de entradas, o perceptron estará enviesado para $g(c_i)$. É importante não confundir esse parâmetro com o termo viés estatístico, que é a esperança de um estimador para alguma quantidade que difere da esperança da verdadeira distribuição de probabilidade.

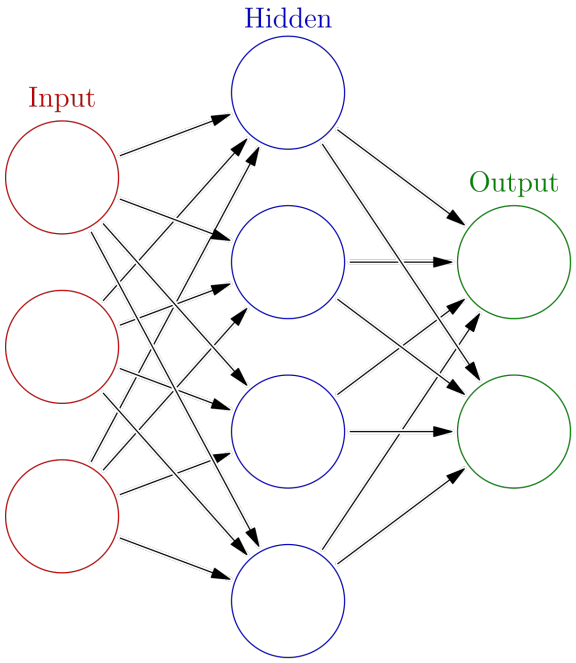
Figura 12 – O *perceptron* original, conforme proposto por ROSENBLATT (1958).



Fonte: YEHOSHUA (2023).

Para formar uma rede neural pró-alimentada, diversos *perceptrons* são conectados, resultando num grafo direcionado acíclico composto por múltiplas camadas interligadas, conforme a Figura 13. O objetivo de uma rede neural artificial é aproximar alguma função f^* desconhecida a partir de suas entradas e saídas conhecidas. Por exemplo, para um classificador, $y = f^*(\mathbf{x})$ mapeia uma entrada $\mathbf{x}$ a uma categoria y. A rede definirá portando um mapeamento $\mathbf{y} = f(\mathbf{x}, \theta)$ e aprenderá os parâmetros de cada *perceptron* que resultam na melhor aproximação de f^*.

Figura 13 – Rede neural artificial, composta por inúmeros *perceptrons*.



Fonte: WIKIPEDIA (2013).

Em resumo, a cada camada i é associada uma função $f^{(i)}$ de forma que a equação que descreve a rede neural pró-alimentada se torna $f(\mathbf{x}) = f^{(n)}(f^{(n-1)}...(f^{(2)}(f^{(1)}(\mathbf{x}))))$ para uma rede de n camadas. Essas estruturas encadeadas são as estruturas mais utilizadas de redes neurais. Introduzindo algumas terminologias importantes, n é chamada de profundidade do modelo (originando o termo aprendizado profundo), a camada final $f^{(n)}$ é a camada de saídas, a primeira camada $f^{(1)}$ é a camada de entradas e as camadas intermediárias são conhecidas como camadas ocultas, devido ao fato de que seus comportamentos não são diretamente observáveis ou determináveis durante as etapas de treinamento ou de validação.

Por fim, as funções de ativação g são famílias de funções escolhidas de antemão durante a arquitetura da rede neural e a etapa de treinamento da rede será responsável por determinar os parâmetros que especificam a função mais provável de aproximar f^* dentre todas as possibilidades pertencentes à família de possíveis funções de ativação. Para fins de generalização e correta modelagem de comportamentos não-lineares, é importante utilizar funções de ativação não-lineares.

2.4.2 Treinamento e retropropagação

Com a estrutura da rede definida, resta determinar os parâmetros $\mathbf{w}$ e $\mathbf{c}$. Em consequência das não-linearidades introduzidas nas redes neurais, calcular os parâmetros torna-se um problema de otimização não-convexo, o que significa que no espaço de configurações possíveis da rede neural existem múltiplos mínimos locais ao invés de apenas um mínimo global para a função de erro utilizada. A consequência dessa propriedade é que a solução encontrada durante a etapa de treinamento de um modelo não-convexo costuma ser um mínimo local, e não o mínimo global.

Desta forma, os algoritmos de otimização utilizados para treinar modelos de aprendizagem profunda são significativamente diferentes de algoritmos de otimização tradicionais. Em aplicações de aprendizagem de máquina, é definida uma medida de performance P que será otimizada de forma indireta. Para isso, é introduzida uma função de custo (ou perda) $J(\theta)$ na esperança de que minimizá-la irá melhorar melhorar a medida P (LECUN; BENGIO; HINTON, 2015). Tipicamente, a função de custo pode ser escrita como a média sobre um conjunto de treinamento, como:

$$J(\theta) = E_{(\mathbf{x},y)\hat{p}_{dados}}(L(f(\mathbf{x},\theta),y)), \tag{20}$$

onde L é a função de perda para cada exemplo e $\hat{p}_{dados}$ é a distribuição empírica dos dados. No caso supervisionado, y é a saída desejada.

A equação 20 define uma função-objetivo com respeito ao conjunto de treinamento, enquanto que o ideal é encontrar uma função-objetivo que seja minimizada sobre toda a distribuição geradora dos dados p_{dados} ao invés de somente sobre a distribuição $\hat{p}_{dados}$. Para isso, são utilizados os métodos de regularização descritos na seção 2.4.4.

Durante o treinamento de uma rede neural, o objetivo é que, dados os pares de exemplos (x, y), a rede neural otimize seus parâmetros a fim de minimizar a função de custo $J(\theta)$. Para isso, primeiro inicializam-se os pesos com valores arbitrários (podendo seguir alguma distribuição, como pesos ortogonais ou conforme alguma função de probabilidade) e depois é calculada a função de custo para lotes de pares (x, y).

Para atualizar os pesos durante a etapa de treinamento de uma rede neural, é utilizada a retropropagação. Na retropropagação, o objetivo é ajustar os pesos da rede neural usando o gradiente descendente estocástico, de forma a minimizar a função de perda. O processo funciona propagando o erro para trás, da camada da saída para as camadas de entrada, atualizando os parâmetros a cada passagem.

Supondo uma rede neural com L camadas, e que l representa a camada correspondente à iteração atual considerada durante o algoritmo de retropropagação, o primeiro passo é executar uma passagem para a frente (*foward-pass*) na rede (isto é, calcula-se as saídas de cada camada utilizando os pesos da iteração atual para uma entrada qualquer) e calcular a perda. Depois, o erro é calculado na camada de saída:

$$\delta^{(l=L)} = -(y - \hat{y}) \odot g'(z) \tag{21}$$

Onde y é o valor real, $\hat{y}$ é o valor previsto, g' é a derivada da função de ativação e $\odot$ representa a multiplicação elemento-por-elemento, também conhecida como o produto de Hadamard. Em seguida, o erro é propagado para trás, da camada l para a camada $l - 1$, com a seguinte fórmula:

$$\delta^{(l)} = ((\mathbf{W^{(l)}})^T \delta^{(l+1)}) \odot g'(z^{(l)}) \tag{22}$$

É importante destacar que os pesos $\mathbf{W^{(l)}}$ não são atualizados neste passo - seus valores apenas são utilizados para propagar o erro.

Finalmente, usando a regra da cadeia, as atualizações dos pesos da rede são calculadas como:

$$\frac{\partial \delta^{(l)}}{\partial \mathbf{W^{(l)}}} = \delta^{(l+1)} (g'^{(l)})^T \tag{23}$$

Nota-se aqui que a atualização do peso é proporcional ao erro, e ainda é modulada pela ativação da camada anterior. Os pesos são então atualizados de acordo com a taxa de aprendizado e o gradiente computado:

$$\mathbf{W^{(l+1)}} = \mathbf{W^{(l)}} - \epsilon \frac{\partial \delta^{(l)}}{\partial \mathbf{W^{(l)}}} \tag{24}$$

Este processo é repetido até que o erro total da rede seja minimizado. É importante lembrar ainda que, quando inserido no contexto de aprendizagem profunda, o método de retropropagação pode ser computacionalmente caro, sendo necessárias estratégias de otimização para lidar com problemas tais como o desaparecimento do gradiente.

2.4.3 Funções de ativação

Nas redes neurais modernas, é recomendada a utilização da função conhecida como unidade linear retificada, ou ReLU (FUKUSHIMA, 1975), definida como $\text{ReLU}(x) = \max(0, x)$. Essa função é não-linear porque não obedece a propriedade $\text{ReLU}(x) +$

$\text{ReLU}(y) = \text{ReLU}(x + y)$, o que permite a construção de curvas arbitrárias no espaço de características, além de ser muito fácil de ser computada do ponto de vista computacional.

No entanto, apesar das vantagens da ReLU, ela também apresenta um problema conhecido como "neurônios mortos", onde uma grande entrada negativa durante a etapa de treinamento fará com que o neurônio pare de aprender completamente. Esse problema ocorre porque a derivada da ReLU para entradas negativas é 0, o que significa que, durante a retropropagação, nenhuma atualização de peso ocorrerá. Uma vez que um neurônio chega nesse estado, é improvável que ele se recupere e comece a aprender novamente.

Dada essa limitação, foram propostas diversas variantes e alternativas à ReLU para tentar minimizar esse efeito e melhorar o desempenho das redes. Algumas dessas funções de ativação são:

Função de ativação sigmoide: definida como $\sigma(x) = \frac{1}{1+e^{-x}}$. Essa função mapeia qualquer valor de entrada para um intervalo entre 0 e 1, tornando-a útil para modelar probabilidades. Porém, ela apresenta o problema de "desaparecimento do gradiente", onde as atualizações de peso se tornam insignificantes durante a retropropagação, levando a um aprendizado mais lento ou até mesmo à paralisação do processo de aprendizado.

Função de ativação tangente hiperbólica (tanh): definida como $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Essa função, assim como a sigmoide, sofre do problema do desaparecimento do gradiente, balanceado em certa medida pelo fato de retornar valores negativos quando x é negativo, proporcionando uma medida de contraste melhor que a sigmoide.

Função de ativação linear: também conhecida como "identidade", retorna simplesmente o próprio valor $I(x) = x$. Utilizada geralmente na camada de saída da rede para transformações lineares.

ReLU com vazamento (Leaky ReLU): uma versão modificada de ReLU, onde há um pequeno valor de saída mesmo para entradas negativas, tentando assim resolver o problema dos neurônios mortos. Ela é definida como $\text{ReLU}_{\text{leaky}}(x) = \max(\alpha x, x)$, onde α costuma ser um valor baixo entre 0 e 0,2.

Softplus: uma das primeiras tentativas de suavizar a ReLU e reduzir o problema de gradientes desaparecendo, é definida como $\text{softplus}(x) = \ln(1 + e^x)$.

ELU (Exponential Linear Units): uma função de ativação que busca suavizar a ReLU, converge para um valor arbitrário de saturação conforme x tende a $-\infty$. Ela é

definida como:

$$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ a(e^x - 1), & \text{if } x \leq 0 \end{cases} \tag{25}$$

Onde a é um hiper-parâmetro que controla o valor de saturação para entradas negativas.

GELU: uma versão modificada de ReLU, que tenta suavizar o intervalo próximo a $x = 0$. Ela é definida como $\text{GELU}(x) = x\Phi(x)$, onde $\Phi(x)$ é a função de distribuição cumulativa da curva normal de média 0 e desvio padrão 1.

SWISH: outra versão modificada de ReLU, definida como $\text{SWISH}(x) = \frac{x}{1+e^{-x}}$. Essa função não é monótona, assumindo valores negativos quando x é muito pequeno e negativo. Na época de seu desenvolvimento, foi demonstrado que essa função apresentava desempenho superior à ReLU e à sigmoide para diversas arquiteturas e aplicações diferentes. (RAMACHANDRAN; ZOPH; LE, 2017)

MISH: Inspirada na SWISH, esta é, atualmente, uma das melhores funções de ativação em termos de performance para redes neurais. Definida como $\tanh \ln(1 + e^x)$, foi demonstrado ((MISRA, 2019)) que possui uma performance superior às funções ReLU, Leaky ReLU e SWISH para diversas aplicações diferentes.

Cada uma dessas funções tem suas vantagens e desvantagens específicas, e a escolha da função de ativação apropriada depende em grande parte do problema em questão e da natureza dos dados de entrada.

2.4.4 Métodos de regularização

Diversas abordagens de regularização são baseadas em limitar a capacidade dos modelos ao adicionar um termo de penalidade $\Omega(\theta)$ à função-objetivo J (GéRON, 2017). Isto é, a função-objetivo regularizada $\tilde{J}$ se torna:

$$\tilde{J}(\theta; \mathbf{X}, \mathbf{y}) = J(\theta; \mathbf{X}, \mathbf{y}) + \alpha\Omega(\theta) \tag{26}$$

onde $\alpha \in [0, \infty)$ é um hiperparâmetro que determina a contribuição do termo de penalidade. A interpretação do termo de regularização é que, além da função de perda convencional J, há também um termo que depende apenas dos parâmetros cujo objetivo é limitar os valores dos parâmetros. Por exemplo, se $\Omega(\theta)$ fosse definido como a soma do quadrado de todos os parâmetros, a rede neural tenderia a se configurar com parâmetros de valores mais baixos daqueles da configuração não regularizada. Diferentes escolhas do

termo Ω resultam em diferentes famílias de soluções sendo preferidas. É comum utilizar regularizadores em função apenas dos pesos $\mathbf{w}$ e não dos vieses $\mathbf{c}$, porque os vieses costumam convergir mais facilmente, além de que a regularização dos termos de viés pode introduzir um subajuste significativo. Alguns exemplos importantes de regularizadores incluem:

Regularizadores $\mathbf{L^2}$: também conhecida como regularização de Tikhonov ou regressão de cume, esse método de regularização tende a aproximar os pesos da origem com a adição do termo $\Omega(\theta) = \frac{1}{2}||\mathbf{w}||_2^2$ à função objetivo.

Regularizadores $\mathbf{L^1}$: de forma análoga à regularização L^2, esse método de regularização é descrito pelo termo $\Omega(\theta) = \frac{1}{2}||\mathbf{w}||_1$, isto é, a soma do valor absoluto dos pesos. Em comparação com a regularização L^2, a regularização L^1 resulta numa solução mais esparsa - haverão mais parâmetros cujo ponto ótimo é zero. Essa propriedade pode ser entendida como um mecanismo de seleção de características, de forma que algumas características dos dados são descartados pela rede neural.

Ampliação do conjunto de dados: é a prática de aumentar o número de exemplos de dados usados para treinamento, geralmente por meio de métodos como inversão, rotação ou corte dos dados existentes. A motivação desse método é que a melhor forma de aumentar o poder de generalização de um algoritmo de aprendizado de máquina é treiná-lo em mais dados. Assim, com a criação de dados através da transformação dos exemplos do conjunto de treinamento, é possível treinar a rede neural em um conjunto de treinamento muito maior sem a necessidade de captar mais dados.

Injeção de ruído: é adicionada uma perturbação aleatória $\epsilon_{\mathbf{W}} = \mathbb{N}(\epsilon; \mathbf{0}, \eta\mathbf{I})$ aos pesos da rede neural. Essa forma de regularização encoraja os parâmetros a serem otimizados para valores onde pequenas perturbações dos pesos têm uma influência pequena na saída da rede neural, ou seja, para uma região que não somente é um mínimo local, mas também possui baixa curvatura. É possível também injetar ruído nos dados durante o conjunto de treinamento, o que configuraria uma regularização de ampliação do conjunto de dados.

Parada antecipada: durante a etapa de treinamento de redes neurais, é comum observar que o erro de treinamento decai rapidamente ao longo do tempo, ao passo que o erro de generalização decai para um ponto mínimo para em seguida crescer lentamente. Portanto, é válido utilizar o modelo com o menor erro de generalização ao invés do último modelo obtido. Essa é a forma de regularização mais comumente empregada na

aprendizagem profunda, devido a sua simplicidade e fácil utilização.

Compartilhamento de parâmetros: às vezes, é útil forçar alguns conjuntos de parâmetros assumirem sempre o mesmo valor. Esse método é amplamente utilizado em redes neurais convolucionais aplicadas em visão computacional graças ao fato de que a classificação das imagens costuma ser invariante a translações e rotações. Isso permite que uma característica encontrada na coluna i de uma imagem possa também ser encontrada com facilidade na coluna $i + 1$ caso a imagem seja transladada.

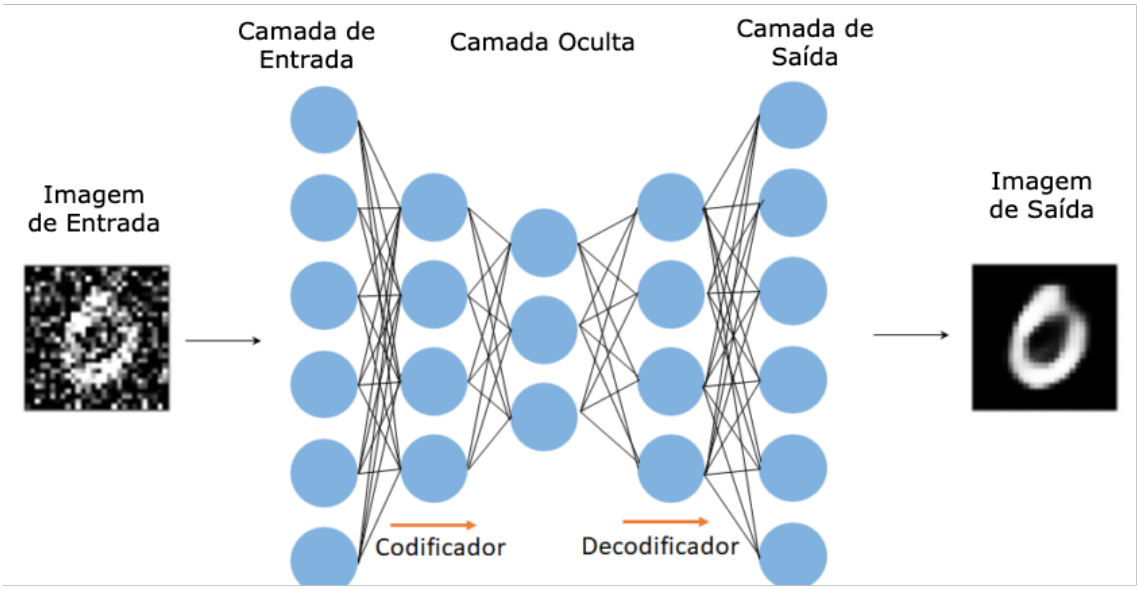
Agregação de bootstrap (ensacamento) e métodos de conjunto: em virtude da propriedade de que existem múltiplas diferentes configurações possíveis de redes neurais para um conjunto de dados, um método de regularização muito eficaz é treinar diferentes modelos e calcular a média da previsão dos modelos. O treinamento é realizado de forma que o conjunto de dados é reamostrado com substituição para k conjuntos. Isto é, cada um destes conjuntos possui a mesma quantidade de exemplos do conjunto original, mas os pontos de dados são amostrados seguindo uma distribuição uniforme de forma que seja possível haver pontos de dados duplicados nesses k conjuntos (por exemplo, para uma amostra $[1, 2, 3, 4, 5]$, uma possível reamostragem é $[2, 3, 3, 1, 4]$). Então, o modelo i é treinado no conjunto i e a previsão de todos os k modelos sobre algum ponto de dado é a média da previsão de cada modelo. Esse método é extremamente poderoso para reduzir o erro de generalização ao custo de mais poder computacional necessário, e costuma ser desencorajado em *benchmarks* de algoritmos em publicações científicas. As competições de aprendizado de máquina são comumente vencidas por modelos que utilizam essa técnica.

Descarte de neurônios (Dropout): Quando as redes neurais são muito grandes, o método de conjunto descrito acima não é muito prático. Uma maneira de replicar a técnica de ensacamento sem a necessidade de treinar múltiplas redes é o Dropout. Durante a etapa de treinamento, são aleatoriamente desconsiderados alguns neurônios da rede (onde a probabilidade desse descarte é um hiperparâmetro que tipicamente vale $0,5$ para camadas ocultas e $0,8$ para camadas de entrada). Ao final do treinamento, os pesos são multiplicados pelas probabilidades de descarte escolhidas a fim de corrigi-los para valores que corresponderiam ao mesmo modelo sem descarte. Esse método é computacionalmente barato e que funciona bem com quase qualquer modelo que pode ser otimizado com o método do gradiente descendente estocástico.

2.4.5 *Autoencoders*

Os *autoencoders* (LEARNING INTERNAL REPRESENTATIONS BY ERROR PROPAGATION, 1986) são uma classe especial de redes neurais artificiais que têm como finalidade aprender representações codificadas (ou compactas) dos dados de entrada, geralmente com o propósito de redução de dimensionalidade. Eles são tipicamente estruturados como uma rede simétrica em torno de uma camada central de menor dimensão (o *bottleneck*, ou gargalo), conforme a Figura 14.

Figura 14 – Arquitetura de um *autoencoder.*



Fonte: DSACADEMY (2022).

Os *autoencoders* são compostos por duas partes principais: o codificador (*encoder*) e o decodificador (*decoder*). O codificador comprime os dados de entrada em um espaço latente (ou espaço de codificação), uma representação de dimensão reduzida do conjunto de entrada, que captura as correlações nos dados. Já o decodificador reconstrói os dados a partir da representação do espaço latente. O objetivo é que a saída seja o mais próxima possível da entrada.

A função de perda de um *autoencoder* é frequentemente chamada de função de reconstrução de perda, penalizando a saída por ser diferente da entrada, como a perda do erro quadrático médio (MSE).

O *autoencoder* aprende a codificar os dados de entrada para um espaço latente e, em

seguida, decodificá-los, ajustando-se iterativamente por meio do treinamento com dados de entrada e minimizando a perda de reconstrução. O que se aprende é uma representação compacta dos dados e uma maneira de transformar os dados para essa representação.

A principal utilidade do espaço latente é que, embora os dados de entrada possam ser de alta dimensão, eles residem em um espaço de dimensão inferior. Assim, a rede pode aprender um mapeamento dos dados de entrada para este espaço latente de menor dimensão. A representação do espaço latente captura as propriedades essenciais dos dados de entrada de uma forma mais compacta. Isso é muito importante para tarefas como redução de dimensionalidade, geração de novos dados, ou até mesmo para treinar modelos mais complexos de aprendizado de máquina.

Os *autoencoders* têm uma variedade de aplicações, incluindo redução de dimensionalidade, remoção de ruídos, e geração de novos dados.

Redução de Dimensionalidade: Para grandes conjuntos de dados multi-dimensionais, os *autoencoders* podem compactar os dados em um espaço latente com dimensões reduzidas, de uma maneira que preserve as características principais ou a estrutura dos dados. Este é um benefício particularmente útil para visualização de dados multidimensionais.

Remoção de Ruído: *Autoencoders* podem ser úteis para melhorar a qualidade dos dados ruidosos. Durante o treinamento, os *autoencoders* recebem dados ruidosos como entrada e dados limpos como saída esperada. Depois de treinado, o *autoencoder* pode receber dados ruidosos e gerar remover o ruído.

Geração de Dados: *Autoencoders* podem ser usados para gerar novos dados que se assemelhem aos dados nos quais foram treinados. Ao passar valores aleatórios da camada do espaço latente para o decodificador, podem ser gerados novos dados.

Os *autoencoders*, no entanto, também têm suas desvantagens. Treinar um *autoencoder* que não perde informações relevantes e ao mesmo tempo mantém a representação compacta é um desafio. O treinamento pode ser lento e não é garantido que o espaço latente codificado seja contínuo e bem formado ou que as reconstruções sejam de alta qualidade. Além disso, eles podem ser usados apenas nos tipos de dados nos quais foram treinados, e a saída pode ser distorcida se for dada uma entrada muito diferente dos dados de treinamento.

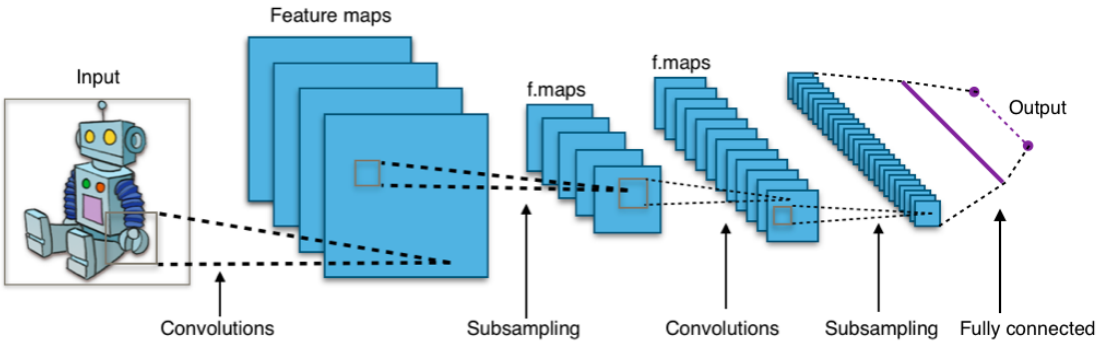
Além disso, também existe o problema na escolha da dimensão do espaço latente. Uma escolha incorreta pode levar a um *autoencoder* que simplesmente aprende a cópia da

identidade (ou seja, um *autoencoder* com uma representação latente muito grande que não comprime suficientemente os dados) ou um que não consegue aprender uma representação útil (ou seja, um *autoencoder* com uma representação latente muito pequena que perde muita informação importante durante a compressão).

2.4.6 Redes neurais convolucionais

As Redes Neurais Convolucionais (Convolutional Neural Networks - CNNs) (SCHMIDHUBER, 2015) são uma classe de redes neurais profundas que se tornaram a referência para muitos problemas de visão computacional. Elas foram projetadas para processar dados com uma topologia de grade, como uma imagem, que é uma grade de pixels, mas podem ser adaptadas para resolver problemas com dados unidimensionais, como sinais analógicos.

Figura 15 – Arquitetura de uma rede neural convolucional.



Fonte: WIKIPEDIA (2018).

A arquitetura de uma CNN, conforme a Figura 15, geralmente consiste em uma pilha alternada das seguintes camadas (GOODFELLOW; BENGIO; COURVILLE, 2016):

Camada convolucional: Esta camada utiliza um conjunto de filtros aprendíveis. Cada filtro é convolvido com a entrada para produzir um mapa de características. A principal vantagem da convolução é que, permitindo que cada filtro funcione como um detector de características, ela permite que a rede aprenda a analisar partes localizadas da entrada.

Dessa forma, um único filtro pode detectar padrões em diferentes regiões da entrada, o que permite que, ao longo das camadas, a rede convolucional aprenda uma hierarquia de características. As camadas iniciais costumam detectar linhas, enquanto que as camadas finais detectam padrões com um alto nível de complexidade.

Camada não-linear (ReLU): Esta camada aplica uma função de ativação não-linear, tipicamente a ReLU, para cada elemento do mapa de características, adicionando não-linearidades necessárias para a rede aprender funções complexas.

Camada de agrupamento (*Pooling*): Esta camada reduz as dimensões espaciais (largura e altura) dos mapas de características. A operação mais comum usada para realizar o agrupamento é o *max pooling*, o qual pega lotes da entrada e mantém somente o valor máximo.

Camada completamente conectado: Esta camada tem conexão com todos os neurônios na camada anterior, da mesma forma que numa rede neural artificial tradicional. É geralmente usada no final da rede para criar a saída final.

Os dados passam por estas camadas e vão sendo transformados e processados até produzirem a saída. Durante o treinamento, a rede ajusta os pesos dos filtros e outros parâmetros para minimizar a função de perda. Os pesos dos filtros convolucionais permitem que a rede seja robusta a variações espaciais e também ajuda a reduzir a quantidade de parâmetros, tornando a rede mais eficiente e mais fácil de treinar.

As CNNs têm sido extremamente eficazes em tarefas que envolvem análise de imagens, como reconhecimento de imagens, detecção de objetos, reconhecimento facial, entre outros. Elas também são utilizadas em processamento de linguagem natural e modelagem de séries temporais. Além disso, CNNs são a base para muitos avanços recentes na área de inteligência artificial, incluindo carros autônomos e sistemas de reconhecimento de fala.

No entanto, CNNs também têm suas desvantagens. Elas exigem grandes quantidades de dados e poder de processamento para treinamento, especialmente para problemas complexos. E, apesar do agrupamento ajudar a fornecer alguma translacional invariância, CNNs ainda não lidam bem com rotações ou redimensionamentos de imagens. Além disso, interpretar as camadas internas de uma CNN profunda pode ser difícil, embora existam várias técnicas recentes que tentam abordar essa questão.

2.5 Super-Resolução de Áudio

A super-resolução de áudio é uma técnica de processamento de sinal que procura melhorar a qualidade de um sinal de áudio $x[n]$. Este problema envolve a tarefa de adicionar pontos no sinal, artificialmente aumentando sua taxa de amostragem, como tentativa de

estimar as frequências maiores do sinal e aumentar sua qualidade percebida.

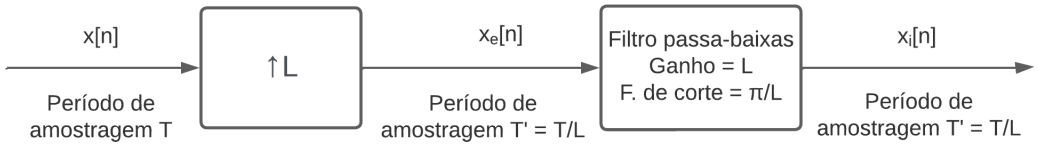
Isso é geralmente alcançado por meio de algoritmos que adicionam detalhes de alta frequência ao sinal de áudio. Porém, desenvolver tais algoritmos é um problema muito complexo. Por ser uma tarefa de inferência (os pontos adicionados ao sinal são meras estimativas), torna-se difícil impedir que o algoritmo adicione ruídos e dados incorretos no sinal.

2.5.1 Super-resolução clássica

Será utilizado o termo super-resolução clássica, ou *upsampling*, para qualquer técnica de ampliação de largura de banda de um sinal de áudio que não faça uso de redes neurais. Nestes casos, utilizam-se métodos de interpolação a partir da teoria de sinais discretos (OPPENHEIM; SCHAFER; BUCK, 1999).

Os métodos interpolativos são chamados desta maneira porque desempenham o papel de adicionar $L - 1$ amostras entre duas amostras consecutivas do sinal original. Isto é, buscam atribuir pontos intermediários para duas amostras consecutivas de uma função qualquer. Existem diversas maneiras diferentes de se realizar este procedimento, cada qual com diferentes características. A mais simples é a interpolação pelo valor da amostra vizinha mais próxima, enquanto que a mais utilizada é a interpolação por *spline* bicúbica. Após qualquer método de interpolação utilizado, é necessário passar o sinal por um filtro passa-baixas de forma a não introduzir artefatos e distorções provenientes de frequências maiores geradas pela interpolação.

Figura 16 – Sistema de ampliação de largura de banda para um sinal $x[n]$.



Fonte: Autoria própria.

A Figura 16 ilustra o processo de ampliação de largura de banda. O primeiro bloco representa a etapa de ampliação de frequência de amostragem de f para fL, e o segundo bloco representa o filtro passa-baixas, que desempenha a função de *anti-aliasing*.

Na interpolação linear, são adicionados $L-1$ pontos entre n e $n+L$. Para um ponto $n+a$ entre n e $n+L$, o valor inserido é dado pela equação $x[n+a] = \frac{L-a}{L}x[n] + \frac{a}{L}x[n+L]$. Esse método costuma não ser muito bom se a frequência de amostragem do sinal original for próxima a sua taxa de Nyquist, porque haverá energia considerável na banda $\frac{\pi}{L} < |\omega| \leq \pi$. Portanto, este método se torna adequado e suficiente para os casos em que o sinal é superamostrado.

Já a interpolação *spline* bicúbica busca encontrar polinômios interpolantes de terceiro grau de forma que, para cada duas amostras do sinal original $x[n]$, há continuidade até a segunda derivada dos polinômios interpolantes destas amostras. Esta técnica costuma apresentar maior suavidade, sendo preferível em relação à interpolação linear na maioria dos casos.

Por fim, a interpolação *sinc*, também conhecida como interpolação de Whittaker-Shannon, é a aplicação do filtro passa-baixas ideal de frequência de corte $f_c = f_s$, onde f_s é a frequência de amostragem do sinal. Sua resposta ao impulso é a função $h_{sinc}[n] = \text{sinc}[\frac{n}{f_c}]$. Entretanto, na prática este filtro não é realizável porque não é um sistema causal e precisa de um número infinito de amostras. Diversos filtros foram criados para aproximar o filtro passa-baixas ideal, em particular o filtro de Lanczos, que pode ser descrito por:

$$L(x) = \begin{cases} \text{sinc}(\pi x)\text{sinc}(\frac{\pi x}{a}), & -a < x < a, \\ 0, & \text{caso contrário} \end{cases} \tag{27}$$

em que a é o parâmetro inteiro que determina o tamanho da janela de Lanczos.

2.5.2 Super-resolução através de Redes Neurais

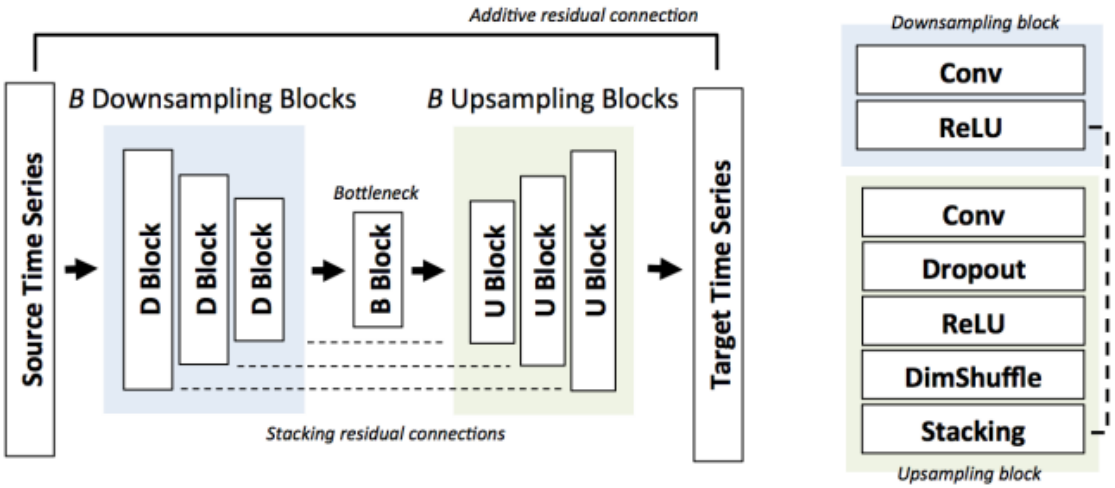
Para solucionar o problema da super-resolução por meio de redes neurais, LI; LEE (2015) foi o pioneiro e utilizou uma arquitetura de rede neural simples e completamente conectada que tinha como entrada e saída a potência espectral de janelas do áudio, posteriormente invertidas com uma ISTFT (Transformada de Fourier de Tempo Curto Inversa). Esse método obteve uma performance melhor que as técnicas de super-resolução anteriormente conhecidas, e abriu as portas para a utilização de redes neurais para a super-resolução de áudio.

Foi desenvolvida por KULESHOV; ENAM; ERMON (2017) uma arquitetura que dispensava a utilização de características do domínio da frequência: a Audio U-Net. Essa

arquitetura é bem-sucedida na tarefa de super-resolução de áudio. Inspirada nos *autoencoders*, a Audio U-Net também é composta por blocos de redução de dimensão (codificador) e blocos de ampliação de dimensão (decodificador). Porém, diferentemente de um *autoencoder*, essa arquitetura não utiliza camadas densas e, sim, filtros convolucionais.

A cada camada do decodificador, são aplicados múltiplos filtros de *stride* (passo) igual a 2 a fim de reduzir a dimensão num fator de 2. Cada um desses filtros é responsável por um canal da camada seguinte, de forma que cada filtro é aprende uma característica da camada. A Figura 17 fornece uma representação visual dessa arquitetura.

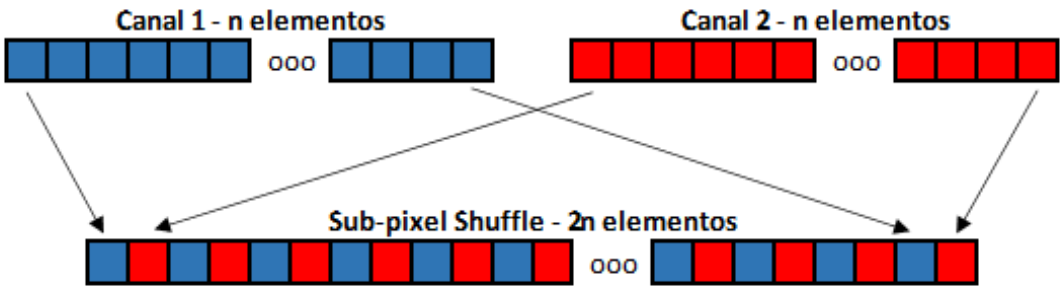
Figura 17 – Arquitetutra da Audio U-Net.



Fonte: KULESHOV; ENAM; ERMON (2017).

No decodificador, a ampliação de dimensionalidade é feita através de uma etapa chamada de *dimensional shuffle* (embaralhamento dimensional), também conhecida como *subpixel shuffling* (embaralhamento sub-pixel) (SHI *et al.*, 2016). Essa camada foi criada para resolver o problema da super-resolução de imagens, e foi adaptada para o caso unidimensional na super-resolução de áudio. O *subpixel shuffling* concatena de forma alternada os elementos da camada anterior, reduzindo o número de canais pela metade, conforme a Figura 18.

Figura 18 – *Subpixel shuffling*.



Fonte: Autoria própria.

3 METODOLOGIA

Neste trabalho, são comparadas diferentes metodologias de super-resolução de áudio. Duas delas, mais simples, atuam como *baselines* de comparação: a interpolação linear e a interpolação *spline*. Outras duas, mais sofisticadas, fazem uso de algoritmos de aprendizado profundo: uma arquitetura denominada *autoencoder* (decodificador e codificador encadeados) e uma arquitetura convolucional inspirada em *autoencoders* conforme KULESHOV; ENAM; ERMON (2017).

O problema da super-resolução pode ser imaginado como um problema de geração: a partir de um áudio de frequência de amostragem f_{LR}, deseja-se gerar um áudio de frequência de amostragem rf_{HR}, onde r é o fator de ampliação. São investigados os fatores de ampliação 2, 4 e 6 para uma frequência-alvo de $f_{HR} = 48$ kHz.

Como o ouvido do ser humano é capaz de captar ondas sonoras de até aproximadamente 20 kHz, e conforme o teorema de amostragem de Nyquist, a frequência de amostragem mínima necessária para representar tal sinal é 40 kHz. Num exemplo onde $r = 4$, para cada segundo de áudio é necessário mapear um vetor de 10 mil dimensões noutro de 40 mil dimensões. As únicas técnicas de aprendizado de máquina capazes de resolver problemas com um alto número de dimensões são as de aprendizado profundo, onde a adição de dimensões é trivial - basta aumentar o número de neurônios de entrada da rede.

3.1 Materiais utilizados

Devido ao uso de técnicas de aprendizado profundo, é necessário treinar os modelos com um grande volume de dados. Há alguns tipos diferentes de sinais de áudio: musicais, de fala, ambientais, ruído, etc. Para que a rede neural aprenda com maior rapidez e a fim de limitar o escopo de sua utilização (permitindo um aprendizado mais direcionado

e inferências mais precisas), é importante selecionar apenas um tipo de sinal de áudio. Neste projeto, optou-se pelo uso de sinais de voz por conta de sua relevância no campo da comunicação.

Foi utilizada a base de dados CSTR VCTK Corpus (YAMAGISHI JUNICHI; VEAUX, 2019), composta por gravações de fala de 110 falantes da língua inglesa com diversos sotaques. Cada um desses falantes leu cerca de 400 passagens de diversos jornais, selecionadas de forma a maximizar a cobertura fonética, totalizando 45 mil arquivos .wav e 44 horas de gravação. Todas as amostras têm uma frequência de amostragem de 48 kHz e uma profundidade de bits de 16.

Esse banco de dados se tornou um *benchmark* do estado da arte para a tarefa de super-resolução de sinais de voz, de forma que os resultados encontrados no presente trabalho poderão ser comparados facilmente com os da literatura através das métricas posteriormente descritas. Foram descartados os falantes *p280* e *p315* da base devido a problemas técnicos.

Para o desenvolvimento dos algoritmos, bem como o pré-processamento, a análise dos sinais e a obtenção dos resultados, foi utilizada a plataforma Google Colaboratory. Essa plataforma atua como um interpretador da linguagem *Python* e permite a utilização de placas de vídeo de alta performance, como a A100 (memória de 80GB e velocidade de 312 TFLOPS) e a V100 (memória de 32 GB e velocidade de 130 TFLOPS). As principais bibliotecas da linguagem *Python* utilizadas são descritas na tabela 1 abaixo. A inferência dos modelos obtidos foi testada no processador Intel Core i5-1035G1.

Biblioteca	**Versão**	**Descrição**
keras	2.12.0	API que facilita o desenvolvimento de redes neurais.
tensorflow	2.12.0	Framework de aprendizado de máquina de código aberto.
librosa	0.10.1	Módulo para processamento de áudio.
scipy	1.10.1	Algoritmos fundamentais para computação científica em Python.
h5py	3.9.0	Permite a leitura e escrita de arquivos HDF5.

Tabela 1 – Bibliotecas *Python* utilizadas e suas respectivas versões.

Por fim, houve a necessidade de se adquirir sinais de áudio para amostragens adicionais e demonstrações do projeto. Foram escolhidos os microfones integrados Realtek presentes no laptop Acer Aspire F5-573G, devido à simplicidade e fácil acesso. Além

disso, arquivos de música públicos também serão utilizados a fim de testar a validade do algoritmo para amostras significativamente diferentes daquelas usadas durante o treino.

3.2 Análise estatística da base de dados

O primeiro passo, antes de modificar qualquer elemento da base de dados, é realizar uma análise estatística. A base de dados escolhida é composta por arquivos de voz, com suas respectivas transcrições em arquivos de texto. Assim, estudou-se as diferentes distribuições para diversas características sonoras (como tempo de silêncio, amplitude máxima do sinal, tom fundamental médio do sinal) e textuais (como palavras e fonemas pronunciados).

O objetivo dessa análise é buscar por *outliers* que fujam da distribuição esperada da base de dados e avaliar a necessidade de transformar os dados de alguma maneira. Esses *outliers* foram removidos das etapas de treinamento e validação, de forma a não interferir no desempenho das redes neurais, e as transformações julgadas como necessárias foram aplicadas.

3.3 Pré-processamento e extração de características

Por conta dos resultados obtidos na análise estatística da base, não foi necessário um pré-processamento extensivo. Além disso, para que os modelos possuam uma rápida inferência, o uso de outras características (como a Transformada de Fourier de Curto Tempo) é indesejável.

Os arquivos de áudio foram divididos em diversos segmentos de tamanho 512 (equivalentes a 10,6 ms) selecionados de forma que o n-ésimo segmento contenha os elementos $[128n, 128n + 512)$. Foi efetuada a decimação para cada fator de subamostragem r após a aplicação de um filtro de Chebyshev tipo I de ordem 8. Assim, a partir de cada segmento y_i são gerados três correspondentes de menor resolução: $x_i^{r=2}$, $x_i^{r=4}$, $x_i^{r=6}$.

O racional da aplicação do filtro se deve ao fato de que, na prática, sinais de áudio decimados incluem distorções e artefatos resultantes das altas frequências. Para reduzir esse efeito e melhor simular a captura ou transmissão de um sinal de baixa resolução, é aplicado ao sinal um filtro antes da decimação para remover as frequências acima da frequência de Nyquist do sinal decimado correspondente. Entretanto, filtros ideais não

são realizáveis, de forma que o sinal resultante ainda possui alguns artefatos oriundos de componentes das altas frequências.

Finalmente, por conta das arquiteturas escolhidas das redes neurais, os tensores X da entrada devem ter o mesmo tamanho e o mesmo formato que os tensores Y da saída. A solução proposta é aplicar uma interpolação *spline* aos vetores x_i^r que os leva de volta à resolução original (a mesma de y_i). Portanto, as redes neurais aprenderão a adicionar as frequências mais altas à super-resolução obtida pelo método de interpolação *spline*.

Para acelerar o carregamento dos dados, os arquivos de áudio foram processados e convertidos em arquivos .h5 (um formado de dado hierárquico) para cada valor de r. Cada arquivo .h5 é composto por 50 gravações escolhidas aleatoriamente sem repetição, e armazena os dados já em pares de vetores (x, y) de tamanho 512. Isso resultou em arquivos .h5 contendo, cada um, aproximadamente 1 GB de dados, compostos por entre 60 mil e 75 mil pares de vetores.

3.4 Arquiteturas escolhidas

Foram utilizadas duas diferentes arquiteturas de rede neural: o *Autoencoder* e uma arquitetura baseada na Audio U-Net, desenvolvida por KULESHOV; ENAM; ERMON (2017).

3.4.1 Autoencoder

Autoencoders são redes neurais cuja entrada e saída possuem o mesmo tamanho e que podem ser divididas em três partes principais: o codificador, a camada oculta (também conhecida como *bottleneck*) e o decodificador. Tipicamente, o objetivo dessas redes neurais é a remoção de ruído.

Nas camadas do codificador, as dimensões da rede neural são reduzidas, permitindo que a rede aprenda uma hierarquia de características dos dados. No decodificador, o processo inverso é realizado. A camada oculta fornece uma representação, conhecida como espaço latente, de baixa dimensão da entrada, e essa representação em geral não inclui o ruído da entrada. Além disso, é útil conectar as camadas de mesmo tamanho entre si, de forma que a rede aprenda apenas $x - y$.

Para essa arquitetura, foram testados 2 tamanhos diferentes: uma rede com camadas de tamanho $[512, 256, 128, 64, 32, 16, 8, 4, 2, 4, 8, 16, 32, 64, 128, 256, 512]$ que será deno-

tada AE$_{large}$, totalizando 525.802 parâmetros, e outra rede com camadas de tamanho $[512, 128, 32, 8, 2, 8, 32, 128, 512]$, totalizando 210.546 parâmetros, denotada AE$_{small}$. Para ambas as redes, as camadas do codificador foram conectadas às camadas do decodificador. Nas camadas do decodificador, foi aplicada a técnica de descarte de neurônios (*Dropout*) com taxa de descarte de 0,5.

3.4.2 Audio U-Net

Para a arquitetura Audio U-Net, foram testados 2 tamanhos diferentes. A primeira variação, denotada U-Net$_{small}$, possui 7 blocos para o codificador e 7 blocos para o decodificador. Para a b-ésima ($b \in [1, 7]$) camada do codificador, há $\min(16b, 32)$ filtros de tamanhos $\max(\frac{16}{b}, 4)$. A segunda variação, denotada U-Net$_{large}$, também conta com 7 blocos codificadores e 7 decodificadores. A b-ésima camada do codificador conta com $\min(128 \cdot 3^{b-1}, 512)$ filtros de tamanhos $\max(2^{7-b} + 1, 9)$.

Para ambas as variações, o decodificador aplica as mesmas camadas do codificador em ordem inversa, com a diferença de que há duas vezes mais filtros por bloco e as *strides* das convoluções valem 1. Ainda, é utilizada a técnica de descarte de neurônios (*Dropout*) com taxa de descarte de 0,5 apenas para o decodificador. A U-Net$_{small}$ possui 148.850 parâmetros, enquanto que a U-Net$_{large}$ possui 94.573.186 parâmetros.

3.5 Métricas de desempenho

Para o treinamento das redes neurais e posterior validação de resultados, é necessário fazer uso de métricas quantitativas. Em análise de sinal, as medidas mais utilizadas são SNR (Relação sinal-ruído) e LSD (Distância logarítmica do espectro) definidas, respectivamente, como:

$$SNR(x, y) = 10 \log_{10} \frac{\sum_{i=1}^{N} x_i^2}{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{28}$$

$$LSD(x, y) = \frac{1}{L} \sum_{\ell=1}^{L} \sqrt{\frac{1}{K} \sum_{k=1}^{K} (X_x(\ell, k) - X_y(\ell, k))^2} \tag{29}$$

Aqui, x e y são os sinais originais e previstos, respectivamente, com número de amostras N. Além disso, $X(\ell, k) = \log |S(\ell, k)|^2$ é a potência logarítmica espectral, $S(\ell, k)$ é a Transformada de Fourier de Curto Termo (STFT) do Sinal e ℓ e k são o índice das jane-

las temporais e índice de frequência, respectivamente. L é o número de janelas temporais do sinal (definido como a razão entre o número de amostras do sinal e o comprimento de janela) e K é o número de índices de frequência. Foram utilizadas janelas temporais de comprimento 512, como é típico na aplicação da STFT para sinais de voz.

Tipicamente, para sinais de voz, a LSD é uma métrica mais correlacionada com a qualidade de áudio percebida. Entretanto, como essa métrica faz uso da STFT, não é útil observá-la durante a fase de validação, já que os vetores nessa etapa contam com apenas 512 elementos. A LSD foi, portanto, utilizada como a principal métrica de desempenho após o treinamento completo das redes e aplicada somente aos sinais de voz do conjunto de testes.

Para as redes neurais, uma métrica de desempenho muito importante é o tempo de inferência, definido como o tempo médio necessário para aplicar a rede num vetor qualquer x. No caso da aplicação da super-resolução em telecomunicações, o tempo de inferência pode ser interpretado com o atraso que um sinal sofre para ter sua resolução ampliada.

3.6 Treinamento das redes neurais

Durante a etapa de treinamento, a base de dados foi dividida em duas: 100 dos 108 falantes compõem o conjunto de treinamento e os restantes (*p376*, *p374*, *p364*, *p363*, *p362*, *p361*, *p360*, *p351*) compõem o conjunto de teste. O objetivo dessa separação é permitir a avaliação do poder de generalização do modelo de aprendizado de máquina. O conjunto de treinamento é alimentado ao modelo, cujo performance é medido utilizando dados do conjunto de teste após o processo de treinamento.

Essa etapa é importante também para a otimização dos hiper-parâmetros (aqueles que não são aprendidos pelo modelo). Para otimizá-los, não foi utilizada toda a base de dados, mas sim apenas 150 arquivos de áudio (3 arquivos .h5) escolhidos aleatoriamente, totalizando em média 195 mil pares de vetores. Foram destinados ao conjunto de validação 5% desses pares de vetores, o qual atua de forma a auxiliar a determinação do poder de generalização da rede neural.

Na fase de otimização de hiper-parâmetros, as redes foram treinadas apenas com $r = 4$. O tamanho de lote utilizado foi de 32, e a rede foi treinada durante 20 épocas. Optou-se por uma taxa de aprendizado inicial de 10^{-4}, com uma taxa de decaimento exponencial de 0,96 a cada 5.000 passos. Ou seja, a cada 5.000 lotes alimentados a uma rede, a taxa de

aprendizado é multiplicada por 0,96. Isso permite uma convergência mais acelerada no início do treinamento, além de reduzir o sobre-ajuste conforme o treinamento progride. Ao final do treinamento, os parâmetros do modelo com o menor erro de validação são salvos, bem como seus hiper-parâmetros e erro de validação resultante.

Para a função de perda, foram avaliadas tanto a função MSE (ou norma L_2) como a função MAE (norma L_1) para determinar qual é mais apropriada para o problema. O otimizador escolhido foi o Adam (método SGD baseado na estimação adaptativa de momentos de primeira e segunda ordem) com $\beta_1 = 0,9$ e $\beta_2 = 0,999$. Foram também investigadas diferentes funções de ativação: a ReLU, *Leaky* ReLU (com $\alpha = 0,2$), SWISH ($f(x) = x\sigma(x)$) e MISH ($g(x) = x\tanh(\text{softplus}(x))$).

Ao fim da etapa de otimização de hiper-parâmetros, foram escolhidos os hiper-parâmetros com melhor desempenho observado (leia-se: maior SNR médio do conjunto de validação) e cada uma das 4 redes (AE_{small}, AE_{large}, U-NET$_{small}$, U-NET$_{large}$) foi treinada com os mesmos hiper-parâmetros para $r = 2$, $r = 4$ e $r = 6$. Nessa etapa final, as redes foram treinadas em 60 arquivos .h5 (3,9 milhões de pares de vetores de tamanho 512) com 30 épocas. Foi utilizada, também, a parada antecipada com paciência de 3 épocas para impedir que o sobreajuste.

3.7 Validação e obtenção dos resultados

Uma vez que todas as redes neurais foram treinadas com uma grande parcela do conjunto de dados, foram comparados seus resultados no conjunto de testes pelas métricas de despenho. Além disso, os métodos de interpolação linear e *spline* bicúbica servirão como referências para as comparações.

Para fins de visualização e demonstração, três arquivos de áudio do conjunto de testes foram selecionados. Foram comparadas, qualitativamente, o espectro da frequência dos sinais originais, dos amostrados e dos reconstruídos pela rede neural de melhor desempenho para os diferentes valores de r. É esperado que os espectros revelem que as altas frequências são desprezíveis para os sinais decimados, e que as redes neurais conseguem reconstruir essas frequências com maior fidelidade que os métodos interpolativos.

4 IMPLEMENTAÇÃO E RESULTADOS OBTIDOS

Neste capítulo, são explorados a implementação e os resultados obtidos das análises e procedimentos propostos no capítulo da Metodologia. Primeiro será abordado a análise proposta sobre as bases de dados. Em seguida, são comparados o desempenho dos modelos sobre a base aleatorizada. Depois disso, a técnica que resultou no modelo de melhor desempenho é utilizada para encontrar a topologia mais adequada, treinada sobre a base de dados não aleatorizada. Em seguida, é possível comparar os dois modelos e obtidos. Por fim, é feita uma demonstração sobre um sistema simples que utiliza a interface proposta.

4.1 Análise estatística da base de dados

A base de dados CSTR VCTK Corpus compreende arquivos .wav e arquivos .txt para múltiplos falantes da língua inglesa. Os arquivos .wav são as gravações de fala capturados a uma taxa de amostragem de 96 kHz e reamostrados a 48 kHz, enquanto que os arquivos .txt contêm as palavras pronunciadas em cada um dos arquivos de áudio presentes na base de dados. Com base nas informações presentes, é possível realizar uma análise estatística tanto do áudio e suas características como das palavras e fonemas pronunciados.

4.1.1 Análise dos arquivos de texto

Apesar de não serem utilizados durante o treinamento das redes neurais, uma análise do conteúdo das gravações pode explicar o comportamento das redes já treinadas. Em particular, é válido fazer a suposição inicial de que, pelo fato de a base de dados conter apenas falas em inglês, as redes neurais apresentarão uma performance inferior quando aplicadas a áudios de línguas que não sejam da língua inglesa.

Os arquivos de texto incluem as frases pronunciadas em cada arquivo de áudio. Por

exemplo, o arquivo p264_060.txt (correspondente ao áudio p264_060.wav) contém a frase “What kind of man does that, Mr Dick?)”, que, em português, significa “Que tipo de homem faz isso, Sr Dick?)”. De imediato, os arquivos de texto contêm algumas informações irrelevantes:

- Ao final de cada arquivo de texto, há um caractere “)” (fecha parênteses);
- Os arquivos de texto contêm pontuação;
- Alguns arquivos de texto contêm abreviações que não correspondem à pronúncia (no exemplo anterior, “Mr Dick” é na verdade pronunciado como “Mister Dick”);
- As palavras são capitalizadas.

Dado que o objetivo desta análise é entender o conteúdo presente em cada gravação, os arquivos de texto foram pré-processados. Foram removidas todas as pontuações, as abreviações foram desfeitas e todos os caracteres foram convertidos para caixa baixa. Assim, o exemplo do arquivo p264_060.txt foi transformado para “what kind of man does that mister dick”.

Essas transformações permitem analisar a frequência e distribuição de letras e palavras nas gravações. Porém, a escrita gráfica não costuma representar os sons pronunciados, especialmente na língua inglesa, cujas letras não possuem uma pronúncia única para diferentes palavras que as contêm. Para lidar com esse problema, os textos foram convertidos para o IPA (alfabeto fonético internacional), que representa com maior fidelidade as pronúncias de cada palavra.

Na base de dados, foram pronunciadas 5.823 palavras únicas em um total de 325.796 vezes. Dessas, 611 palavras únicas (10,5%) foram pronunciadas apenas uma vez, e as três palavras mais frequentes foram “the” (21.183 vezes), “a” (11.138 vezes) e “is” (9.247 vezes). A tabela 2 apresenta a distribuição de repetições de palavras, e a Figura 19 apresenta as 60 palavras mais frequentes na base de dados.

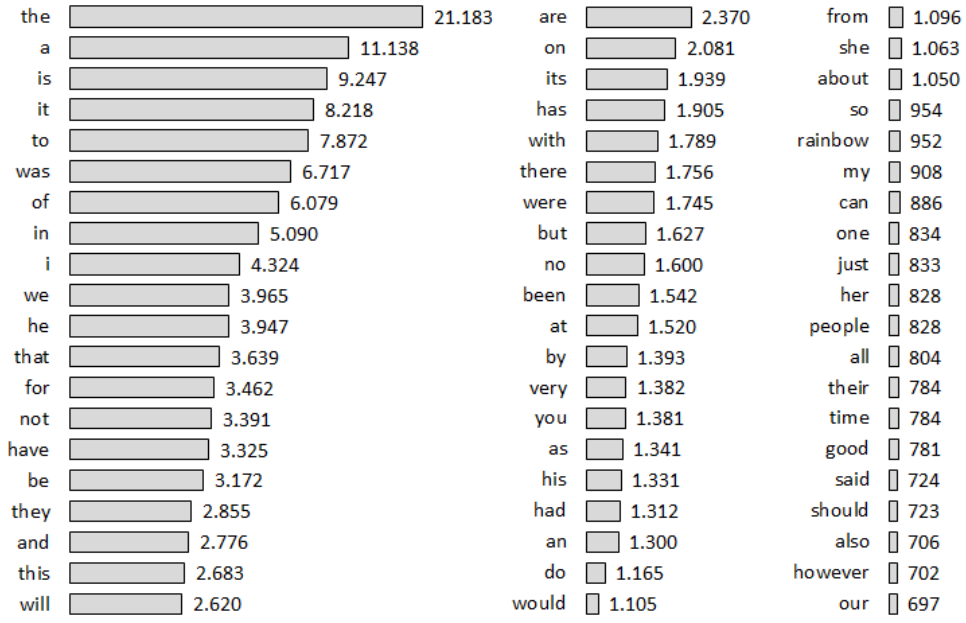
Tabela 2 – Distribuição de palavras por número de repetições.

Número de repetições	**Palavras Únicas**	**Frequência (%)**
Abaixo de 5	2.130	36,58%
De 5 a 9	996	17,10%
De 10 a 19	982	16,86%
De 20 a 49	900	15,46%
De 50 a 99	335	5,75%
De 100 a 999	437	7,5%
De 1.000 a 9.999	41	0,70%
Acima de 10.000	2	0,03%

Fonte - Autoria própria.

A média do número de repetições por palavra foi de 56, com desvio padrão de 436. Conforme a Figura 19, nota-se que as palavras mais frequentes costumam ser curtas e conectoras. Dentre as 60 palavras mais repetidas, apenas três delas são substantivos: “rainbow”, “people” e “time”. Destaca-se que a palavra “rainbow” foi repetida 952 vezes porque os participantes foram convidados a ler a Passagem do Arco-íris, um parágrafo de domínio público frequentemente utilizada para análises de discurso.

Figura 19 – As 60 palavras mais frequentemente pronunciadas da base de dados e suas respectivas frequências.

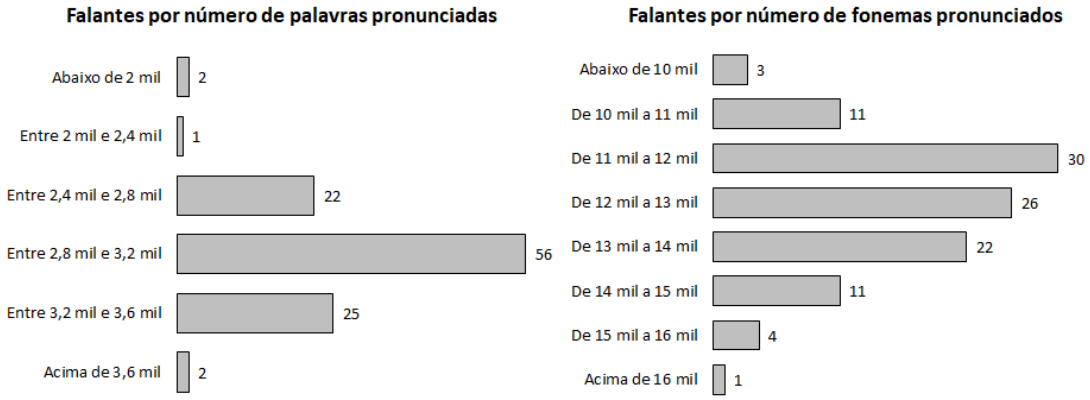


Fonte - Autoria própria.

Cada locutor pronunciou, em média, 3.017 palavras ao longo de todas as suas gravações, com um desvio padrão de 310 palavras. Já a média de fonemas pronunciados por cada locutor foi de 12.450, com um desvio padrão de 1.488 fonemas. A amplitude de ambas as distribuições é baixa: o locutor menos frequente, p225, pronunciou 1.932 palavras e 7.816 fonemas, enquanto que o locutor mais frequente, p263, pronunciou 3.817 palavras e 15.484 fonemas.

O locutor mais frequente pronunciou apenas duas vezes mais palavras que o locutor menos frequente, o que demonstra que a base de dados CSTR VCTK corpus é bem balanceada. Em virtude desse balanço, é esperado que as redes neurais não sejam enviesadas para um locutor em particular caso fossem treinadas com toda a base de dados. Conforme a Figura 20, ambas as distribuições podem ser consideradas distribuições normais.

Figura 20 – Distribuições de número de falantes por palavras pronunciadas e por fonemas pronunciados.



Fonte: Autoria própria.

Estes resultados demonstram que, em termos de números de palavras e fonemas pronunciados por cada locutor, a base de dados é balanceada o suficiente para ser utilizada em aplicações de aprendizado de máquina sem que exista viés para um locutor em particular. Para avaliar a possibilidade de generalização de um modelo treinado a partir destes dados, é importante que a distribuição das palavras pronunciadas na base de dados seja similar à distribuição das palavras ditas no dia-a-dia. A tabela 3 apresenta a distribuição real das palavras da língua inglesa, e é similar à distribuição da base de dados utilizada. Isso sugere que, em termos de conteúdo, a base CSTR VCTK Corpus é uma boa representação da língua inglesa em áudio e permite que um modelo treinado a partir dela consiga generalizar para áudios não pertencentes à base.

Tabela 3 – As 60 palavras mais comuns da língua inglesa.

Ordem	Palavra	Ordem	Palavra	Ordem	Palavra	Ordem	Palavra
1	the	16	he	31	or	46	who
2	be	17	as	32	an	47	get
3	to	18	you	33	will	48	which
4	of	19	do	34	my	49	go
5	and	20	at	35	one	50	me
6	a	21	this	36	all	51	when
7	in	22	but	37	would	52	make
8	that	23	his	38	there	53	can
9	have	24	by	39	their	54	like
10	I	25	from	40	what	55	time
11	it	26	they	41	so	56	no
12	for	27	we	42	up	57	just
13	not	28	say	43	out	58	him
14	on	29	her	44	if	59	know
15	with	30	she	45	about	60	take

Fonte: Oxford University Press, 2011. Acesso em 2023.

4.1.2 Análise dos arquivos de áudio

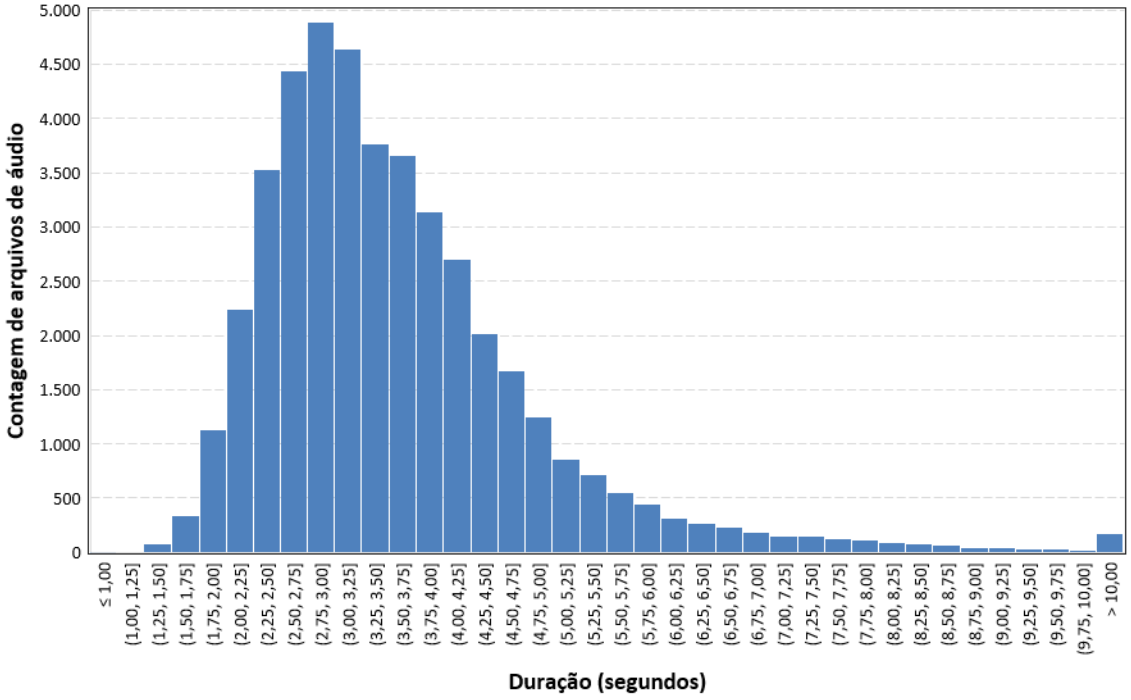
Quanto aos arquivos de áudio, é possível analisá-los tanto no domínio do tempo como no domínio da frequência. No domínio do tempo, algumas características relevantes são: duração do áudio, tempo de silêncio do áudio e amplitude do sinal. É particularmente importante avaliar quanto da base de dados é silêncio, porque se a duração dos intervalos de silêncio for muito superior à duração dos intervalos com falas, as redes neurais serão enviesadas a retornar uma saída nula, o que é indesejável.

Ao todo, na base constam 44.257 arquivos de áudio, totalizando 44 horas, 2 minutos e 46 segundos de gravações. Dessa forma, a duração média de um arquivo de áudio é de 3,58 segundos, contando com um desvio padrão de 1,33 segundos. O arquivo mais longo é o arquivo p310_023.wav, com uma duração de 19,28 segundos, enquanto que o arquivo mais curto é o arquivo p317_424.wav, com uma duração de 0,09 segundos. Ao inspecionar a sua transcrição, p317_424.txt, é indicado que nesse arquivo é lida a frase “That is not bigotry., o que não corresponde ao áudio devido a sua duração. Isso indica

que existem arquivos com falhas na base de dados que devem ser retirados da base.

Uma inspeção mais cuidadosa revela que apenas 10 arquivos de áudio possuem uma duração inferior a 1 segundo. Destes, apenas p363_354.wav (0,9 segundos) e p361_173.wav (0,85 segundos) são audíveis e correspondem ao texto indicado. Os outros 8 arquivos foram descartados da base. A Figura 21 apresenta o histograma da duração dos arquivos de áudio.

Figura 21 – Distribuição do número de arquivos de áudio por comprimento.

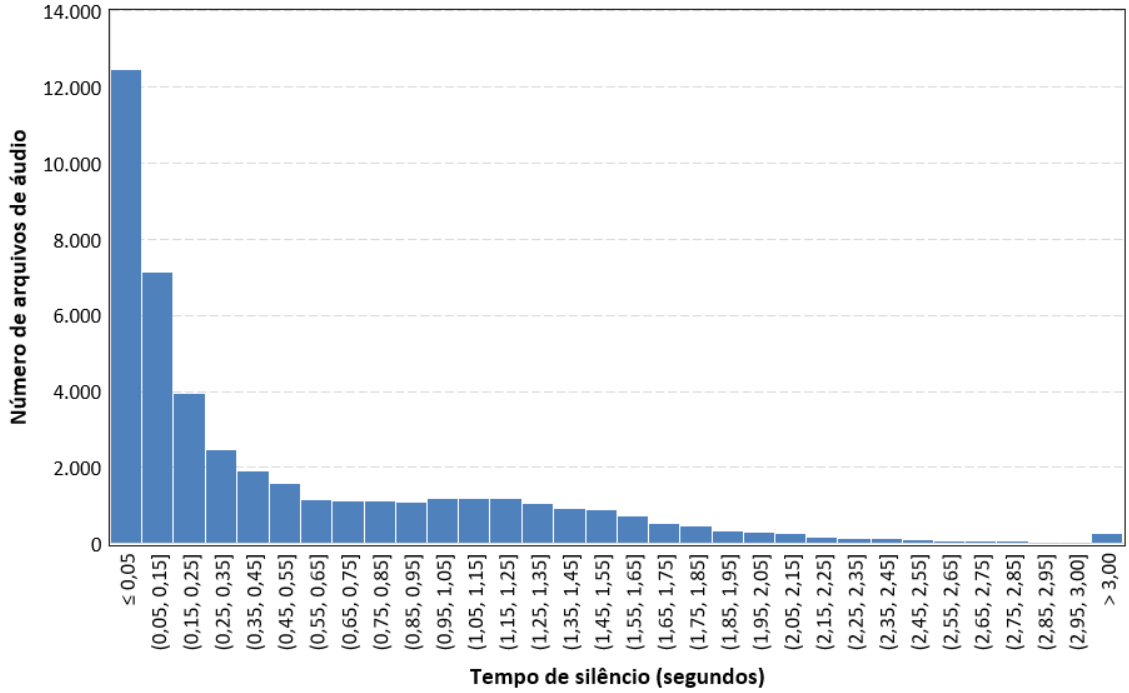


Fonte: Autoria própria.

É importante destacar que as redes neurais foram treinadas em segmentos de tamanho 512 (o que corresponde a 10,6 ms de duração), e o efeito disso é que arquivos mais longos terão uma participação maior durante o treinamento. Conforme a Figura 21, essa participação é negligível, dado que 89,09% dos arquivos possui uma duração entre 1,5 e 5 segundos.

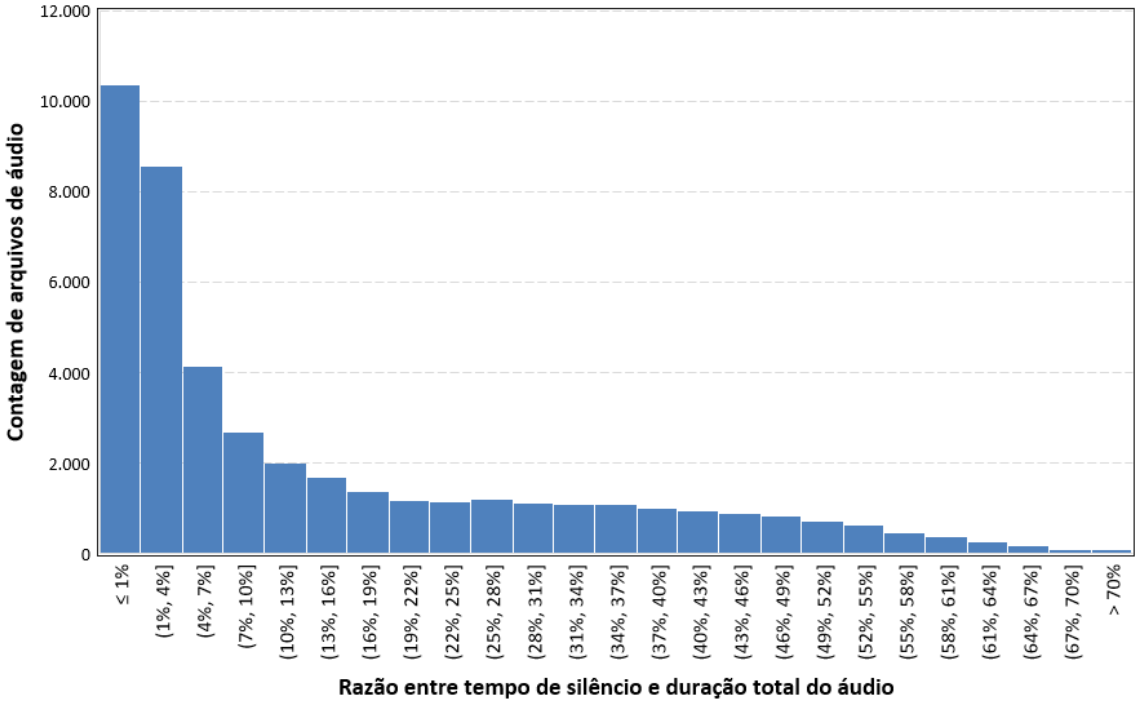
Em relação ao silêncio presente nos arquivos de áudio, foi analisado a quantidade de silêncio no início e no final de cada arquivo de áudio. Aqui, silêncio é considerado todo intervalo contínuo de amplitude de 60 dB abaixo da amplitude máxima do arquivo de áudio. Nessa caracterização, o silêncio no início dos arquivos contabiliza, no total, 4 horas, 9 minutos e 45 segundos (9,45% da base), enquanto que o silêncio no fim dos arquivos contabiliza 2 horas, 14 minutos e 19 segundos (5,08% da base).

Figura 22 – Distribuição do número de arquivos de áudio por tempo de silêncio.



Fonte: Autoria própria.

Figura 23 – Distribuição do número de arquivos de áudio pela razão entre silêncio e comprimento.



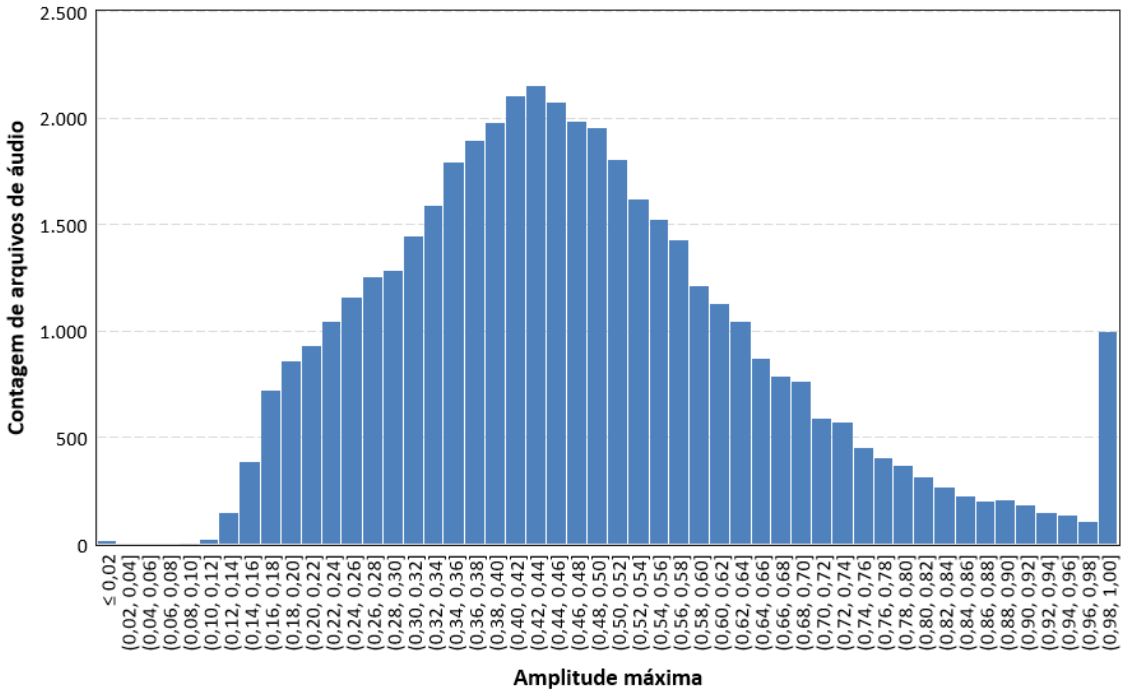
Fonte: Autoria própria.

Como esses comprimentos não são tão significativas, e devido ao fato de que deseja-

se que as redes neurais também aprendam a ampliar a taxa de amostragem de segmentos silenciosos corretamente, não foram removidos nem modificados da base de dados. As figuras 22 e 23 apresentam as distribuições de tempo de silêncio por arquivo e a razão entre tempo de silêncio e comprimento total do áudio. A grande maioria dos arquivos de áudio contém um tempo de silêncio baixo, corroborando com a conclusão de que não há necessidade de truncar os arquivos de áudio e reduzir o tempo de silêncio.

Uma outra característica temporal importante é a amplitude absoluta máxima. Se essa medida variar em ordens de grandeza, é necessário padronizar os arquivos de forma que a amplitude seja contida no intervalo $(-1, 1)$. O valor médio da amplitude absoluta máxima é de 0,474, com desvio padrão de 0,188. O valor mais alto dessa amplitude é de 0,99997, encontrado em 627 arquivos diferentes. A Figura 24 apresenta a distribuição de amplitude absoluta máxima.

Figura 24 – Distribuição do número de arquivos de áudio pela amplitude absoluta máxima.



Fonte: Autoria própria.

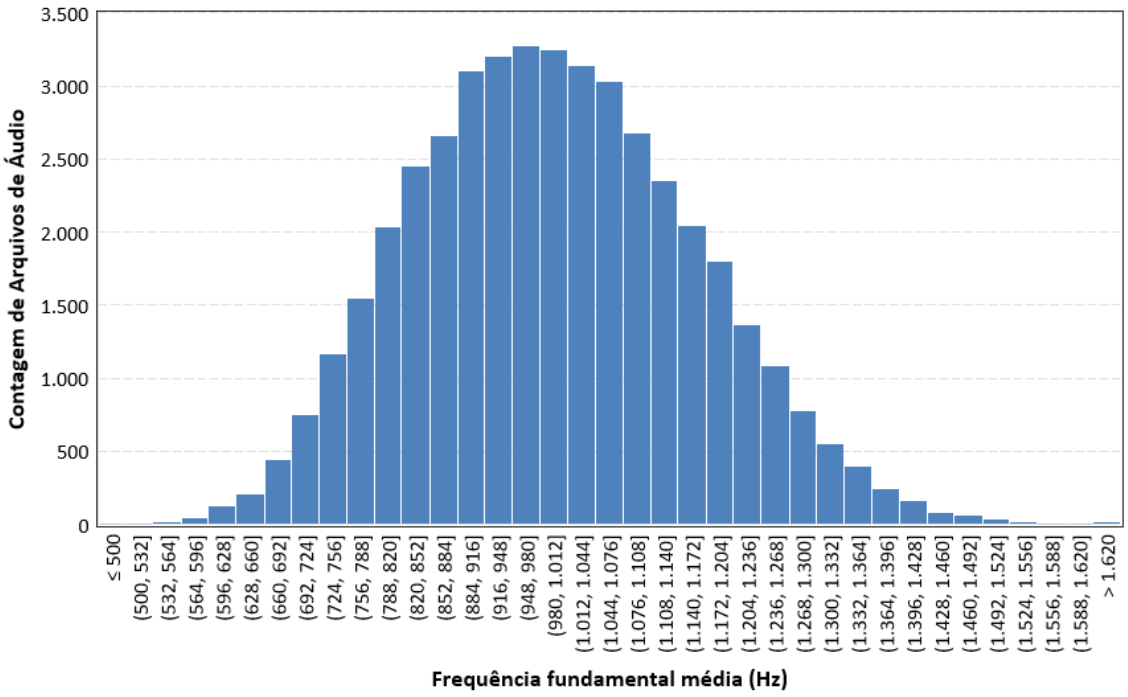
Como a amplitude máxima dos arquivos de áudio se concentra majoritariamente em uma ordem de grandeza, optou-se por não padronizá-los. Isso permite que as redes neurais aprendam também a lidar com diferentes amplitudes sonoras, e remove a necessidade de se modificar a amplitude uma vez que as redes estejam em produção. Portanto, não há a necessidade de transformar o arquivo de áudio nem no eixo temporal nem no eixo da

amplitude, de forma que não são adicionadas etapas desnecessárias para aplicar a super-resolução em um sinal de voz qualquer, mantendo o tempo de inferência otimizado.

Já no domínio da frequência, duas características são importantes: a frequência fundamental (o tom de um som) e o centroide espectral (correspondente ao "centro de massa" do espectro de um sinal). É esperado que as duas medidas sigam distribuições normais dado que diferentes falantes possuem diferentes timbres e frequências de fala.

Os dados apresentaram uma frequência fundamental média de 997,0 Hz, com desvio padrão de 163,8 Hz. Conforme a Figura 25, nota-se que a distribuição da frequência fundamental média é uma distribuição normal e simétrica. É importante destacar que estes dados correspondem à média das frequências fundamentais de cada segmento de tamanho 512 dos arquivos de áudio e não correspondem à verdadeira frequência fundamental da voz humana, que vale entre 85 Hz a 155 Hz. Métodos de captura e de filtragem também podem ser responsáveis por disfarçar um dos harmônicos como a frequência fundamental do sinal.

Figura 25 – Distribuição do número de arquivos de áudio pela frequência fundamental média.

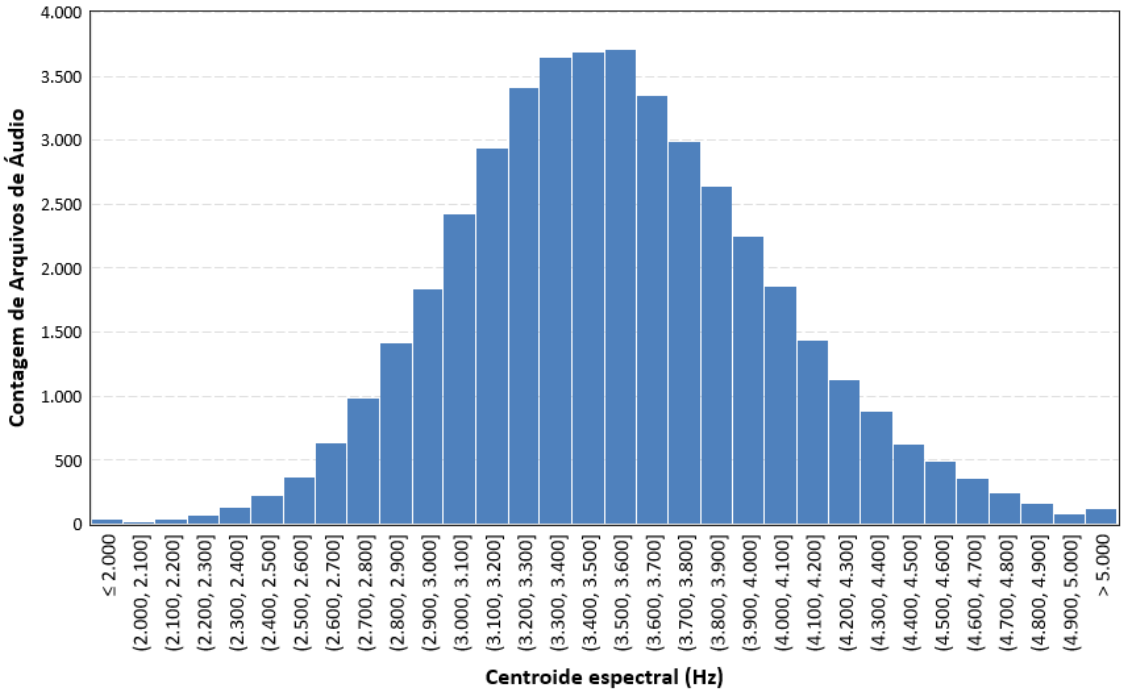


Fonte: Autoria própria.

Outra característica importante de áudios, o centroide espectral é uma medida que pondera as frequências presentes num sinal pelas suas energias, equivalente à obtenção

do centro de massa de um objeto sólido. Ele é uma medida de tendência central para sinais no domínio da frequência, e pode ser interpretado como: metade da energia do sinal se concentra nas frequências abaixo do centroide espectral e a outra metade se concentra nas frequências acima do centroide espectral. Conforme a Figura 26, observa-se que o valor médio do centroide espectral dos arquivos de áudio vale 3.531 Hz com desvio padrão de 490,5 Hz.

Figura 26 – Distribuição do número de arquivos de áudio pelo centroide espectral.

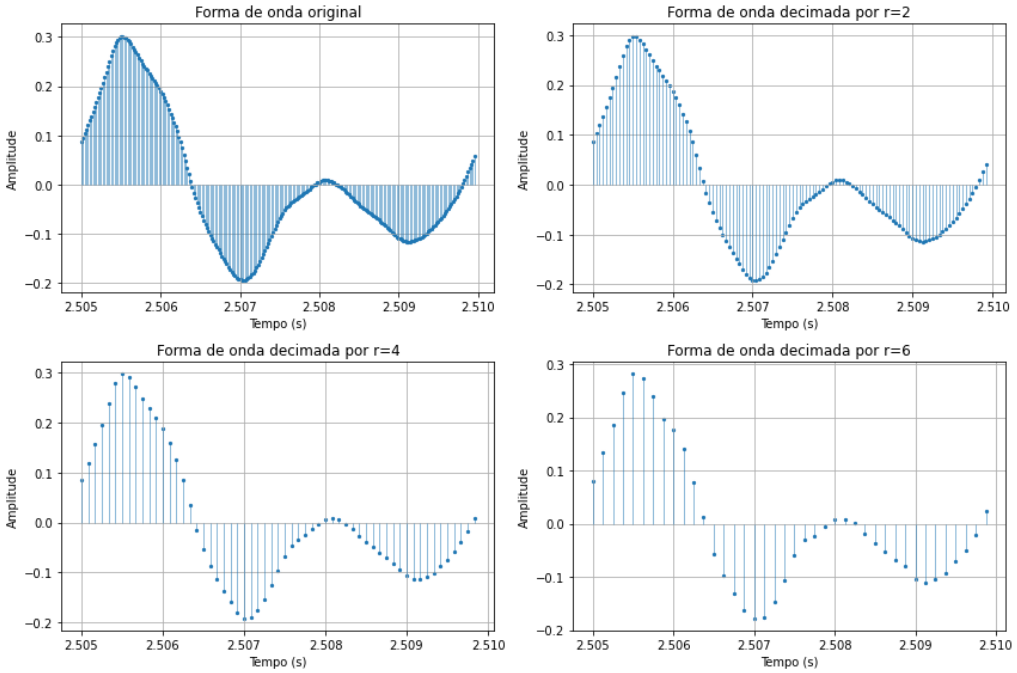


Fonte: Autoria própria.

A distribuição do centroide espectral também apresenta uma curva normal. Por conta do formato da distribuição da frequência fundamental média e do centroide espectral, é esperado que a rede neural obtenha um desempenho bom para arquivos de áudio próximos aos valores médios encontrados na base de dados. Isso porque há poucos exemplos na base cuja frequência fundamental e centroide espectral divergem significativamente dos valores médios, o que implica que a rede neural não conseguirá aproximar muito bem a super-resolução nestes casos.

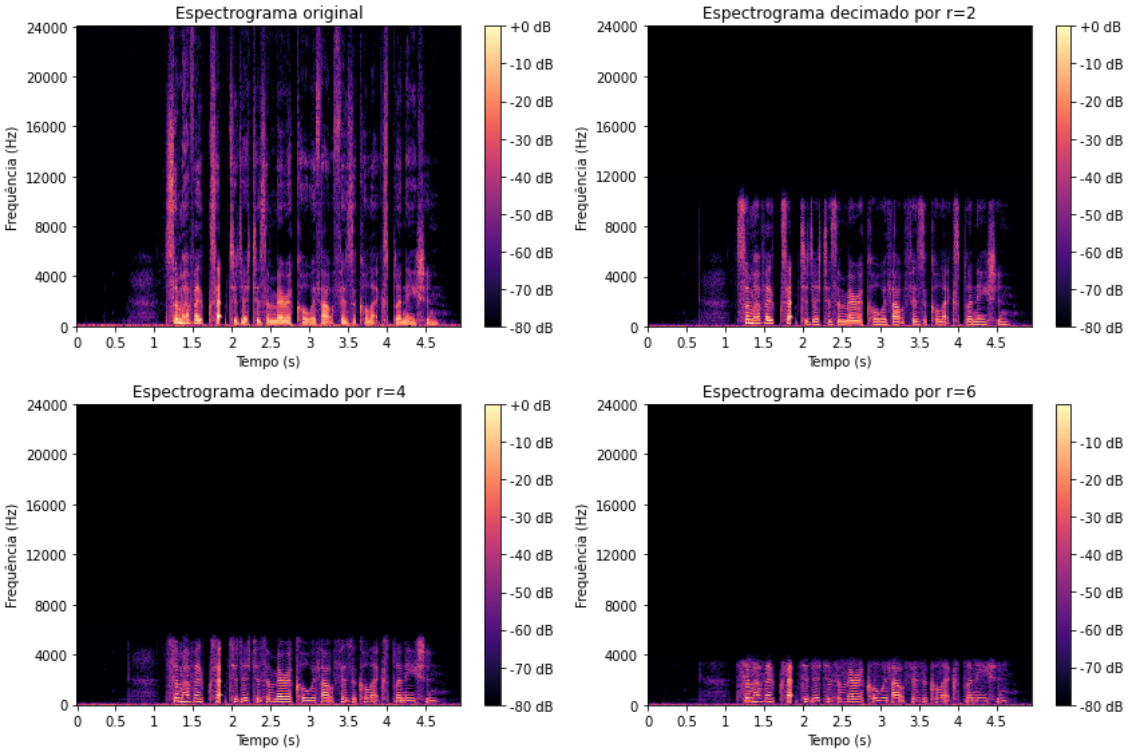
Finalmente, é importante analisar o formato de onda e espectro de um único arquivo de áudio, a fim de permitir a tomada de novas conclusões. A Figura 27 apresenta o formato de onda do arquivo original e do áudio decimado para os diferentes valores de r, e a Figura 27 apresenta os efeitos dessas decimações nos espectrogramas respectivos.

Figura 27 – Forma de onda do arquivo p233_013.wav e suas decimações para t entre 2,505 e 2,510 segundos.



Fonte: Autoria própria.

Figura 28 – Espectrograma do arquivo p233_013.wav e suas decimações.



Fonte: Autoria própria.

Os espectrogramas deixam claro o efeito de reduzir a taxa de amostragem: a frequência mais alta observada é reduzida pelo fator de decimação, e as frequências mais baixas são distorcidas. A super-resolução busca reconstruir essas frequências mais altas e reduzir as distorções das frequências mais baixas.

4.2 Treinamento das redes e otimização de hiper-parâmetros

A primeira etapa do treinamento das redes neurais foi a busca pelos hiper-parâmetros. Nessa fase, é completamente ignorado o conjunto de testes, e para cada iteração do treinamento foram utilizados apenas 3 arquivos .h5 (compondo 150 gravações de áudio diferentes) para treinamento e validação e $r = 4$. A parcela de validação desse conjunto foi escolhida aleatoriamente com embaralhamento de forma que 5% do conjunto de treinamento fosse destinado exclusivamente para validação. As redes foram treinadas em 20 épocas com um tamanho de lote de 32 nessa etapa. Assim, a rede é treinada, a cada época, com cerca de 170 mil pares de vetores de tamanho 512.

O primeiro conjunto de hiper-parâmetros avaliado foi a função de perda. A literatura tipicamente faz uso da função de perda MSE (erro médio quadrático). Foi considerada a hipótese de que a função de perda MAE (erro médio absoluto) pudesse ter um resultado melhor, então foi necessário comparar os modelos treinados tanto com a MSE como com a MAE.

O segundo conjunto de hiper-parâmetros avaliado foi a função de ativação. Na literatura, a Audio U-Net foi desenvolvida utilizando a função ReLU (KULESHOV; ENAM; ERMON, 2017). Porém, sabe-se que existem modificações da ReLU que permitem uma performance melhor. Dessa forma,foram avaliadas as funções Leaky ReLU, SWISH e MISH.

Para escolher qual função de ativação e função de perda utilizar, foi treinada a rede U-Net$_{small}$ nas 8 combinações possíveis de função de perda e função de ativação. A escolha dessa arquitetura para a otimização de hiper-parâmetros tem o seguinte racional: seria muito custoso (tanto em tempo de computação como em unidades de computação oferecidas) treinar a U-Net$_{large}$ nessa quantidade de dados, devido ao fato de possuir 94,6 milhões de parâmetros treináveis contra apenas 148,9 mil da U-Net$_{small}$. Essa é também a menor arquitetura dentre as analisadas, podendo ser mais rapidamente treinada.

A tabela 4 apresenta os resultados obtidos ao final de 20 épocas para a métrica SNR

nos respectivos conjuntos de validação. É possível observar que a função de perda MAE permitiu melhores resultados para todas as 4 funções de ativação observadas, assim como a função de ativação MISH obteve o melhor desempenho em ambos os cenários. Cada um desses treinamentos levou cerca de 2 horas para completar com a utilização da GPU V100 (130 TFLOPS).

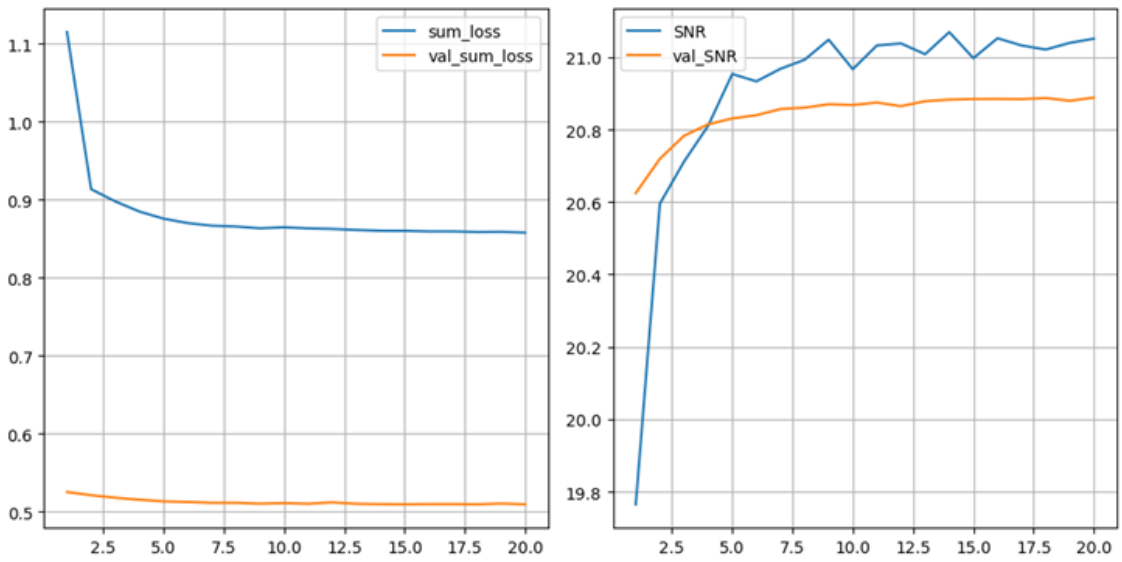
Tabela 4 – Resultados de SNR de validação médio para a U-Net$_{small}$.

	ReLU	Leaky ReLU	SWISH	MISH
MAE	17,82	20,89	19,68	**21,13**
MSE	17,12	20,77	19,31	**21,08**

Fonte: Autoria própria.

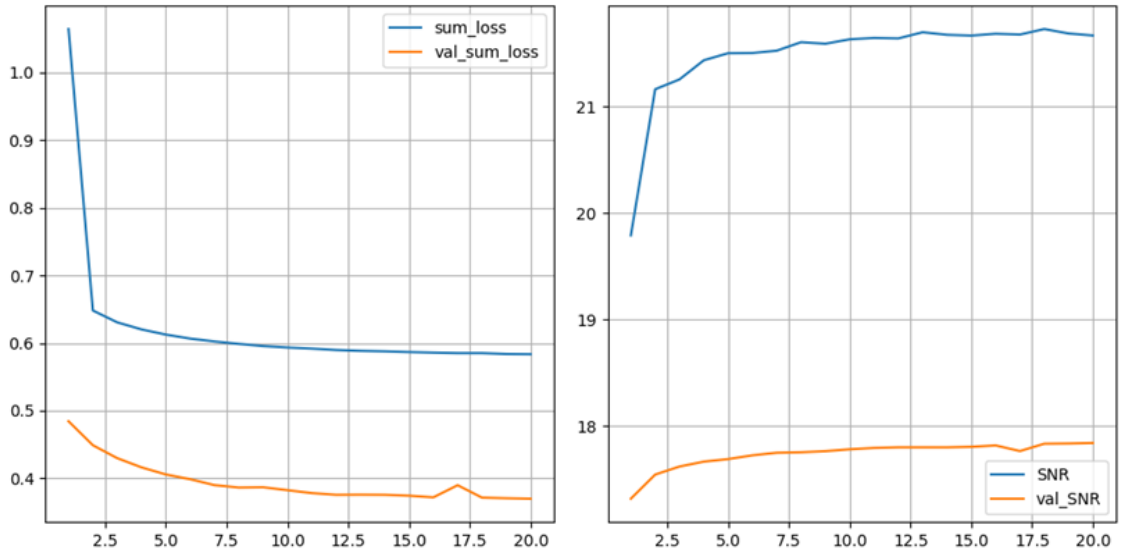
Além disso, a convergência é relativamente rápida. Já na primeira época de treinamento a rede neural U-Net$_{small}$ com perda MAE e ativação MISH apresentou um SNR de validação de 20,6, lentamente crescendo ao longo das 20 épocas, atingindo um SNR de 21,13 ao final do treinamento. De maneira similar, a rede com a ReLU também convergiu rapidamente, apesar de ter atingido um SNR máximo de 17,84. As figuras 29 e 30 apresentam as curvas de erro absoluto total (à esquerda) e de SNR (à direita) para os conjuntos de treino e validação.

Figura 29 – Treinamento da rede U-Net$_{small}$ utilizando a função de perda MAE, a função de ativação MISH e taxa de aprendizado 10^{-4}.



Fonte: Autoria própria.

Figura 30 – Treinamento da rede U-Net$_{small}$ utilizando a função de perda MAE, a função de ativação ReLU e a taxa de aprendizado 10^{-4}.



Fonte: Autoria própria.

Em ambos os casos, a perda de validação ficou abaixo da perda de treinamento. Normalmente, espera-se que o contrário aconteça, dado que a rede não aprende com o conjunto de validação. Contudo, pelo fato de a rede neural conter camadas Dropout, o erro de treinamento apresentado corresponde à rede neural operando com somente metade de seus neurônios ativos, enquanto que o erro de validação é calculado ao final de cada época com todos os neurônios ativos.

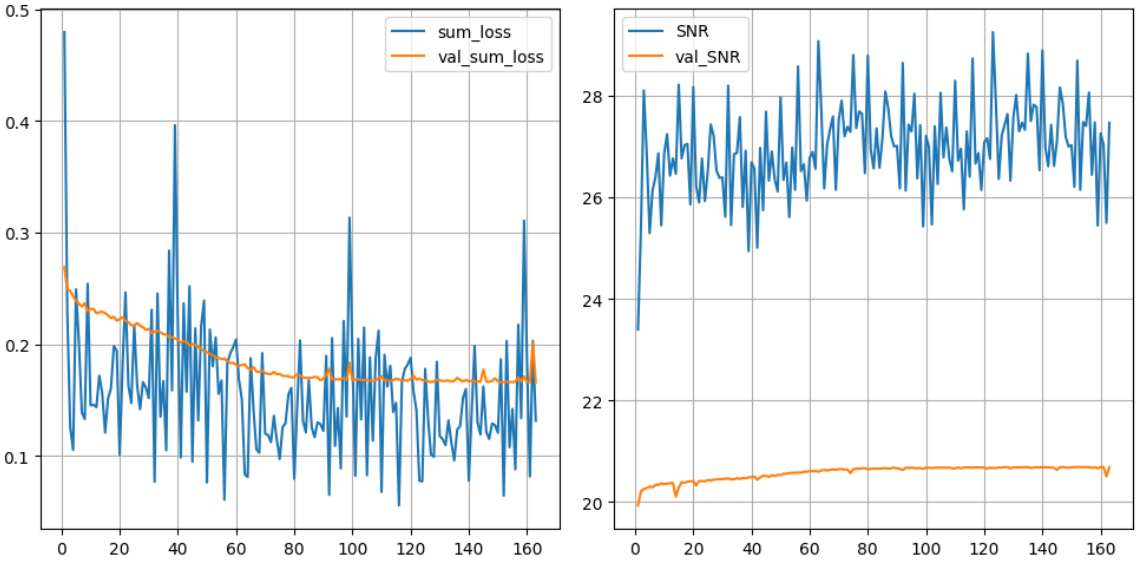
Além desses hiper-parâmetros, foi feita uma busca de tentativa e erro da taxa de aprendizado. Foram avaliadas taxas de aprendizado α de 10^{-3}, 10^{-4} e 10^{-5}. Observou-se que para $\alpha = 10^{-3}$ o treinamento divergiu, enquanto que com $\alpha = 10^{-5}$ o treinamento convergiu muito lentamente. Assim, para o treinamento final das redes neurais, optou-se por uma taxa de aprendizado com decaimento exponencial, cujo valor inicial α_0 corresponde a 10^{-4} e com taxa de decaimento de 0,96 a cada 500 mil lotes.

4.3 Resultados e comparações

Dados os hiper-parâmetros obtidos acima, as 4 arquiteturas (AE$_{small}$, AE$_{large}$, U-Net$_{small}$ e U-Net$_{large}$) foram treinadas para os três diferentes valores de r (2, 4 e 6) em 60 arquivos .h5 (correspondentes a 3000 arquivos de áudio .wav) escolhidos aleatoriamente. Como cada arquivo .h5 é muito grande (aproximadamente 1 GB de dados), não é possível

carregar todo o conjunto de treinamento na memória simultaneamente. Isso impede que a curva de erro de treinamento seja monótona, tendo em vista que a cada carregamento de arquivo .h5 a rede neural vê arquivos de áudios e falantes que nunca tinha visto antes, o que explica as oscilações frequentes da 31. Como o conjunto de validação foi o mesmo para todos os 12 modelos, a curva de erro de validação apresenta um comportamento mais suave.

Figura 31 – Treinamento da rede UNet$_{small}$ para $r = 4$.



Fonte: Autoria própria.

Uma vez treinados os 12 modelos, resta comparar suas métricas de desempenho. A SNR é trivial, pois já era analisada durante o treinamento. Para a obtenção da métrica LSD, que utiliza a transformada de Fourier de curto tempo (STFT), é necessário comparar os arquivos inteiros de áudio entre si. Para cada um dos 12 modelos, os seguintes passos foram tomados:

1. É aplicado um filtro passa-baixas ao sinal de forma a atenuar as distorções causadas pela etapa seguinte (Filtro de Chebyshev tipo I de ordem 1);
2. O áudio é decimado conforme a taxa de ampliação de largura de banda desejada;
3. O áudio decimado é interpolado por *splines* bicúbicas;
4. Cada áudio interpolado é segmentado em vetores de tamanho 512;
5. A cada segmento, aplica-se os 12 diferentes modelos treinados;
6. Os segmentos são reunidos novamente. O áudio resultante é o áudio super-resolvido, e pode ser comparado diretamente com o original;

Tabela 5 – Resultados médios obtidos para as diferentes arquiteturas de redes neurais no conjunto de testes.

		Spline	**AE**$_{small}$	**AE**$_{large}$	**U-Net**$_{small}$	**U-Net**$_{large}$
	SNR	22,28±3,51	25,88±3,70	26,32±3,53	26,06±2,97	**27,22±3,24**
r=2	LSD	3,64±0,13	2,54±0,12	**2,53±0,11**	3,22±0,08	3,1±0,10
	t_{med}	**0,013**	0,089	0,123	0,121	0,686
	SNR	17,01±5,32	20,513±5,04	20,58±5,19	21,72±4,44	**22,63±4,38**
r=4	LSD	5,13±0,29	**3,38±0,28**	3,42±0,25	4,71±0,24	4,57±0,20
	t_{med}	**0,010**	0,084	0,117	0,114	0,678
	SNR	13,06±4,63	18,20±5,41	17,75±4,05	**20,12±5,37**	19,03±4,69
r=6	LSD	5,87±0,43	**3,61±0,4**	3,75±0,40	5,26±0,36	5,04±0,30
	t_{med}	**0,008**	0,118	0,106	0,126	0,674

Fonte: Autoria própria.

7. Calcula-se a métrica LSD entre y e $\hat{y}$.

Além disso, o tempo de inferência passa a ser cronometrado a partir do passo 3 até o passo 6, a fim de simular a aplicação de cada modelo na prática. A tabela 5 demonstra os resultados obtidos. Os 9 modelos menores foram treinados em cerca de 2 horas cada um por uma GPU A100, enquanto que os 3 modelos maiores (U-Net$_{large}$) levaram cerca de 12 horas cada um.

Os valores de cada célula da tabela correspondem ao valor médio acrescido ou subtraído de um desvio-padrão da distribuição de resultados. Uma conclusão interessante é que, para valores mais altos de r, uma rede menor é preferível para minimizar a LSD. Uma possível explicação para este fenômeno é que, para sinais de menor taxa de amostragem (e, portanto, r maior), existem menos frequências a serem consideradas e filtradas pela rede neural.

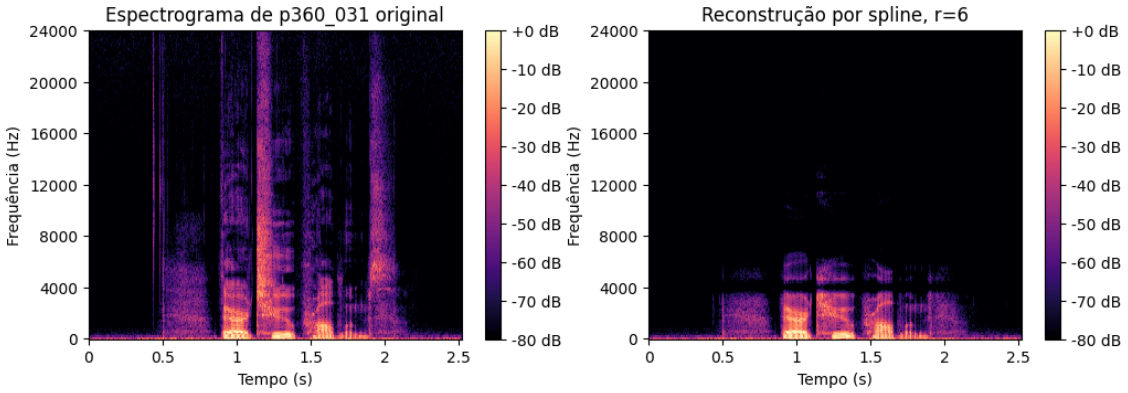
É fundamental salientar a importância do tempo de inferência. Um dos objetivos do presente trabalho era determinar se é possível resolver o problema de super-resolução de áudio e colocá-lo em prática. A tabela 5 apresenta o tempo mediano para a inferência dos arquivos de áudio de testes, e demonstra que, com modelos pequenos, é possível utilizar a rede neural em tempo real - A U-Net$_{small}$ consegue performances similares à U-Net$_{large}$ sendo entre 5 a 6 vezes mais rápida.

Foram, também, selecionados alguns modelos e analisados suas reconstruções para o áudio p360_031. Conforme a Figura 32, observa-se que a interpolação *spline* não reconstrói frequências próximas à taxa de amostragem f_s do sinal decimado (4 kHz) mas reconstrói as frequências próximas de $3f_s/2$.

Já a arquitetura *autoencoder*, conforme Figuras 33 e 34, consegue adicionar frequências de até 24 kHz, apesar dessa reconstrução possuir baixa intensidade e poucos detalhes. Além disso, as frequências próximas de 4 kHz não são reconstruídas, assim como na interpolação *spline*. O tamanho das redes não parece ter tido muita influência na reconstrução do áudio ao observar o espectro de ambas.

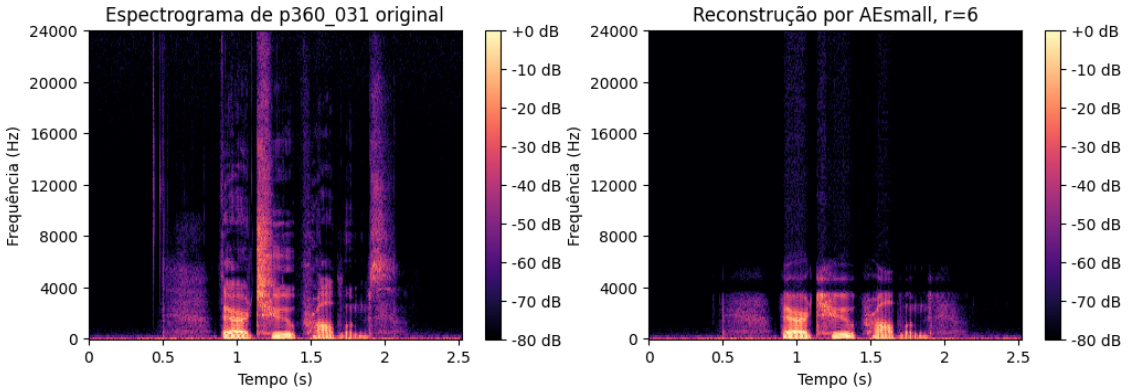
Por fim, as arquiteturas U-Net, conforme Figuras 35 e 36 conseguem adicionar as frequências mais altas com muito mais detalhes, como se observa no intervalo entre 1 e 1,6 segundos. Além disso, reconstroem as frequências próximas a 4 kHz perdidas pela decimação. É possível concluir, também, que a U-Net$_{small}$ aprendeu a adicionar um tom constante de 8 kHz (o que configura um artefato indesejado), enquanto que a U-Net$_{large}$ não apresenta esse comportamento. Os espectros também demonstram que a arquitetura maior é capaz de adicionar mais detalhes nas frequências mais altas reconstruídas.

Figura 32 – Reconstrução por interpolação spline para $r = 6$.



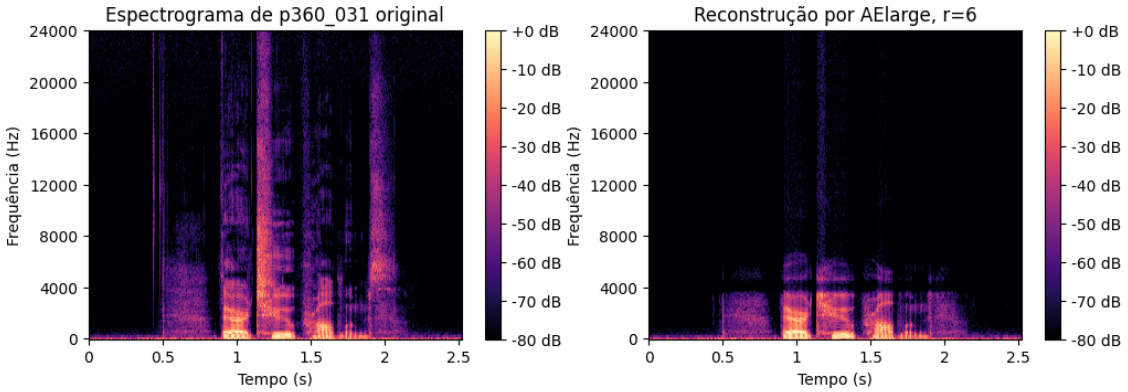
Fonte: Autoria própria.

Figura 33 – Reconstrução pela AE_{small} para $r = 6$.



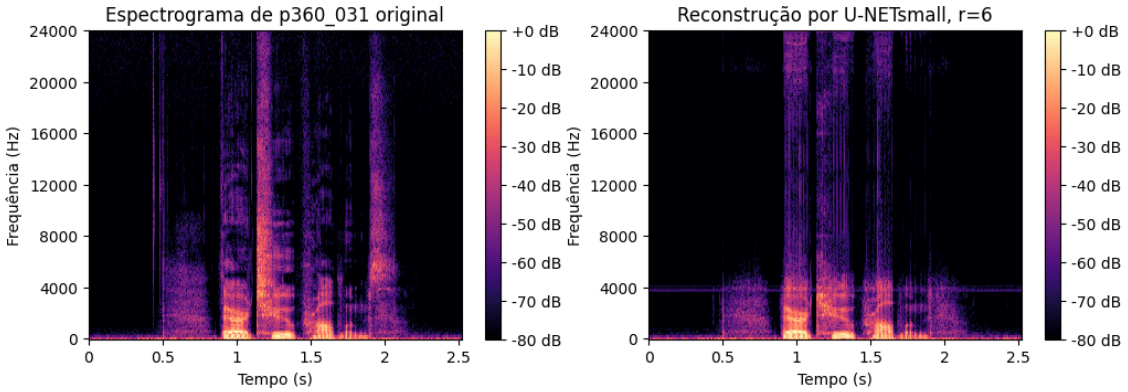
Fonte: Autoria própria.

Figura 34 – Reconstrução pela AE_{large} para $r = 6$.



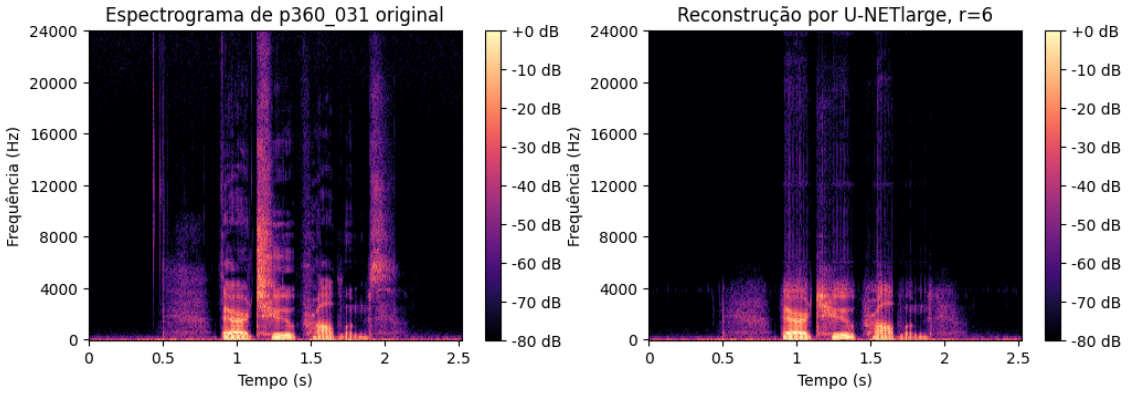
Fonte: Autoria própria.

Figura 35 – Reconstrução pela U-Net_{small} para $r = 6$.



Fonte: Autoria própria.

Figura 36 – Reconstrução pela U-Net$_{large}$ para $r = 6$.

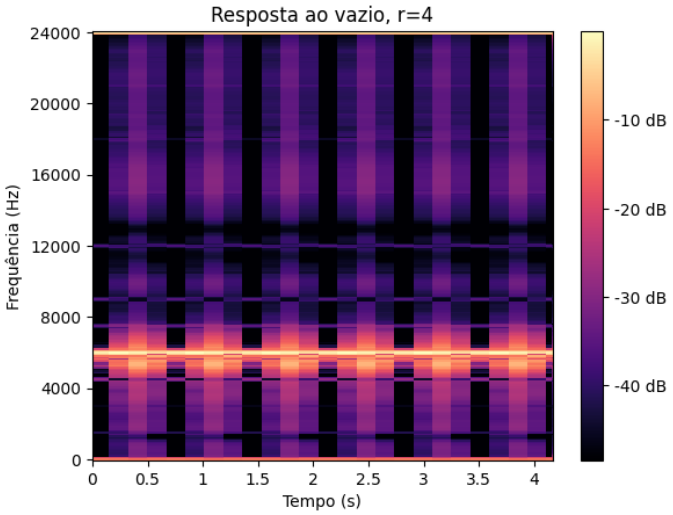


Fonte: Autoria própria.

4.4 Artefatos e possibilidades de melhoria

A fim de se maximizar a qualidade do sinal produzido e a fidelidade ao sinal original, é importante minimizar o número de artefatos gerados. A Figura 37 mostra que, em uma entrada vazia, há vários artefatos menores produzidos pela U-Net$_{large}$ para o caso em que r=4. É interessante perceber que a rede neural adiciona um tom de 6 kHz ao longo de toda a duração do sinal, assim como um vale em 12 kHz.

Figura 37 – Espectrograma da U-Net$_{large}$ aplicada à um sinal vazio para $r = 4$.



Fonte: Autoria própria.

Impedir artefatos é um problema extremamente difícil, principalmente porque não se há controle propriamente dito da rede neural durante seu treinamento. A maior fonte de artefatos parece ser a região próxima à frequência de amostragem de um sinal, portanto

uma possível abordagem é generalizar a rede neural para diversas frequências - isto é, treiná-la com diferentes frequências de amostragem iniciais e finais. O desafio, nesse caso, é garantir que a rede neural consiga identificar quais são as frequências desejáveis e indesejáveis.

Existem diversas outras possibilidades de aperfeiçoamento dos resultados. A primeira é aumentar a diversidade de sinais de áudio - incluir sinais musicais, vocalizações em outras línguas e sons artificiais na base de dados. Induzir efeitos nos áudios também poderia trazer resultados melhores, como ecos, reverberações e distorções. Além disso, as arquiteturas propostas nesse trabalho também seriam capazes de aprender a remover ruído dos arquivos de áudio além de aumentar sua resolução, podendo levar ao estudo de técnicas sofisticadas de remoção de ruído.

5 CONCLUSÕES

O presente trabalho buscou analisar o desempenho de duas técnicas do aprendizado profundo (*autoencoders* e *autoencoders* convolucionais) aplicadas ao problema da super-resolução de áudios de fala.

O objetivo, além de avaliar o desempenho das técnicas, foi desenvolver uma operação de aprendizado profundo de ponta a ponta: desde a escolha da base de dados, sua análise e processamento, até o treinamento e posterior utilização dos algoritmos avaliados.

Buscou-se, sobretudo, que os modelos treinados pudessem ser capazes de atuarem em tempo real, e, para isso, foram propostas duas versões para as duas arquiteturas avaliadas: uma menor e, portanto, mais rápida de ser treinada e de inferir, e outra maior e mais robusta. Era esperado que as arquiteturas menores tivessem um desempenho pior, porém este não foi o caso. Esse resultado demonstra a importância de partir de modelos menores em projetos de aprendizagem profunda e escalar somente conforme necessário.

Diversas escolhas foram realizadas para permitir aos modelos uma rápida inferência. Primeiro, optou-se por não extrair características desnecessárias, como a STFT (Transformada de Fourier de Curto Tempo) dos sinais de áudio. Depois, os arquivos de áudio foram segmentados em inúmeros vetores de tamanho 512, correspondendo ao mínimo intervalo de tempo que o ser humano consegue diferenciar em um sinal sonoro para uma taxa de amostragem de 48 kHz. Isso permite que as redes neurais sejam modeladas com um número menor de parâmetros, o que acelera tanto seu tempo de treinamento como seu tempo de inferência.

Por fim, foi escolhido aplicar uma interpolação *spline* ao sinal de áudio de baixa resolução antes da rede neural. Isso remove o ônus da rede neural de mapear um vetor noutro de maior dimensão, permitindo-a aprender apenas a identidade e remoção de distorções/-geração de frequências maiores.

Com base nessas premissas, foram propostas 15 combinações diferentes de hiper-parâmetros (que variavam a função de perda, as funções de ativação e a taxa de aprendizado). A seleção desses hiper-parâmetros se deu durante a fase de validação, na qual as redes neurais foram treinadas com uma amostra muito pequena do banco de dados em um número limitado de épocas. Em especial, foi demonstrado que a mera troca da função de ativação da ReLU para a MISH, na arquitetura avaliada, aumentou o SNR de 17,12 para 21,08.

Os resultados demonstraram que cada uma das arquiteturas propostas possui suas vantagens e desvantagens. A maior arquitetura, embora apresente os melhores valores de SNR, possui um tempo de inferência muito grande para ser utilizada em tempo real e em telecomunicações. Por outro lado, as arquiteturas menores, embora tenham um desempenho médio alto, possuem também uma variância relativamente alta, o que as torna um pouco inconsistentes na resolução do problema.

Surpreendentemente, os menores valores de LSD (distância logarítmica do espectro), métrica de dissimilaridade entre dois sinais de mesmo tamanho, foram observados num *autoencoder* de pequeno porte. Acredita-se que isso tenha sido uma consequência de utilizar entradas e saídas das redes neurais de baixa dimensionalidade. Uma possível investigação futura é avaliar a relevância do uso de diferentes tamanhos de entrada e saída para redes neurais aplicadas a sinais de áudio.

Por fim, para trabalhos futuros, sugere-se a realização de diferentes estudos e metodologias. A primeira é a avaliação de diferentes arquiteturas de redes neurais aplicadas à super-resolução de áudio. Recentemente, técnicas mais sofisticadas, como, por exemplo, os fluxos normalizantes (ZHANG *et al.*, 2021), têm sido desenvolvidas para a resolução do problema.

Uma outra melhoria importante é a investigação da utilização destas redes neurais na remoção de ruído e distorções de sinais de áudio, bem como a redução de artefatos gerados pela rede neural. Além disso, seria interessante a utilização de outras bases de dados, tais como bases de áudios em português, áudios de música de diversas regiões, etc.

Por último, sugere-se a utilização de outras metodologias para gerar os pares de áudios de baixa resolução e alta resolução. Por exemplo, capturar um único som com dois dispositivos diferentes simultaneamente - um de alta qualidade e outro de baixa qualidade - a fim de tentar reproduzir as características do dispositivo de alta qualidade por meio

de redes neurais aplicadas ao dispositivo de baixa qualidade. Imagina-se que o futuro da teoria de processamento de sinais se encontra na cada vez mais crescente utilização de redes neurais artificiais enquanto filtros e codificadores/decodificadores.

REFERÊNCIAS

ABE, M.; YOSHIDA, Y. More natural sounding voice quality over the telephone! An algorithm that expands the bandwidth of telephone speech. **NTT review**, [S.l.], v. 7, n. 3, p. 104–109, 1995.

ALBON, C. **Machine Learning with Python Cookbook**: practical solutions from preprocessing to deep learning. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2018.

DING, H.; SOON, Y.; YEO, C. K. Over-attenuated components regeneration for speech enhancement. **IEEE transactions on audio, speech, and language processing**, [S.l.], v. 18, n. 8, p. 2004–2014, 2010.

DONG, C. *et al.* Image super-resolution using deep convolutional networks. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v. 38, n. 2, p. 295–307, 2015.

DONG, Y. *et al.* A Time-Frequency Network with Channel Attention and Non-Local Modules for Artificial Bandwidth Extension. *In*: ICASSP 2020 - 2020 IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2020. **Proceedings [...]** IEEE, 2020.

DSACADEMY. **Introdução aos Autoencoders**. Data Science Academy. Deep Learning Book, 2022. Disponível em: <https://www.deeplearningbook.com.br/>. Acesso em: 30 jul. 2023.

FOURIER, J. **Théorie de la propagation de la chaleur dans les solides**. [S.l.: s.n.], 1807.

FUEMMELER, J. A.; HARDIE, R. C.; GARDNER, W. R. Techniques for the regeneration of wideband speech from narrowband speech. **EURASIP Journal on Applied Signal Processing**, [S.l.], v. 2001, n. 4, 2001.

FUKUSHIMA, K. Cognitron: A self-organizing multilayer neural network. **Biological Cybernetics**, [S.l.], v. 20, p. 121–136, 1975.

GAUSS, C. F. **Theoria Interpolationis Methodo Nova Tractata**. [S.l.: s.n.], 1866.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. (Adaptive Computation and Machine Learning series).

GéRON, A. **Hands-on machine learning with Scikit-Learn and TensorFlow** : concepts, tools, and techniques to build intelligent systems. [S.l.]: O'Reilly, 2017.

KULESHOV, V.; ENAM, S. Z.; ERMON, S. Audio Super Resolution using Neural Networks. , [S.l.], 2017.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning Internal Representations by Error Propagation**. Cambridge, MA, USA: MIT Press, 1986. p. 318–362.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, [S.l.], v. 521, n. 7553, may 2015.

LEDIG, C. *et al.* Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. , [S.l.], 2016.

LI, K.; LEE, C.-H. A deep neural network approach to speech bandwidth expansion. *In*: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2015., 2015. **Proceedings [...]** IEEE, 2015.

LIM, T. Y. *et al.* Time-frequency networks for audio super-resolution. *In*: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP), 2018., 2018. **Proceedings [...]** [S.l.: s.n.], 2018. p. 646–650.

MIRON, M.; DAVIES, M. High frequency magnitude spectrogram reconstruction for music mixtures using convolutional autoencoders. , [S.l.], 09 2018.

MISRA, D. Mish: A self regularized non-monotonic neural activation function. **CoRR**, [S.l.], v. abs/1908.08681, 2019.

MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-hill New York, 1997. v. 1, n. 9.

OPPENHEIM, A. V.; SCHAFER, R. W.; BUCK, J. R. **Discrete-Time Signal Processing**. 2nd ed. ed. [S.l.]: Prentice Hall, 1999. 172-176 p.

RAKHECHA, A. **Gradient Descent and its Types**. Disponível em: <https://towardsdatascience.com/https-medium-com-dashingaditya-rakhecha-understanding-learning-rate-dd5da26bb6de>. Acesso em: 15 jul. 2023.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for Activation Functions. **CoRR**, [S.l.], v. abs/1710.05941, 2017.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, [S.l.], v. 65, n. 6, p. 386–408, 1958.

SCHMIDHUBER, J. Deep learning in neural networks: an overview. **Neural Networks**, [S.l.], v. 61, p. 85–117, jan 2015.

SHANNON, C. Communication in the Presence of Noise. **Proceedings of the IRE**, [S.l.], v. 37, n. 1, p. 10–21, jan 1949.

SHI, W. *et al.* Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2016. **Proceedings [...]** [S.l.: s.n.], 2016. p. 1874–1883.

SHI, W. *et al.* **Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network**. 2016.

WAGH, A. **Gradient Descent and its Types**. Disponível em: <https://www.analyticsvidhya.com/blog/2022/07/gradient-descent-and-its-types/>. Acesso em: 15 jul. 2023.

WIKIPEDIA. **Quantização - Wikipédia, a enciclopédia livre**. Flórida: Wikimedia Foundation, 2006. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Quantiza%C3%A7%C3%A3o>. Acesso em: 09 jul. 2023.

WIKIPEDIA. **Butterworth filter - Wikipedia, The Free Encyclopedia**. Flórida: Wikimedia Foundation, 2007. Disponível em: <https://en.wikipedia.org/w/index.php?title=Butterworth_filter>. Acesso em: 15 jul. 2023.

WIKIPEDIA. **Spectrogram - Wikipedia, The Free Encyclopedia**. Flórida: Wikimedia Foundation, 2008. Disponível em: <https://en.wikipedia.org/w/index.php?title=Spectrogram>. Acesso em: 09 jul. 2023.

WIKIPEDIA. **Chebyshev filter - Wikipedia, The Free Encyclopedia**. Flórida: Wikimedia Foundation, 2009. Disponível em: <https://en.wikipedia.org/w/index.php?title=Chebyshev_filter>. Acesso em: 09 jul. 2023.

WIKIPEDIA. **Artificial neural network - Wikipedia, The Free Encyclopedia**. Flórida: Wikimedia Foundation, 2013. Disponível em: <https://en.wikipedia.org/w/index.php?title=Artificial_neural_network>. Acesso em: 23 jul. 2023.

WIKIPEDIA. **Convolutional neural network - Wikipedia, The Free Encyclopedia**. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network>. Acesso em: 30 jul. 2023.

WIKIPEDIA. **Amostragem de sinal - Wikipédia, a enciclopédia livre**. Flórida: Wikimedia Foundation, 2022. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Amostragem_de_sinal>. Acesso em: 09 jul. 2023.

WIKIPEDIA. **Som - Wikipédia, a enciclopédia livre**. Flórida: Wikimedia Foundation, 2023. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Som>. Acesso em: 09 jul. 2023.

WIKIPEDIA. **Hearing range - Wikipedia, The Free Encyclopedia**. Flórida: Wikimedia Foundation, 2023. Disponível em: <https://en.wikipedia.org/w/index.php?title=Hearing_rangeoldid=1152440721>. Acesso em: 09 jul. 2023.

WOLPERT, D. H. The Lack of A Priori Distinctions Between Learning Algorithms. **Neural Computation**, [S.l.], v. 8, n. 7, p. 1341–1390, 10 1996.

YAMAGISHI JUNICHI; VEAUX, C. M. K. **CSTR VCTK Corpus**: english multi-speaker corpus for cstr voice cloning toolkit (version 0.92). [S.l.]: University of Edinburgh. The Centre for Speech Technology Research (CSTR), 2019.

YASUKAWA, H. Restoration of wide band signal from telephone speech using linear prediction error processing. *In*: PROCEEDING OF FOURTH INTERNATIONAL CONFERENCE ON SPOKEN LANGUAGE PROCESSING. ICSLP'96, 1996. **Proceedings [...]** [S.l.: s.n.], 1996. v. 2, p. 901–904.

YEHOSHUA, R. **Perceptrons**: the first neural network model. Disponível em: <https://towardsdatascience.com/perceptrons-the-first-neural-network-model-8b3ee4513757>. Acesso em: 23 jul. 2023.

ZHANG, K. *et al.* WSRGlow: a glow-based waveform generative model for audio super-resolution. **ArXiv**, [S.l.], v. abs/2106.08507, 2021.