CATÓLICA
ESCOLA SUPERIOR DE BIOTECNOLOGIA

PORTO

APLICAÇÃO DE MÉTODOS DE APRENDIZAGEM DE MÁQUINA PROFUNDA PARA REDUÇÃO DE RUÍDO EM SINAIS DE FALA

por
José Luís Meias Coutinho de Paiva

Junho 2024

CATÓLICA
ESCOLA SUPERIOR DE BIOTECNOLOGIA

PORTO

APLICAÇÃO DE MÉTODOS DE APRENDIZAGEM DE MÁQUINA PROFUNDA PARA REDUÇÃO DE RUÍDO EM SINAIS DE FALA

Tese apresentada à Escola Superior de Biotecnologia da Universidade Católica Portuguesa para cumprir os requisitos necessários à obtenção do grau de Mestre em Engenharia Biomédica

por
José Luís Meias Coutinho de Paiva

Supervisor: Prof. Doutor Pedro Miguel de Luís Rodrigues
Escola Superior de Biotecnologia, Universidade Católica Portuguesa

Co-Supervisor: Prof. Doutor Bruno Catarino Bispo
Universidade Federal de Santa Catarina

Junho 2024

Resumo

A digitalização e evolução dos meios de comunicação têm sido temas amplamente discutidos, e as tecnologias relacionadas também são alvo de debate. A comunicação tornou-se mais rápida e eficiente com os avanços tecnológicos, mas surgiram problemas como a distorção do som por ruídos, ecos e reverberações. Para mitigar esses distúrbios, a Inteligência Artificial (IA) pode desempenhar um papel crucial. Este trabalho visou criar mecanismos de IA para identificar e atenuar o Eco Acústico, focando-se nas Redes Neuronais Convolucionais (CNN) para identificar diferentes perturbações sonoras.

Foi usada uma CNN para identificar automaticamente 4 tipos de sinais sonoros: sinal limpo, sinal limpo + eco, sinal limpo + ruído e sinal limpo + reverberação. Os sinais limpos foram extraídos da base de dados TIMIT da *Defense Advanced Research Projects Agency* (DARPA) e manipulados para obter os restantes sinais com diferentes tipos de perturbação. Os espetrogramas de cada sinal foram calculados e usados como entrada para 4 CNNs. A melhor CNN atingiu uma precisão de 98%, demonstrando grande capacidade para diferenciar os diferentes tipos de sinal + perturbações em condições controladas.

Na Atenuação do Eco Acústico, os resultados desejados foram mais difíceis de alcançar. Diversas abordagens foram utilizadas, incluindo o uso de sinais de fala manipulados e diferentes saídas esperadas, como sinais de fala reconstruidos sem eco e espetrogramas. Foram utilizadas Redes Neuronais Profundas (DNN), CNNs e Autoencoders Convolucionais, mas os resultados não foram satisfatórios, onde a métrica PESQ, usada para avaliar a qualidade da fala percebida em sinais de áudio, obteve uma média de 1,12 para os Autoencoders, abaixo das expectativas.

Em resumo, a tarefa de Classificação foi bem-sucedida e serve de base para trabalhos mais complexos. Os resultados menos promissores na Atenuação do Eco Acústico indicam a necessidade de melhorias e a possível utilização de outras técnicas, como redes BLSTM, GRU e Autoencoders recorrentes.

Palavras-chave: IA, Eco, Reverberação, Ruído, PESQ, *Autoencoders* Convolucionais

Abstract

The digitalisation and evolution of communication media have been widely discussed topics, and the related technologies are also subject to debate. Communication has become faster and more efficient with growing technological advancements, but issues such as sound distortion due to noise, echo, and reverberations surged into existence. To mitigate these disturbances, AI can play a crucial role. This work aimed to create AI mechanisms to identify and attenuate Acoustic Echo, focusing on CNNs to identify different sound disturbances.

A CNN was used to automatically identify 4 types of sound signals: clean signal, clean signal + echo, clean signal + noise, and clean signal + reverberation. The clean signals were extracted from the TIMIT database of DARPA and manipulated to obtain the remaining signals with different types of disturbances. The spectrograms of each signal were calculated and used as input for 4 CNNs. The best CNN achieved an accuracy of 98%, demonstrating a great ability to differentiate the different types of signals + disturbances under controlled conditions.

In Acoustic Echo Attenuation, the desired results were harder to achieve. Various approaches were used, including the use of manipulated speech signals and different expected outputs, such as reconstructed speech signals without echo and spectrograms. DNNs, CNNs, and Convolutional Autoencoders were used, but the results were unsatisfactory, with the PESQ metric, used to evaluate the perceived speech quality in audio signals, averaging 1.12 for the Autoencoders, below expectations.

In summary, the Classification task was successful and serves as a basis for more complex work. The less promising results in Acoustic Echo Attenuation indicate the need for improvements and the possible use of other techniques, such as BLSTM networks, GRU, and recurrent Autoencoders.

Keywords: AI, Echo, Reverberation, Noise, PESQ, Convolutional *Autoencoders*

Agradecimentos

Para começar, quero agradecer aos meus Orientadores, o Professor Dr. Pedro Miguel Rodrigues da Escola Superior de Biotecnologia (ESB) da Universidade Católica Portuguesa (UCP) do Porto e o Professor Dr. Bruno Catarino Bispo da Universidade Federal de Santa Catarina (UFSC), pelo apoio e orientação dados ao longo desta tese e pelo desafio que propuseram como tema.

Em segundo lugar, quero agradecer à própria UCP e, mais concretamente, à ESB pela oportunidade de usufruir das suas instalações e serviços durante o meu percurso académico.

Por último, quero agradecer à minha família, os meus avós, tios, tias, primos e, especialmente, os meus Pais por todo o apoio e carinho que me proporcionaram, pela motivação para ir em frente com os meus objetivos e pela ajuda a construir as bases da minha vida futura.

José Luís Paiva

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ACC	*Accuracy*
AEC	*Acoustic Echo Cancellation*
ANN	*Artificial Neural Network*
BD	Base de Dados
CNN	*Convolutional Neural Network*
DARPA	*Defense Advanced Research Projects Agency*
DL	*Deep Learning*
DNN	*Deep Neural Network*
DTFT	*Discrete-Time Fourier Transform*
EAM	Erro Absoluto Médio
ECB	Entropia Cruzada Binária
ECCE	Entropia Cruzada Categórica Esparsa
EQM	Erro Quadrático Médio
ESB	Escola Superior de Biotecnologia
F1	*F1-score*
FN	*False Negative*
FNN	*Feedforward Neural Network*
FP	*False Positive*
FT	*Fourier Transform*
GRU	*Gated Recurrent Unit*
IA	Inteligência Artificial
IRM	*Ideal Ratio Mask*
ITU	*International Telecomunications Union*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
NN	*Neural Network*

PESQ	*Perceptual Evaluation of Speech Quality*
PH	Perda de *Huber*
PPV	*Positive Predictive Value*
ReLU	*Rectified Linear Unit*
RIR	*Room Impulse Response*
RNN	*Recurrent Neural Network*
STFT	*Short Time Fourier Transform*
Tanh	Tangente Hiperbólica
TN	*True Negative*
TP	*True Positive*
TPR	*True Positive Rate*
UCP	Universidade Católica Portuguesa
UFSC	Universidade Federal de Santa Catarina
VS Code	*Visual Studio Code*

CAPÍTULO 1

Introdução

O capítulo que se segue introduz o tema e a estrutura da Tese. Apresenta a motivação para a realização do trabalho, os objetivos, o estado da arte e a organização do documento. Faz a contextualização teórica da Tese, define o conceito de sinal de fala, expõe o que são os distúrbios sonoros e como afetam os sinais sonoros, focando a análise em distúrbios provenientes de eco acústico, ruído gaussiano e reverberação. O capítulo termina com uma apresentação sobre a temática de redes neuronais, desde a definição e tipos de rede usadas até aos componentes bases das mesmas, bem como as suas aplicações.

1.1 Motivação

A motivação para a realização deste trabalho conjuga diversas razões: o desejo de desenvolver capacidades de trabalho em aprendizagem computacional, aprender a desenvolver um sistema capaz de fazer decisões por si mesmo e de o aplicar em situações reais; a vontade em desenvolver capacidades em contexto científico, isto é, melhorar métodos de investigação aprendidos em contexto educacional aperfeiçoando o conhecimento adquirido e técnicas para o difundir; o acreditar que todas essas valências serão fundamentais para um bom desempenho profissional futuro; a convicção de que este trabalho pode trazer uma contribuição positiva para os avanços na área de inteligência artificial aplicada à área de comunicação, nomeadamente desenvolvendo um sistema capaz de melhorar a qualidade de tele-conferências entre profissionais de saúde e pacientes, permitindo aumentar a inteligibilidade no processo de comunicação remota [1]. Finalmente, reconhecendo a grande utilidade dos aparelhos auditivos cujo funcionamento depende de uma boa qualidade de som, daí a necessidade de haver mecanismos necessários para eliminar distúrbios como o eco ou ruído [2], acredita-se que

direcionando este trabalho para a identificação de distúrbios auditivos e cancelamento de eco acústico, ele poderá constituir uma mais valia para auxiliar os pacientes que necessitam destes aparelhos.

1.2 Objetivos

O objetivo principal desta tese de mestrado é adaptar uma série de Redes Neuronais Artificiais (ANN – *Artificial Neural Network*) para duas áreas de atuação: uma para discriminar distúrbios auditivos, sendo eles o eco acústico, o ruído gaussiano e a reverberação e outra para atenuar eco acústico em sinais de fala. Haverá um especial foco em Redes Neuronais Convolucionais (CNN – *Convolutional Neural Network*) e Autoencoders. Para melhor compreensão da ordem de trabalho, foram delineados uma série de objetivos específicos:

1. Obtenção de uma base de dados que contenha sinais de fala variados;
2. Pesquisa bibliográfica que permita criar uma base de conhecimentos tanto sobre sinais de fala como de redes neuronais;
3. Programar em Python um modelo de Redes Neuronal (NN – *Neural Network*) para discriminar distúrbios auditivos;
4. Programar em Python, vários modelos para Cancelamento de Eco Acústico (AEC – *Acoustic Echo Cancellation*);
5. Análise e discussão dos resultados obtidos, com as respetivas conclusões;

1.3 Dificuldades e soluções adotadas

O primeiro problema encontrado para a realização deste trabalho foi a obtenção de uma Base de Dados (BD) rica em sinais de fala e de Respostas ao Impulso de uma sala (RIR) para criar sinais de eco para treino e teste das redes neuronais. Este problema foi superado com o download da BD TIMIT pertencente à DARPA a partir do site *GitHub* (https://github.com/philipperemy/timit) e com a ajuda do coorientador desta tese que forneceu as RIR. A segunda dificuldade encontrada foi o desenho das redes neuronais. Dado o relativo desconhecimento da programação de NN no início do trabalho, foi necessário realizar uma aprendizagem simultânea com a realização do trabalho. A terceira dificuldade prendeu-se com a avaliação objetiva do trabalho. Foi necessário pesquisar métricas já implementadas noutro trabalhos para fazer a sua escolha final.

1.4 Redes Neuronais

Para falar de redes neuronais, é necessário definir o que é Inteligência Artificial (IA). IA é o estudo de técnicas de construção de máquinas e computadores capazes de aprender, raciocinar e atuar como

um ser humano, sendo possível usar este tipo de técnicas para analisar dados cuja escala está muito além daquela que um ser humano é capaz de analisar [3]. Dentro de IA, existe um subconjunto de técnicas denominado por *Machine Learning* (ML). ML permite automatizar a aprendizagem e melhorar uma máquina ou sistema recorrendo a algoritmos em detrimento de programação explícita. Quanto mais dados forem fornecidos aos algoritmos, melhor será o desempenho dos modelos resultantes [4]. *Deep Learning* (DL) é um subconjunto de técnicas de ML como esquematizado na Figura 1.1. A grande diferença entre ML e DL, está na capacidade de DL de realizar uma extração automática de caraterísticas a partir dos dados fornecidos enquanto que ML requer que a extração de caraterísticas seja realizada por mão humana [5]. Outra diferença está no volume de dados. Como a extração de caraterísticas é automatizada, a DL é capaz de lidar com uma quantidade de dados muito superior a ML.

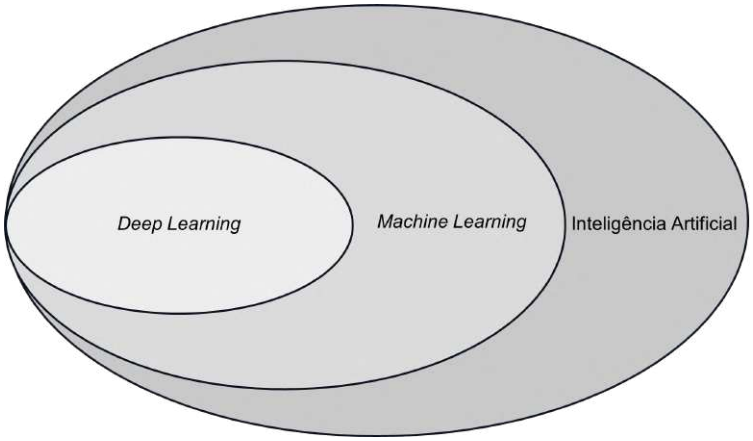
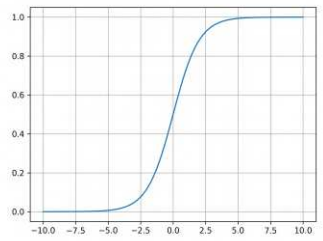
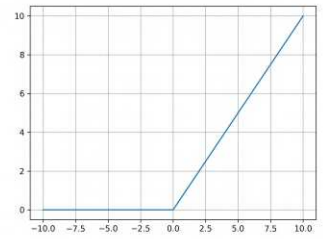
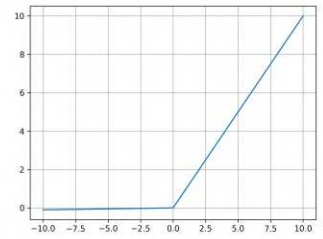
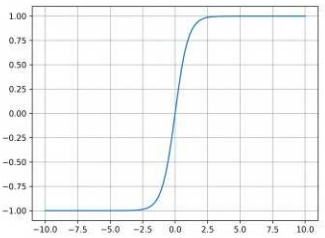
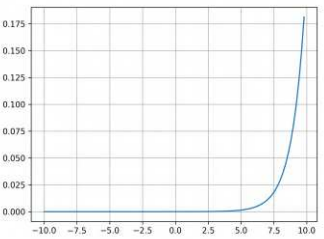


Figura 1.1: Diagrama de Venn que representa a relação entre IA, ML e DL.

Redes neuronais (NN), também conhecidas como Redes Neuronais Artificiais (ANN), são uma ferramenta de ML inspirada no funcionamento do cérebro humano, compostas por camadas de neurónios interligados entre si, simulando neurónios cerebrais humanos, que processam uma pequena parte dos dados fornecidos e que depois transmitem ao próximo neurónio na cadeia de processamento. A cada entrada no neurónio é atribuído um peso que irá determinar a importância de cada entrada para o resultado na saída do neurónio. Quanto maior for o peso, maior é importância da variável de entrada a que está associado. Depois da atribuição, as variáveis são multiplicadas pelos respetivos pesos e o seu resultado somado. Segue-se uma função de ativação (ver tabela 1.1 para as funções de ativação mais usadas) para determinar a saída que será comparada com um limiar ajustável. Se a saída for superior ao limiar, o neurónio será ativado e a sua informação será passada para a próxima camada da rede [7]. Estas entradas e saídas das NN são normalmente representados por uma estrutura de dados chamada de Tensor. Os tensores podem ser escalares, isto é, têm um único valor para representar um único ponto de dados, vetores, uma sequência de dados, matrizes, onde os dados estão organizados em linhas e colunas em representação de dados bidimensionais como imagens, ou

Tabela 1.1: Funções de ativação mais comuns [6]. Todas as representações gráficas foram geradas em Python.

Função Sigmoide	Atribui um valor entre 0 e 1 à entrada da função. Pode ser interpretado como probabilidade.	
ReLU (*Rectified Linear Unit*)	Atribui valor 0 a todos os valores de entrada negativos, deixando os valores positivos inalterados.	
***Leaky* ReLU**	Difere da ReLU na inclusão de um declive pequeno para os valores negativos, evitando neurónios sem atividade.	
Tanh (Tangente Hiperbólica)	Recorre a à função tangente hiperbólica para atribuir valores entre -1 e 1 aos valores de entrada do neurónio. Dá relevância aos valores negarivos.	
Softmax	Converte vetores de números reais numa distribuição probabilística. A soma da probabilidades previstas é 1.	

simplesmente multi-dimensionais, com três ou mais dimensões, normalmente usados para representar imagens RGB ou dados agrupados, nos quais a dimensão extra representa o tamanho do grupo. Estes tensores também são usados para guardar a informação sobre os pesos e limiares da rede [8].

Durante o treino e teste de uma rede, é necessário avaliar o seu desempenho. Para tal é usada uma função de perda como, por exemplo, a função do Erro Quadrático Médio (EQM) (ver tabela 1.2 para as funções de perda mais comuns). A rede irá tentar minimizar ao máximo a função de perda ajustando os valores dos pesos e limiares até atingir o ponto de convergência, isto é, o ponto onde o valor de perda é mínimo [7] como é demonstrado na Figura 1.2.

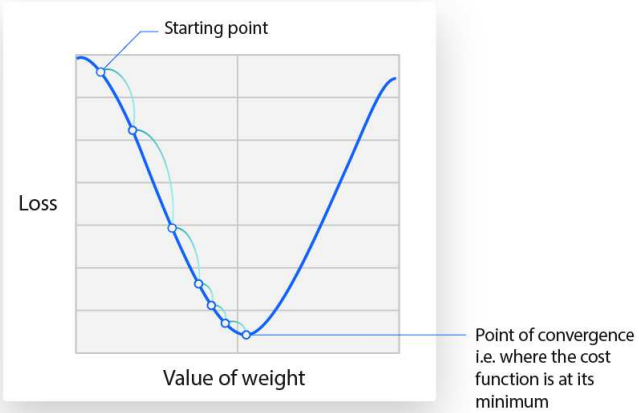


Figura 1.2: Representação gráfica do processo de minimização da perda em função do valor dos pesos [7].

Este processo de minimização de perda é mediado por algoritmos de otimização (ver tabela 1.3 para alguns algoritmos de otimização). Estes irão iterar sobre os dados e ajustar os pesos e limiares de acordo com os seus critérios. A taxa de mudança da função de perda em relação a estes dois parâmetros é chamada gradiente. Este irá indicar a direção na qual o algoritmo de otimização tem de atualizar os pesos e limiares na minimização das perdas. Este é um processo iterativo que se irá repetir até todos os dados de treino serem processados [9].

Há vários algoritmos para realizar este processo e um dos mais usados é o algoritmo Adam. Este método faz uso de médias em queda exponencial, um tipo de média móvel que atribui mais peso a dados mais recentes em detrimento dos dados mais antigos. Ele irá calcular esta média dos gradientes e do quadrado dos mesmos para depois usá-las para ajustar a taxa de aprendizagem, que controla a quantidade de parâmetros da rede atualizados em cada iteração de treino, e, consequentemente a convergência para o ponto de minimização de perda [11]. Este algoritmo foi apresentado por Kingma e Ba (2017) [12] e os resultados obtidos durante os seus testes sugerem que este otimizador é robusto e apropriado para aplicações de ML e DL.

Tabela 1.2: Algumas funções de perda mais comuns [10]. Para melhor compreensão, temos a seguinte nomenclatura: n é o número de amostras, i é a iteração, y é o valor real, $\hat{y}$ é o valor previsto, δ é um valor de limiar escolhido pelo usuário, p é a probabilidade prevista de p ser a classe real y e $\hat{y}_{i,y_i}$ é a probabilidade prevista da amostra da iteração i para a classe y correta.

Erro Quadrático Médio (EQM)	Calcula a média do quadrado da diferença entre os valores previstos e verdadeiros. Sensível a *outliers* devido ao foco em erros de maior dimensão. Converge na direção da média da distribuição alvo.	$EQM = \frac{1}{n} * \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
Erro Absoluto Médio (EAM)	Mede a diferença absoluta entre os valores previstos e reais. Menos sensível a *outliers* que a EQM. Foca-se mais na magnitude dos erros que na sua direção.	$EAM = \frac{1}{n} * \sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$
Perda de *Huber* (PH)	Combinação da EQM e da EAM. Quando o erro é pequeno comporta-se como a EQM, quando este é grande comporta-se como a EAM. Útil em tarefas cujos dados contenham ruído ou *outliers*.	$PH(y,\hat{y}) = \begin{cases} \frac{1}{2} * (y-\hat{y})^2, \lvert y-\hat{y} \rvert \leq \delta \\ \delta * (\lvert y-\hat{y} \rvert - \frac{1}{2}\delta), \lvert y-\hat{y} \rvert \geq \delta \end{cases}$
Entropia Cruzada Binária ou Perda Logarítmica (ECB)	Usada em classificação binária, mede a dissimilaridade entre a probabilidade prevista e a as classes binárias reais. Codifica as classes em vetores *one-hot*, nos quais a classe real tem valor de 1 e a outra classe tem valor 0.	$ECB(y,p) = \begin{cases} -log(p), y = 1 \\ -log(1-p), y = 0 \end{cases}$
Entropia Cruzada Categórica Esparsa (ECCE)	Usada em classificação multiclasse, mede a dissimilaridade entre a probabilidade prevista e a as classes em uso. Codifica as classes como valores inteiros, permitindo a escolha direta da classe em vez de se fazer um somatório de todas as classes possíveis se estas estivessem codificadas em vetores *one-hot*.	$EQM = \frac{1}{n} * \sum_{i=1}^{n} log(\hat{y}_{i,y_i})$

Tabela 1.3: Alguns dos algoritmos de otimização disponíveis [13] [14] Para melhor compreensão, temos a seguinte nomenclatura: w é o vetor de pesos, a é o rácio de aprendizagem, dw é o gradiente de w, t é o número da iteração, v é o vetor de pesos anteriores, β, β_1 e β_2 são rácios de decomposição exponencial e $\hat{v}$ e $\hat{v}$ são as equações de correção de viés.

Descida de Gradiente	O mais simples dos algoritmos, este calcula o mínimo local. Durante a sua operação, apresenta grande oscilação no eixo vertical, pois não guarda memória dos gradientes anteriores.	$w_t = w_{t-1} - a * dw_t$
Momento	Derivado do algoritmo anterior, este pretende aumentar a deslocação no eixo horizontal e diminuí-la no eixo vertical. A primeira equação guarda de valores de gradiente anteriores e a segunda realiza a normal descida do gradiente. Em suma, este algoritmo converge mais rapidamente e minimiza o risco de usar rácios de aprendizagem maiores para acelerar o processo todo.	$v_t = \beta * v_{t-1} + (1-\beta) * dw_t$ $w_t = w_{t-1} - a * v_t$
***AdaGrad* (Algoritmo de Gradiente Adaptativo)**	Este algoritmo avalia as componentes de peso e adapta o rácio de aprendizagem a cada uma. Pesos com maior gradiente irão ser atribuídos rácios de aprendizagem menores e vice versa. Isto permite lidar com o desaparecimento ou explosão de gradientes. A maior vantagem deste algoritmo é a automatização do ajuste do rácio de aprendizagem, o que leva à sua maior desvantagem, pois este mesmo rácio vai sendo reduzido durante as sucessivas iterações através da sua constante divisão por uma componente que é sempre positiva.	$v_t = v_{t-1} + dw_t^2$ $w_t = w_{t-1} - \frac{a}{\sqrt{v_t} + \varepsilon} * dw_t$
***RMSProp* (Propagação da Raiz Quadrada Média)**	Elaborado para corrigir a maior falha do *AdaGrad*, em vez de dar importância igual a todos os gradientes acumulados, este dá maior relevo aos gradientes mais recentes, pois não faz uma soma acumulada do quadrado destes. Em vez disso, faz uma média móvel exponencial do quadrado dos gradientes, permitindo também uma convergência mais rápida. Já o rácio de aprendizagem, não sofre a degradação que sofre em *AdaGrad*, apesar de ainda haver alguma.	$v_t = \beta v_(t-1) + (1-\beta) * dw_t^2$ $w_t = w_{t-1} - \frac{a}{\sqrt{v_t} + \varepsilon} * dw_t$
***Adam* (Estimativa do Momento Adaptativo)**	Desenhado para combinar as capacidades dos algoritmos *RMSProp* e de Momento, o algoritmo *Adam* guarda a memória das médias móveis exponenciais computadas e quadradas. Com estas, o algoritmo faz uma correção do viés, para manter uma aproximação mais correta da tendência dos gradientes anteriores, cálculo este que é depois usado para calcular o vetor de pesos corrente. Este algoritmo é capaz de se adaptar a quase todos os tipos de redes neuronais e requer pouca memória para trabalhar.	$v_t = \beta_1 v_{t-1} + (1-\beta_1) * dw_t$ $s_t = \beta_2 s_{t-1} + (1-\beta_2) * dw_t^2$ $\hat{v}_t = \frac{v_t}{1-\beta_1^t}$ $\hat{s}_t = \frac{s_t}{1-\beta_2 t}$ $w_t = w_{t-1} - \frac{a * \hat{v}_t}{\sqrt{\hat{s}_t} + \varepsilon} * dw_t$

As NN podem ser aplicadas a vários tópicos como classificação, onde dados são classificados em diferentes categorias (classificação de áudio, de imagem ou texto, por exemplo), regressão, onde a rede faz previsão de valores contínuos (por exemplo, preços de ações, número de clientes ou previsão de procura), deteção de objetos, processamento de linguagem e muitos outros [15]. Este trabalho irá focar-se no uso de Redes Neuronais Profundas (DNN). Estas NN derivam a sua designação da sua complexidade apesar de não haver consenso sobre qual critério a usar para definir esta caraterística, seja pelo gráfico computacional da rede ou o gráfico da modelagem probabilística da rede mais comummente usada em modelos probabilísticos profundos, como Chang (2021) [16] para o seu trabalho. Seja por que critério for, é seguro dizer que redes ditas profundas são de maior complexidade do que aquelas usadas em ML [15]. Para efeitos desta dissertação, vai ser considerado como rede profunda, qualquer rede como duas ou mais camadas escondidas, como a que está representada na figura 1.3.

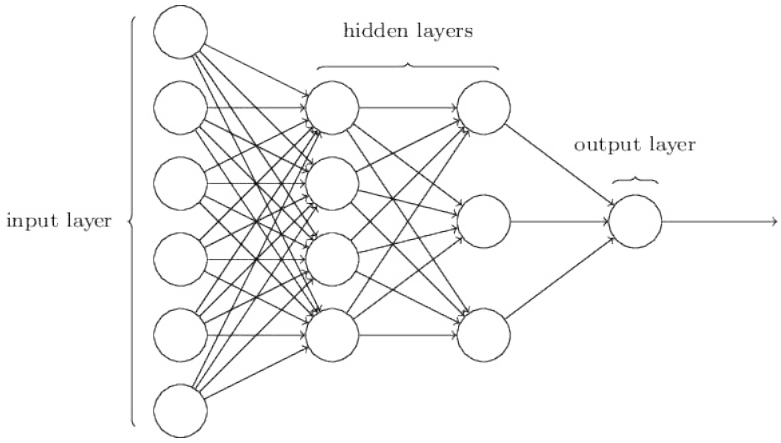


Figura 1.3: Esquematização de uma DNN [17]

1.4.1 Aprendizagem

Um conceito muito importante no estudo de NN, é a aprendizagem. Há vários tipos de aprendizagem, mas os três principais são: aprendizagem supervisionada, aprendizagem não-supervisionada e aprendizagem reforçada [18]. Na aprendizagem supervisionada, os dados fornecidos estão etiquetados, isto é, os dados já estão categorizados e, consequentemente, a saída de cada entrada é conhecida. Com esta a informação, a rede irá trabalhar os dados de entrada para obter os padrões presentes que permitam relacionar as variáveis extraídas com os valores de saída [18]. Por exemplo, no caso de uma tarefa de classificação de imagens, se houver duas categorias de dados 'A' e 'B', a rede irá aprender as caraterísticas das imagens que permitem dizer a qual categoria pertencem e usá-las para classificar imagens novas de acordo com estas caraterísticas aprendidas. Já na aprendizagem não-supervisionada, os dados não têm qualquer etiqueta, o que leva a rede a aprender os padrões escondidos nos dados e criar etiquetas próprias para mais tarde agrupar os dados de acordo com as semelhanças que

encontrou entre os dados [18]. Exemplificando com imagens outra vez, tendo um conjunto de imagens de cães e gatos que não estão identificadas como tal, a rede irá procurar pelas semelhanças nos padrões que extrai das imagens e agrupá-las de acordo com as suas conclusões, ficando num grupo imagens com cães e noutro imagens de gatos. Por último, há a aprendizagem por reforço. Este tipo de aprendizagem é bastante diferente dos outros dois, pois melhora o seu modelo através de um retorno de informação das iterações de treino anteriores, essencialmente realizando um processo de tentativa e erro. Em vez de etiquetar as saídas, um modelo com este tipo de aprendizagem atribui uma pontuação e retorna informação para a próxima iteração. Este processo pode ser encontrado nos veículos de condução autónoma. Quando este evita uma colisão, o algoritmo atribui uma pontuação positiva, fazendo com que o veículo continue a evitar colidir porque lhe foi dito que não colidir é bom. Por outro lado, se colidir, o algoritmo retorna uma pontuação negativa e o veículo irá tentar evitar colidir na próxima iteração [18].

1.4.2 Tipos de camada

Sendo camadas arranjos de neurónios, estas podem ter diferentes funções dependendo do seu tipo.

1.4.2.1 Camada de Densificação

A mais básica das camadas, os seus neurónios formam ligações a todos os neurónios das camadas adjacentes como se pode ver pela Figura 1.4. Em cada neurónio, irá ser realizada uma multiplicação dos dados de entrada por uma matriz de pesos seguido de uma soma do viés do neurónio. Para além disso, a Densificação altera as dimensões dos dados para um vetor unidimensional e é usada no final de muitas NN para compilar os dados de saída finais. Em tarefas de classificação é colocada no final da rede com um número de neurónios igual ao número de classes.

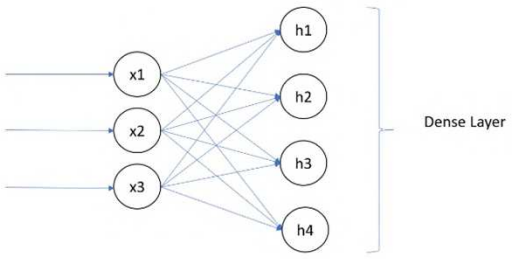


Figura 1.4: Representação esquemática de uma Camada de Densificação [19].

1.4.2.2 Camada Convolucional

Usada no processamento de imagem, esta camada é caraterizada pela operação de convolução (que lhe dá o nome) que é realizada de acordo com o esquema da Figura 1.5. Recorre a filtros deslizantes, pequenas grelhas de tamanho especificado pelo utilizador denominadas por *kernels*, para realizar esta operação. Cada filtro extrai caraterísticas específicas a partir dos dados de entrada, como padrões complexos, texturas ou bordas [20]. Aos conjuntos de dados extraídos das imagens dá-se o nome de mapas de caraterísticas e há um por cada filtro empregue pela camada. Estes mapas são depois empilhados para formar a saída da camada, sendo a dimensão da pilha (nº de filtros, x, y), onde x e y são a dimensão de cada mapa resultante da operação de convolução. Os seus valores dependem do tamanho da imagem de entrada, do tamanho do kernel e do deslocamento do mesmo. Uma última caraterística deste tipo de camada, é a sua organização hierárquica. As primeiras camadas extraem caraterísticas de baixo nível, como rebordos ou cantos e as camadas seguintes extraem caraterísticas como formas e mesmo objetos. Isto permite à rede aprender mais sobre as relações entre as caraterísticas de uma imagem [20].

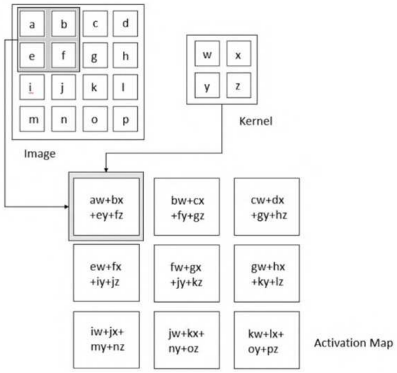


Figura 1.5: Operação de convolução [15]

1.4.2.3 Camada de Ativação

Usada para aplicar uma função de ativação a cada elemento do mapa de dados resultante da operação de camadas anteriores, introduzindo não-linearidade aos dados na rede [6]. Qualquer uma das funções de ativação apresentadas na tabela 1.1 pode ser usada aqui.

1.4.2.4 Camada de *Pooling*

Esta camada é usada para diminuir a dimensionalidade dos dados. Para isto, esta faz uma operação de *downsampling* na qual divide o mapa de caraterísticas de entrada em parcelas de tamanho especificado pelo utilizador e depois sumaria os dados a uma única entrada como se pode ver pela Figura 1.6. Há duas maneiras principais de fazer isto, a primeira, na qual a camada escolhe

o valor máximo de cada parcela do mapa de caraterísticas, chamada de *Max Pooling*, ou pode fazer a média dos valores da parcela, a *Average Pooling* [6]. Para além disso, existe também a *Global Pooling*, que realiza estas operações, não em parcelas, mas na totalidade do mapa de caraterísticas.

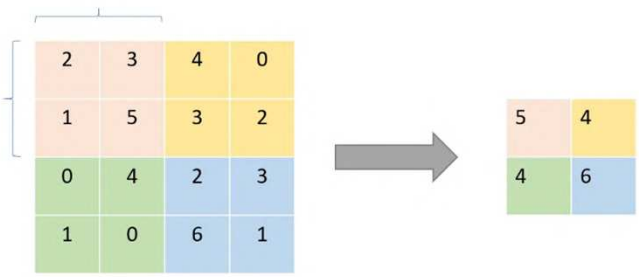


Figura 1.6: Camada *Max Pooling* [6].

1.4.2.5 Camada de de Normalização em *Batch*

Parte do conjunto de camadas de Normalização, esta camada realiza a sua Normalização em *batch* (conjunto de amostras agrupadas para reduzir os requisitos de memória) e serve para acelerar a velocidade de convergência da rede em que se insere. Atua por meio de uma subtração da média e da divisão pelo desvio padrão, dando origem a um *batch* de saída com média igual a 0 e desvio padrão igual a 1 [6]. Este processo está exemplificado na seguinte Figura 1.7.

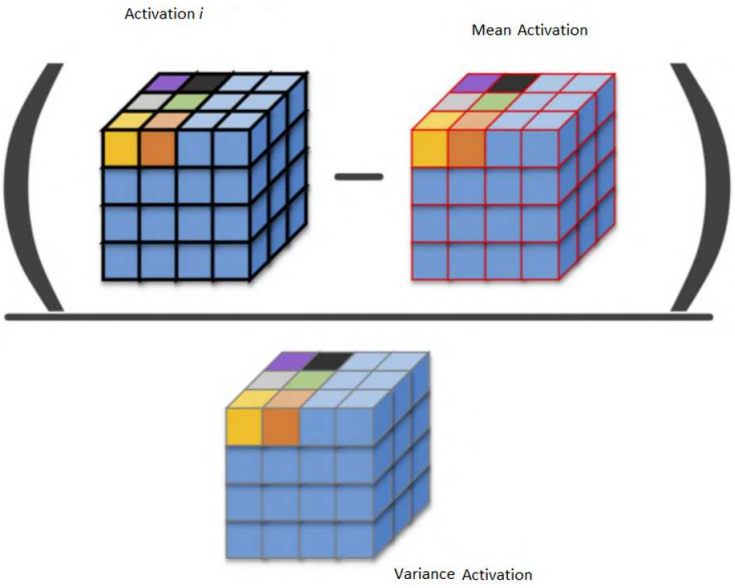


Figura 1.7: Representação esquemática da equação que rege a atuação de uma Normalização em *Batch* [6].

1.4.2.6 Camada de *Dropout*

É usada para reduzir o output de camadas anteriores para impedir o sobreajuste da rede. Faz isto reduzindo a 0 o valor de uma fração do output das camadas anteriores (como demonstrado na Figura 1.8), cujo rácio é fornecido pelo utilizador [6].

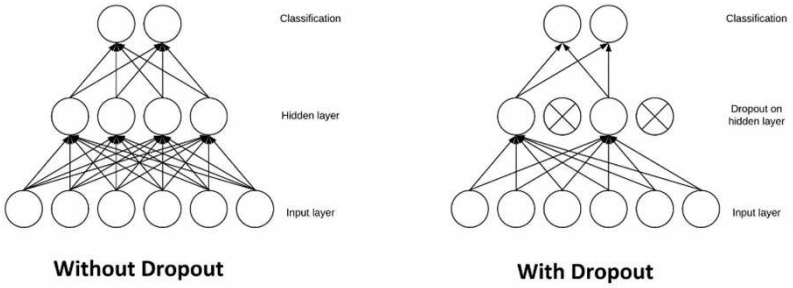


Figura 1.8: Diferença entre rede sem camada Dropout e rede com camada *Dropout* [21]

1.4.2.7 Camada de *Flatten*

Transforma os mapas de caraterísticas em vetores unidimensionais, preservando a ordem dos dados e o seu volume [22] como a Figura 1.9 mostra.

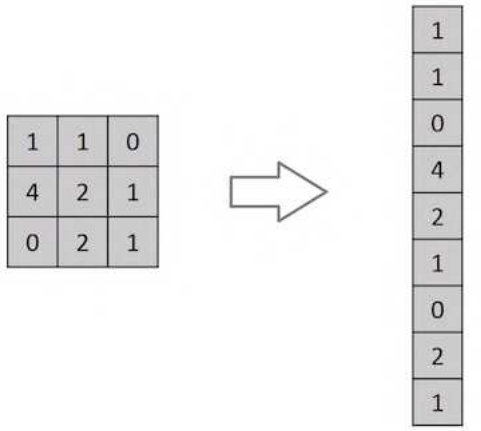


Figura 1.9: Exemplificação da atuação de uma camada *Flatten* [23]

1.4.2.8 Camada de *Reshape*

Altera as dimensões dos dados de entrada, sem alterar os dados ou a sua ordem. As dimensões da sua saída são especificadas pelo utilizador, podendo mesmo adicionar dimensões onde antes não existiam [24].

1.4.2.9 Camada de *Resizing*

Desenhada especificamente para imagens, esta camada é capaz de alterar as dimensões de uma imagem, mantendo a proporção entre a largura e altura intacta. Se necessário, esta camada é capaz de realizar interpolação para preencher as dimensões especificadas [25].

1.4.3 Tipos de Rede

Existe uma grande variedade de NN todas com caraterísticas próprias, mas passíveis de agrupar de acordo com as suas semelhanças. Como já foi referido, este trabalho foca-se nas DNN, pela sua capacidade de trabalhar problemas complexos com maior flexibilidade do modelo resultante para aplicações futuras. Disto isto, irá ser feita uma análise a uma seleção de NN importantes, sendo elas as *Feedforward Neural Networks* ou Redes Neuronais de Alimentação Direta (FNN), as *Recurrent Neural Networks* ou Redes Neuronais Recorrentes (RNN), as CNN e os *Autoencoders*.

1.4.3.1 Redes Neuronais de Alimentação Direta

As FNN são redes nas quais a informação flui apenas numa direção, da entrada para a saída, passando pelas camadas escondidas (assim chamadas, pois não são diretamente observáveis de fora) onde o grosso das computações da rede são realizadas. São geralmente usadas em reconhecimento de padrões em imagens, fala ou texto, análise de séries temporais, deteção de anomalias e muitas mais [15]. Um subtipo desta rede é a FNN com retropropagação. A diferença está no uso de um algoritmo de retropropagação que ajusta os pesos e limiares dos neurónios da rede, propagando o erro entre a saída prevista e a saída real para trás na rede, na direção da entrada [26]. A imagem seguinte (Figura 1.10) exemplifica este tipo de rede e a diferença entre os seus subtipos.

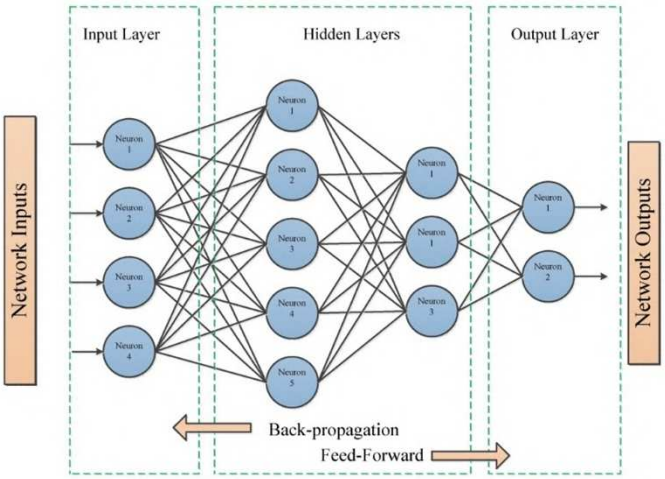


Figura 1.10: Estrutura de uma FNN com uma demonstração da diferença entre uma FNN convencional e uma FNN com retropropagação [26].

1.4.3.2 Redes Neuronais Recorrentes

Desenhadas para processar dados sequenciais, estas NN são particularmente úteis no tratamento de séries temporais, traduções automáticas, reconhecimento de fala e outras tarefas que requerem conhecimento do contexto das entradas da rede. Estas redes partilham os seus pesos ao longo do processo de treino através da manutenção de uma memória, ou estado, oculta. Esta memória oculta é recorrentemente atualizada e guarda a informação recolhida em cada passo para que possa ser utilizada no processamento de entradas atuais e futuras. Como tal, cada neurónio da rede recebe duas entradas, o tensor de entrada e o estado oculto do passo temporal anterior e devolve duas saídas, o tensor de saída e uma atualização do estado oculto [8] como está esquematizado na Figura 1.11.

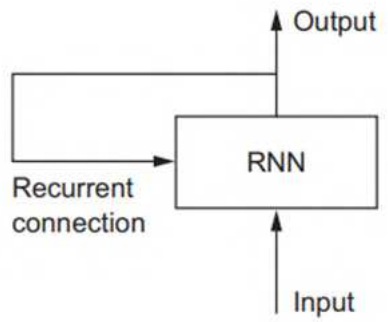


Figura 1.11: Esquematização de uma RNN [8].

Teoricamente, este tipo de rede consegue manter a informação dos passos temporais mais antigos, na prática isto não é possível, porque a rede sofre daquilo a que se chama de desvanecimento de gradiente. Este fenómeno acontece quando os gradientes de otimização se tornam demasiado pequenos, levando a atualizações de pesos com valores pequenos e a abrandamento do processo de aprendizagem com provado por *Bengio et al.* (1994) [27]. Como consequência tiveram de ser desenvolvidos novos algoritmos para ultrapassar esta dificuldade. Um exemplo é a RNN criada por *Hochreiter* e *Schmidhuber* (1997) [28] chamada de *Long Short-Term Memory* (LSTM). Esta recorre a uns mecanismos chamados de portas que permitem à rede controlar o fluxo de informação ao longo da sua estrutura. Este tipo de rede contém 4 componentes chave, sendo eles [8]:

- O estado da célula, que funciona como memória da informação relevante dos passos temporais anteriores e dura toda a sequência;
- A porta de entrada, que determina qual da informação atual é relevante para o estado oculto da rede;
- A porta de esquecimento, que determina qual a informação do estado da célula anterior tem de ser descartada;
- - E a porta de saída, que determina a relevância da saída da célula para o estado oculto corrente.

Isto é facilmente visualizado na Figura 1.12.

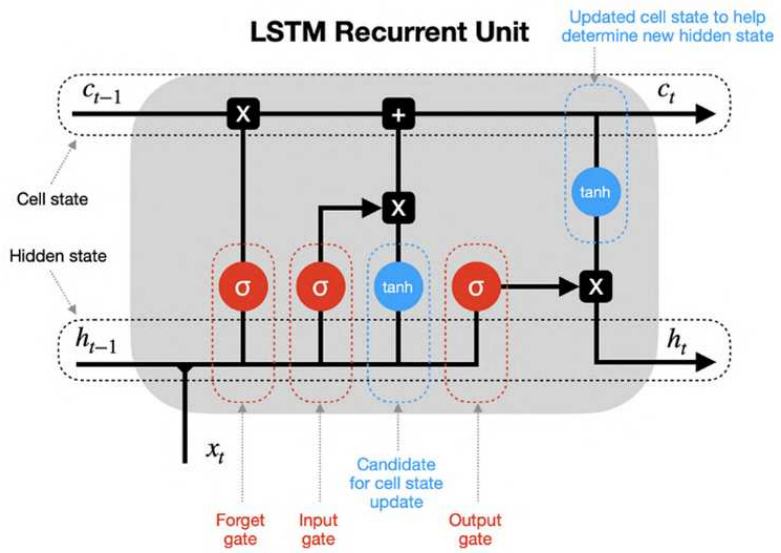


Figura 1.12: Representação de uma unidade de uma rede LSTM [29].

Outro exemplo de RNN, são as *Gated Recurrent Units* (GRU). Estas RNN são muito semelhantes às LSTM, mas onde as LSTM têm três portas e o estado de célula (Ct), as GRU apresentam apenas duas portas, a porta de *reset* e a porta de atualização, e combina o estado de célula com o estado oculta para simplificar a arquitetura da rede. A porta de *reset* determina o que é descartado do estado oculto anterior e a porta de atualização determina quanta da informação mais recente deve ser incorporada no estado oculto. Devido à sua simplicidade em comparação com as LSTM, as GRU requerem menos tempo de treino e capacidade computacional [30].

1.4.3.3 Redes Neuronais Convolucionais

As CNN são um algoritmo de DL que pretende imitar o córtex visual, e especializa-se no processamento de dados que apresentem uma topologia de grelha. O maior exemplo deste tipo de dados são as imagens, sendo estas um conjunto de pixeis organizados em grelha e com valores que denotam a cor e intensidade dos mesmos. Dito isto, as CNN são bastante úteis no processamento de imagem, seja para as classificar, detetar objetos contidos nelas ou para as segmentar [20]. Também são capazes de legendar imagens em conjunto com RNN [31]. Tipicamente, uma CNN é composta por camadas de Convolução, Pooling e Densificação, podendo também conter camadas de Normalização, *Dropout* ou *Flatten* [20] [32] como demonstrado pela Figura 1.13.

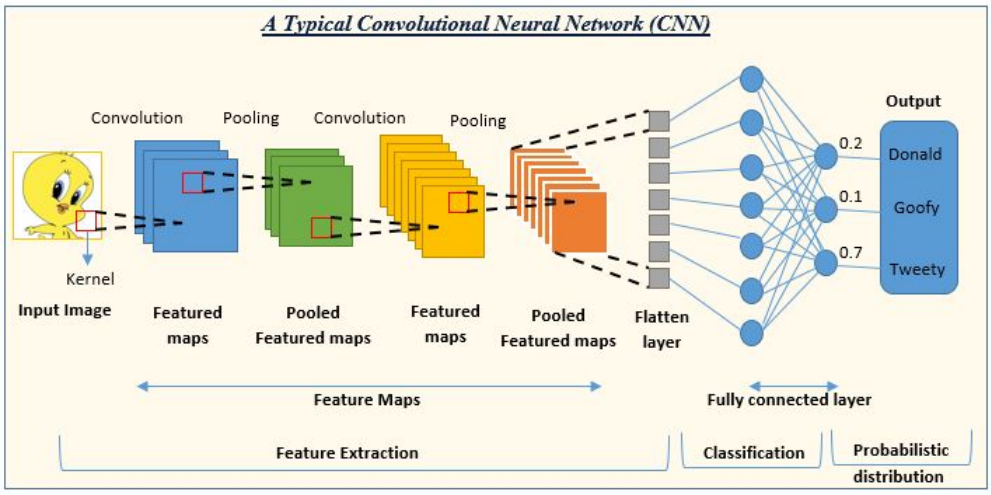


Figura 1.13: Representação de uma estrutura típica de uma CNN de classificação [32].

1.4.3.4 *Autoencoders*

Autoencoders são compostos por duas partes, um codificador e um descodificador. O codificador típico é composto for várias camadas ocultas hierarquizadas que irão comprimir a entrada da rede essencialmente diminuindo a dimensionalidade da entrada e originando aquilo a que se chama de representação latente. O descodificador recebe esta estrutura e reverte o processo, reconstruindo os dados de entrada da rede. Como a representação latente possui uma dimensionalidade diminuída em relação à representação original, há perda de informação. Esta perda é controlada e permite à rede ignorar informação não relevante e focar-se nos dados mais importantes [20]. Há vários tipos de

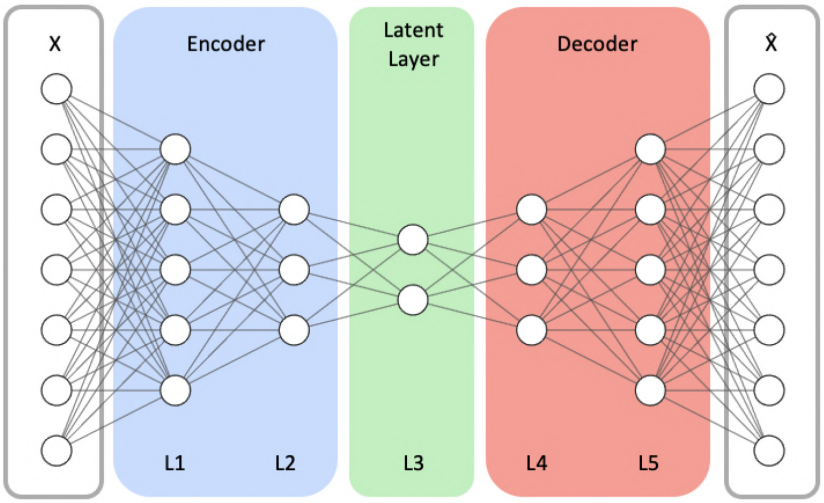


Figura 1.14: Exemplo de *Autoencoder* Empilhado [33].

Autoencoder e o primeiro que se vai analisar é o *Autoencoder* empilhado (Figura 1.14). Este tipo de rede é bastante simples, contendo apenas um número maior de camadas ocultas. Quantas mais tiver, mais complexos podem ser os dados fornecidos. No entanto, se esta rede for demasiado potente, irá fazer a codificação e descodificação tão bem, que reconstrui por inteiro os dados originais, impedindo que aprenda caraterísticas úteis [20].

O segundo tipo de *Autoencoder* a ser analisado é *Autoencoder* Convolucional (Figura 1.15), que, como as CNN, é especializado no tratamento de imagem. Este recorre a camadas de Convolução em vez de Densificação para aprender as caraterísticas das entradas. O codificador é em tudo uma típica CNN, reduz a largura e altura da entrada e aumenta a profundidade. O descodificador, recorre a camadas de Convolução Transpostas ou, de Deconvolução, para reconstruir a imagem de entrada [20].

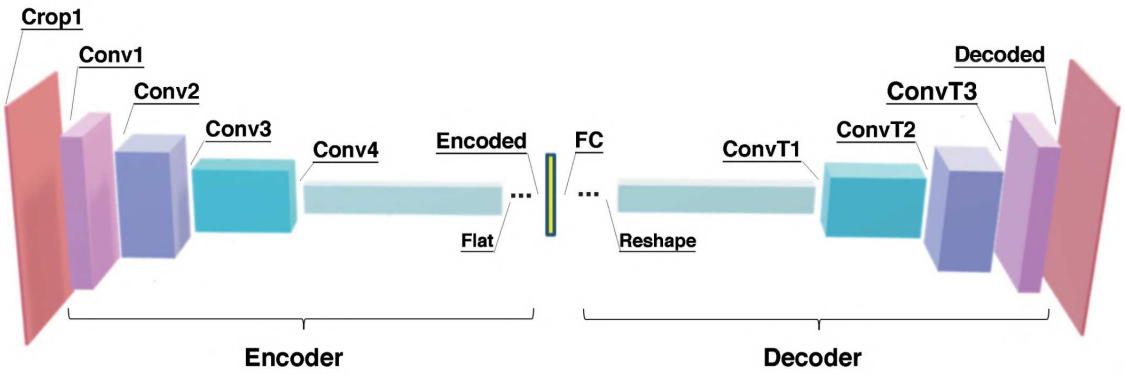


Figura 1.15: Representação exemplo de um *Autoencoder* Convolucional [34].

O terceiro e último tipo de *Autoencoder* em análise é o *Autoencoder* Recorrente (Figura 1.16). Este está focado para trabalhar com sequências e recorre a um par de RNN como codificador e descodificador. A primeira reduz a sequência de dados para um vetor enquanto a segunda usa esse mesmo vetor para reconstruir essa mesma sequência [20].

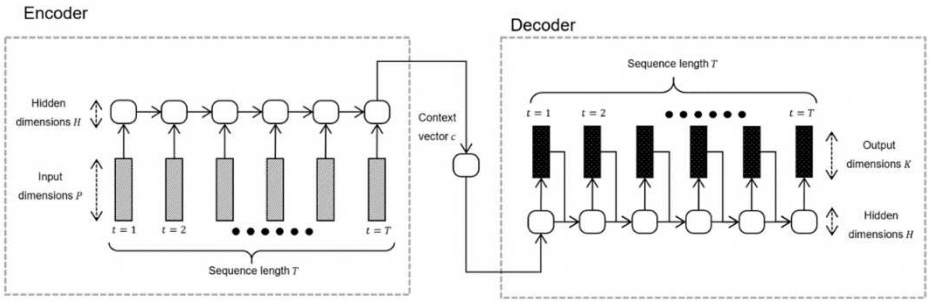


Figura 1.16: Representação de um *Autoencoder* Recorrente [35].

1.4.4 Métricas de avaliação

Para avaliar e comparar os resultados obtidos por uma NN, são necessárias métricas objetivas. Para tal fora desenvolvidas vários algoritmos e equações para o efeito. Cada tipo de tarefa para a qual se use NN tem métricas próprias. Em tarefas de classificação (como aquela que se realiza neste trabalho), existem diversas métricas, mas as mais interessantes para este trabalho são as seguintes 4:

1. Exatidão ou *Positive Predictive Value* (PPV)
2. *Recall*, Sensibilidade ou *True Positive Rate* (TPR)
3. *F1−score* (F1)
4. *Accuracy* ou Precisão (ACC)

Estas métricas estão dependentes de quatro conceitos, TP, os *True Positives* ou Positivos Reais, TN, os *True Negative* ou Negativos Reais, FP, os *False Positives* ou Falso Positivos e os FN, os *False Negatives* ou Falso Negativos, demonstradas na matriz de confusão seguinte (Figura 1.17):

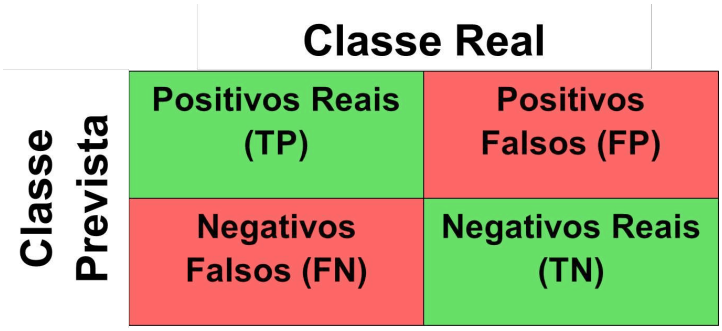


Figura 1.17: Matriz de confusão.

1.4.4.1 Exatidão

A Exatidão é a proporção de Positivos Previstos que são de facto Positivos Reais, essencialmente diz quantas das previsões positivas, estão corretas [36]. Calcula-se pela fórmula:

$$PPV = \frac{TP}{TP+FP} \tag{1.1}$$

Um modelo com alta Exatidão apresentará poucos Falsos Positivos.

1.4.4.2 *Recall*

Corresponde à proporção de Positivos Reais que são previstos como positivos, isto é, das amostras positivas, quantas foram identificadas corretamente como tal [36] e é calculada pela fórmula:

$$TPR = \frac{TP}{TP+FN} \tag{1.2}$$

Um valor elevado desta métrica significa poucos Falsos Negativos.

1.4.4.3 $F1-score$

Esta métrica resulta da média harmónica da Exatidão com a $Recall$ e quantifica a capacidade de um modelo para equilibrar estas duas métricas, calculando-se da seguinte maneira [36]:

$$F1 = 2 * \frac{PPV * TPR}{PPV + TPR} \tag{1.3}$$

Altos valores de F1-score, indicam que o modelo equilibra bem a Exatidão e a $Recall$.

1.4.4.4 Precisão

Por fim, a Precisão é o rácio entre os Resultados Reais, Positivos ou Negativos, e o total de resultados [36].

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{1.4}$$

Quanto mais alto o valor da Precisão, maior o número Positivos e Negativos Reais corretamente classificados. No entanto, esta é uma métrica tendenciosa, uma vez que se houver um desequilíbrio na quantidade de dados em cada classe, o resultado poderá não refletir a capacidade real do modelo ser adulterado tanto para cima como para baixo. Por exemplo, num exercício de classificação binária, se uma das classes tiver muitos mais dados do que a outra e a rede for capaz de classificá-la melhor do que a classe mais diminuta, o modelo irá apresentar uma Precisão elevada que não contempla a classificação da outra classe.

Já para tarefas relacionadas com som, como a atenuação de eco acústica tratada neste trabalho, criou-se uma série de métricas para avaliar a qualidade do som. Das mais importantes e a que se usa neste trabalho, é a PESQ ou *Perceptual Evaluation of Speech Quality*, Esta métrica é um algoritmo padronizado pela União Internacional de Telecomunicações (ITU) muito usada em trabalhos de IA relacionados com a fala e pretende avaliar a qualidade percecionada da fala [37]. O algoritmo simula a resposta do sistema auditivo humano à perceção do sinal processado e faz isto com uma comparação entre um sinal de referência e o sinal processado e devolve uma pontuação calculada entre -0,5 e 4,5 [38]. Quanto maior a pontuação, melhor é a qualidade do sinal. É de notar que esta métrica deve ser sempre confrontada com a perceção humana do sinal em causa.

1.5 Fala

Pode-se definir a fala como um processo complexo que engloba três etapas principais: produção de ondas sonoras pelas cordas vocais, a transmissão das mesmas pelo meio e a perceção das ondas sonoras pelo ouvinte [39]. Na primeira etapa, as cordas vocais vibram criando diferenciais de pressão no meio, neste caso ar, dando origem a ondas sonoras. Estas ondas serão moldadas em sons reconhecíveis como fala pela articulação do trato vocal (boca, nariz, garganta, língua, lábios e maxilar) e ao mesmo tempo há uma ressonância nas várias cavidades do trato vocal que amplifica certas frequências que dão à voz de cada pessoa os seus traços únicos [39]. De seguida, as ondas irão propagar-se pelo ar até serem captadas pelos ouvidos do ouvinte. Estes irão converter as ondas sonoras em sinais elétricos e transmiti-los para o cérebro, um processo denominado de transdução [39]. Por último, o cérebro irá interpretar os sinais elétricos permitindo a perceção da fala pelo ouvinte [39]. A seguinte imagem esquematiza o processo (Figura 1.18).

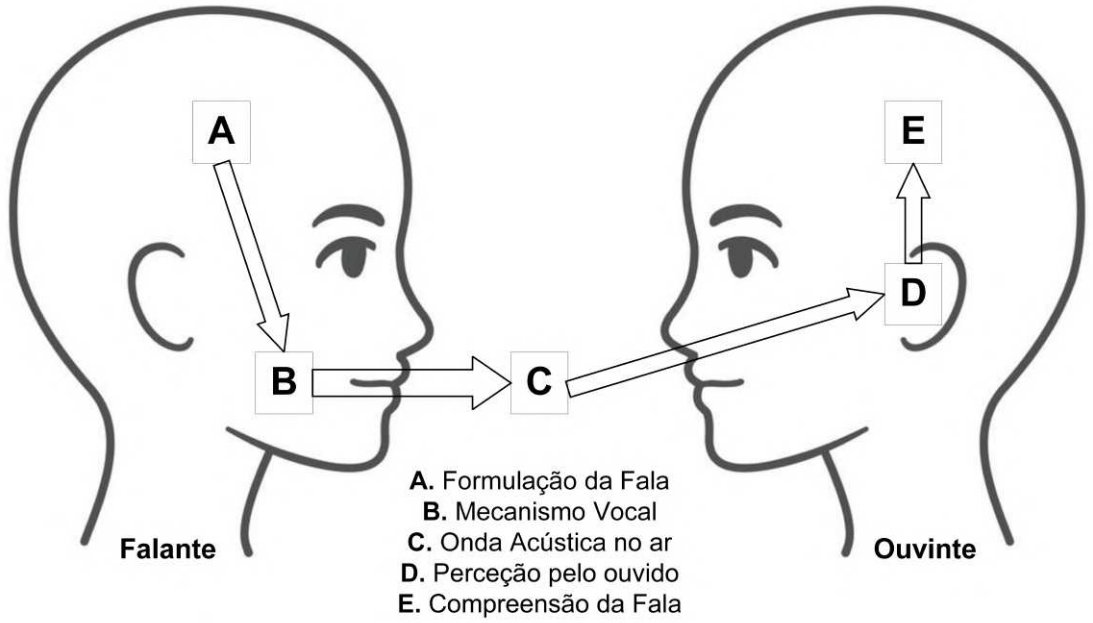


Figura 1.18: Processo da fala. A. Formulação da fala; B. Vocalização; C. Propagação da onda sonora; D. Perceção pelo ouvido; E. Compreensão da fala [40].

1.5.1 Sinal da Fala

Dado o dinamismo de todo o aparato produtor da fala, o sinal biomédico gerado não será constante, ou seja, o sinal de fala é não-estacionário. Regra geral, as frequências que o compõem estão entre os 30 e 10000Hz, sendo que a maior parte da energia espetral está nas frequências entre 200 e 3500Hz [41]. A Figura 1.19 mostra 3 exemplos de sinais de fala.

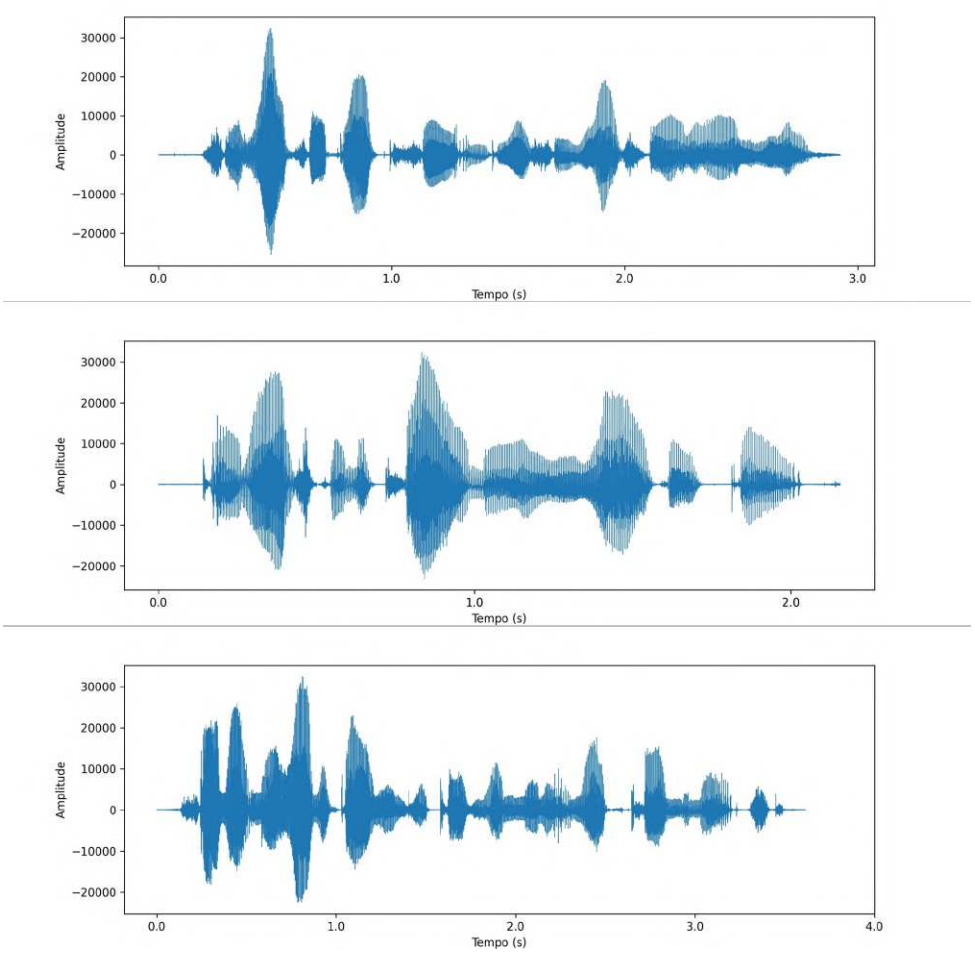


Figura 1.19: Exemplos de sinais de fala retirados da base de dados TIMIT.

1.5.2 Processamento de Sinal

1.5.2.1 Transformada de *Fourier*

A representação normal de um sinal é feita no domínio temporal, registando uma grandeza ao longo do tempo. No entanto, esta não é a única maneira de representar um sinal. Também é possível representá-lo no domínio frequencial, isto é, faz-se uma análise espectral a partir dos componentes de frequência calculados através do produto de interno do sinal com funções sinusoidais infinitas. O espetro resultante é um sumário global do sinal no domínio frequencial.

Uma das ferramentas mais usadas para obter o espetro de um sinal, é a Transformada de *Fourier* (FT). Esta é uma ferramenta matemática capaz de decompor funções complexas, sinais incluídos, nas suas frequências constituintes com as suas respetivas amplitudes, sendo que a representação é feita no domínio frequencial e não no domínio temporal. O uso da inversa da FT, permite reconstruir a função decomposta e voltar ao domínio temporal [42].

Dada a natureza do processo de amostragem de um sinal, é necessário usar a forma discreta da FT, a Transformada Discreta de *Fourier* (DTFT - *Discrete-Time Fourier Transform*). Esta irá transformar um sinal discreto, com frequência de amostragem fs, $x[n] = x(n * Ts)$ no qual $n = 0, \ldots, N-1$, sendo N o número total de amostras, num espetro $X[k]$ em que $k = 0, ..., N-1$, o mesmo N de $x[n]$. A partir daqui é possível concluir que $x[n]$ é um sinal periódico de período fundamental N e frequência fundamental $\Omega = 2 * \frac{\pi}{N}$ e que k é uma frequência sinusoidal associada a $X[k]$ e que a frequência discreta equivalente é $\Omega = k * \Omega_0$. A partir disto obtêm-se o seguinte par complementar de fórmulas usadas para calcular a DTFT [42]:

$$x[n] = \frac{1}{N} * \sum_{k=0}^{N-1} X[k] * e^{j*k*\Omega_0*n}, n = 0, ..., N-1 \tag{1.5}$$

$$X[k] = \frac{1}{N} * \sum_{k=0}^{N-1} x[n] * e^{-j*k*\Omega_0*n}, k = 0, ..., N-1 \tag{1.6}$$

Com estas fórmulas é possível comprovar que é possível calcular um sinal e o seu espetro a partir um do outro, mostrando a sua complementaridade em pleno. Será de notar que esta transformada é especialmente usada em computadores, uma vez que a discretização dos sinais é um dos requisitos para que um computador consiga processar um sinal contínuo.

1.5.2.2 Transformada de *Fourier* de Curta Duração

Um dos grandes problemas da FT é a sua incapacidade de lidar com sinais não-estacionários, isto é, sinais que variam ao longo do tempo, pois os espetros de frequência resultantes da transformada não mostram as mudanças que estes sinais sofrem no domínio temporal. Para lidar com este problema é usada a Transformada de *Fourier* de Curta Duração (STFT). Esta transformada faz uso de um processo de janelamento deslizante no qual uma função janela (função cuja amplitude aproxima-se de zero nas suas bordas) de comprimento constante é usada para delimitar secções de um sinal que irão ser transformadas, passando a sua representação do domínio temporal para o domínio frequencial. Esta janela será então deslocada ao longo do sinal até que o seu espetro esteja completo. Uma vez que este é construído janela a janela, irá apresentar maior detalhe que o espetro resultante da DFT uma vez que este é criado a partir do sinal inteiro resultando num espetro global [43].

Matematicamente, a função janela $w[n]$ pressupõe que o segmento de sinal que delimita é estacionário e que é não-nula no intervalo $[0, L-1]$ onde L é o tamanho da janela. Para além disto, e referente à fórmula apresentada a seguir, N é o número de amostras contidas na janela e $N_T = \frac{N}{L}$ [43]:

$$X_W[m,k] = \sum_{n=0}^{N-1} x[n] * w[n-m] * e^{-j*k*\Omega_0*n}, k = 0, ..., L-1; m = 0, \ldots, N_T - 1 \tag{1.7}$$

Existe uma grande variedade de funções janela, como a janela retangular, a *Hamming*, a *Hanning* (*Hann*), a *Blackman* e muitas outras. Cada uma tem caraterísticas próprias e aplicações para as quais

foram desenvolvidas. Estas funções servem para atenuar as perdas de energia da frequência verdadeira do sinal para as frequências adjacentes que não estão presentes no sinal, permitem melhorar a resolução temporal ou frequencial da STFT e ainda ajudam a diminuir a complexidade computacional no cálculo da STFT [44].

Neste trabalho e em ambas as tarefas, foi usada a função janela Hann (Figura 1.20), uma função pertencente à família de funções janela Hamming [44] devido à sua capacidade de reduzir o vazamento espetral sem comprometer demasiado a resolução frequencial.

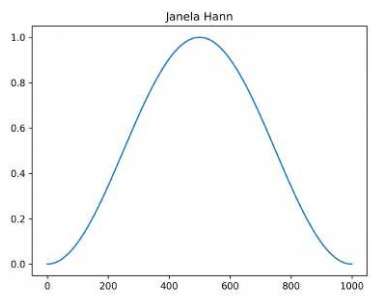


Figura 1.20: Representação gráfica da função janela *Hann* gerada em Python.

1.5.2.3 Espetrogramas

Um espetrograma é nada mais, nada menos do que a representação gráfica do espetro de um sinal calculado a partir do quadrado da magnitude da STFT desse mesmo sinal. É uma ferramenta muito usada na análise de áudio, música e, mais importante para este trabalho, fala [44] (a Figura 1.21 mostra um exemplo de uma espetrograma calculado a partir de um sinal retirado da BD TIMIT).

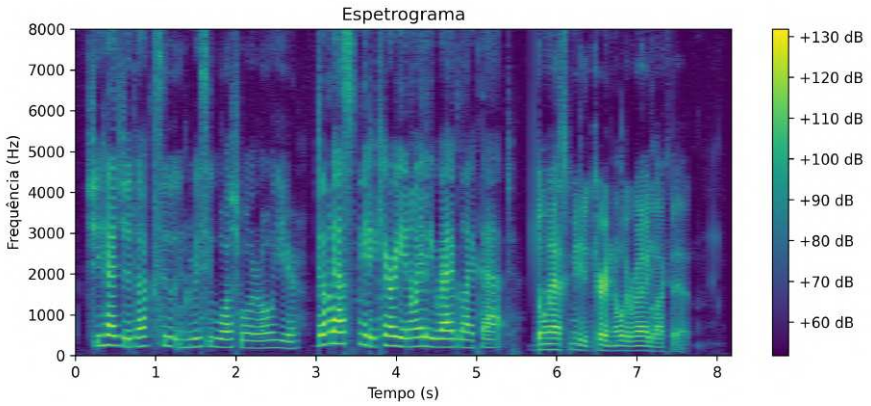


Figura 1.21: Espetrograma gerado em Python de um sinal de áudio retirado da BD TIMIT.

1.5.3 Problemática em estudo

A importância e valor do estudo de técnicas de IA para trabalhar com som têm vindo a aumentar ao longo dos anos. Este é muito importante, sendo o veículo de comunicação mais usado, permitindo passar conceitos de pessoa para pessoa que de outro modo seria difícil de fazer. Isto revela-se nas tecnologias desenhadas para aumentar a capacidade de comunicação do ser humano como os telefones, transmissões de rádio, teleconferências e outras todas aplicáveis em vários setores da sociedade. Por exemplo, no setor da saúde, com consultas remotas, por telemóvel ou teleconferência, e nos aparelhos auditivos, que permitem recuperar a capacidades auditivas perdidas por razões várias. No entanto, estas tecnologias não são perfeitas, pois durante a aquisição do som, também são captados distúrbios sonoros como o ruído (variações indesejadas ou aleatórias que perturbam o som pretendido), o eco (a reflexão do som numa superfície ou obstrução que retorna ao ouvido do ouvinte) ou excesso de reverberação (a persistência do som mesmo depois da fonte sonora se silenciar) [45]. Como tal, é necessário desenvolver sistemas capazes de identificar e atenuar estes distúrbios.

1.5.4 Estado da arte

O estudo de DL tem visto repetidos avanços ao longo do tempo e cada vez mais isto se verifica. Na área de classificação de áudio para melhorar perceção do ambiente circundante de um veículo autónomo [46] onde é usada uma CNN para classificar diferentes classes áudio, como por exemplo o som de uma sirene de ambulância. Outro trabalho, também neste tópico de classificação de áudio, faz um estudo doo uso de modelos de CNN pré-treinados para uso em classificação de imagem como *ImageNet* ou *DenseNet*, em tarefas de classificação de espetrogramas de áudio, um processo denominado de aprendizagem por transferência, onde uma rede treinada numa tarefa é usada noutra tarefa diferente, mas semelhante [47]. Neste caso, os autores pretendem usar como entradas espetrogramas de áudio, que podem ser considerados de imagens de áudio. Já na área de atenuação de eco, há uma grande variedade de métodos aplicados, como em Zhang e Wang (2018) [48] onde criaram um modelo de NN do tipo BLSTM, contendo duas LSTMs unidirecionais que processam a entrada em direções opostas e cujo objetivo final é criar uma máscara, um vetor de coeficientes que, quando multiplicado ponto a ponto, irá atenuar as frequências indesejadas no sinal. Já Fazel et al. (2019) [49], recorrem a duas GRU, uma para estimar o eco e outra que estima o sinal limpo com base no eco estimado. Zhang e Wang (2021) [50], continuaram a sua investigação e criaram uma Rede Convolucional Recorrente que combina uma rede convolucional sobre a forma de *Autoencoder* com uma LSTM e cujo objetivo é atenuar eco captado não por apenas um microfone, mas por vários e depois agregar os resultados para obter um sinal sonoro o mais próximo possível do original. Como os aparelhos modernos de comunicação estão equipados com mais do que um microfone ou altifalante, este método apresenta-se como mais realista, por estar adaptado a esta situação.

CAPÍTULO 2

Metodologia

Todo o trabalho de código foi realizado em *Python* 3.10.8, num processador *AMD Ryzen 9 5900HX*, RAM de 16GB e com a placa gráfica *NVIDIA GeForce RTX 3070*. A linguagem de programação *Python* apresenta várias bibliotecas, uma delas e talvez a mais importante para este trabalho, é a biblioteca *Tensor Flow*, uma biblioteca desenhada para trabalhar com IA.

2.1 Evolução do Método

Antes de descrever o método usado neste trabalho de tese de mestrado, será feita uma descrição da evolução do método usado ao longo do trabalho e, para melhor comparar as várias etapas, será usado o mesmo sinal de fala (e o seu espetrograma) para demonstrar as diferenças. Nas etapas onde este sinal é aumentado através de períodos de silêncio ou concatenação de outros sinais do mesmo tipo, será feita uma representação dos mesmos antes de apresentar os resultados das etapas em questão. Também serão fornecidos links de *Google Drive* que contêm os ficheiros de áudio para cada etapa.

A seguinte lista descreve as várias tentativas feitas para obter resultados neste trabalho e o anexo 1 contém as imagens referentes às tentativas mais importantes (Tentativas 3, 4, 7 e 8. A 9 e 10 são estudadas com maior profundidade mais à frente no documento):

1. Inicialmente, o trabalho começou a ser feito em *MatLab*. No processamento dos sinais de fala, apenas se usou um ficheiro por instância em vez de encadear 4 ficheiros diferentes num único sinal. Cada sinal foi convolucionado com os RIR fornecidos para originar os dados comprometidos e foram criadas 9 CNN para trabalhar os sinais. Estas foram aplicadas de quatro maneiras diferentes.

- Primeiro, tentou-se fazer com que as redes tivessem como saída os sinais reconstruidos, sem o eco. Para tal, usou-se como entrada os sinais comprometidos e os sinais originais como alvo. Nesta tentativa os sinais resultantes eram praticamente iguais aos sinais comprometidos, com alguma distorção.
- Segundo, tentou-se criar uma rede capaz de obter o sinal comprometido a partir do sinal original. Os sinais originais foram a entrada e os sinais comprometidos o alvo. Para além de não resultar, este exercício não tinha utilidade para a tarefa, sendo impossível usar os resultados para de algum modo retirar o eco dos sinais comprometidos.
- Em terceiro, o objetivo foi obter o RIR a partir dos sinais comprometidos e usar este para fazer uma desconvolução e assim obter o sinal original. Não foi possível obter resultados com este método, sendo que qualquer tentativa de desconvolucionar o sinal corrompido resultou em sinais de ruído branco, isto é, som que contém todas as frequências audíveis a intensidades aproximadas, aquilo a que se chama de estática coloquialmente.
- Houve uma tentativa de fazer o processo com os espetrogramas dos sinais da mesma maneira que se tentou com os sinais integrais, com o espetrograma do sinal comprometido como alvo da rede e o espetrograma do sinal original na entrada, mas também não se obtiveram resultados.
- Por último, tentou-se recriar o trabalho de *Zhang e Wang* (2018) [48] e obter uma IRM (*Ideal Ratio Mask*), que, quando multiplicada pelo sinal corrompido, iria abafar o eco. Não foi possível obter resultados, sendo esta também a última tentativa em *MatLab*.

2. Dadas as dificuldades que houve em criar redes coerentes e funcionais em *MatLab*, a incapacidade de obter resultados tangíveis e os requisitos de memória impossíveis de cumprir, foi decidido mudar de linguagem de programação para *Python*. Aqui foi necessário aprender uma linguagem de programação de raiz, desde os básicos de trabalhar em *Python*, como analisar dados com esta linguagem, até como criar e trabalhar NN em *Python*. Isto implicou pesquisar as várias bibliotecas de *Python* que contêm as funções necessárias para trabalhar os sinais, criar as redes e apresentar os dados. Entre todas as bibliotecas, as mais importante foram a biblioteca *Numpy*, *TensorFlow* e *Keras*.

3. Durante o processo de aprendizagem de *Python* e de *Deep Learning*, tocou-se no tópico de classificação. Daqui retirou-se o modelo de classificação do tutorial de reconhecimento de áudio simples do site *TensorFlow* [51]. Depois de comprovada a sua utilidade no reconhecimento de áudio, tentou-se adaptar este modelo para a tarefa de atenuação. Isto foi feito retirando as camadas de *Dropout*, para manter o máximo volume de dados possível e evitar perdas desnecessárias, e a camada de *Flatten*, sendo esta desnecessária, visto que a entrada que foi usada para obter o RIR, como na 3ª tentativa em *MatLab*, já é um vetor unidimensional (a Figura A.3 representa um RIR de referência). Também se tentou usar os espetrogramas dos

sinais com eco para obter espetrogramas limpos (o *kernel* das camadas de Convolução usado foi de tamanho [1, 1], o que permite manter o tamanho da entrada da camada aumentando o tempo de processamento) mas, mais uma vez, não se obteve resultados satisfatórios em ambos os casos. O sinal alvo usado está exemplificado na Figura A.1(`https://drive.google.com/file/d/1C9fhr0GcCeoVmholvx_PrAgRrXlBmOj4/view?usp=drive_link`) e o sinal de entrada está representado na Figura A.2 (`https://drive.google.com/file/d/1rl2Rcw8blUcWPgF0FBXlcUuHbukJfHJQ/view?usp=drive_link`). No primeiro caso a rede não devolveu um sinal como o que está demonstrado na Figura A.3, mas sim uma única amostra, e no segundo o sinal de fala reconstruido saiu ainda mais distorcido do que aquilo que já estava com a adição do eco (Figura A.4) (`https://drive.google.com/file/d/13Hp4x4bONpYI4VzqLQ09BI5R7jNYI6tt/view?usp=drive_link`).

4. Aqui mudou-se o processamento dos sinais ao adicionar períodos de silêncio antes e depois e tentou-se usar CNNs para atenuar o eco (um exemplo de um sinal de fala alvo está representado na Figura A.5 (`https://drive.google.com/file/d/1LhlmbvpoVELoFRn6wQGHffrG2jip43tG/view?usp=drive_link`) e o de entrada na Figura A.6 (`https://drive.google.com/file/d/1pFc2dp5slX6_07PICgX9VMTM156LXzBJ/view?usp=drive_link`)). A ideia era reduzir o tamanho adicional de cada sinal de fala ao mesmo tempo que se atenuava o eco.

   - Primeiro tentou-se mais uma vez trabalhar sobre os sinais diretamente. As redes seguiam a estrutura comum das CNN, com Convoluções, Ativações, Normalizações e camadas de *Pooling* (a representação do resultado final está na Figura A.7 (`https://drive.google.com/file/d/1BKybOXH6ebEGXRUXWe_tp-eLsd7bsOK2/view?usp=drive_link`)).
   - Depois tentou-se trabalhar sobre os espetrogramas dos sinais adaptando as redes já usadas para o efeito, sendo o resultado obtido exemplificado na Figura A.8 (`https://drive.google.com/file/d/1r-OvZCV2LoYRg243yNM8py9NcHPp5YDE/view?usp=drive_link`).

   Em ambos os casos não houve resultados satisfatórios.

5. Houve uma tentativa de usar DNNs, mas estas não eram adequadas para atenuar eco, dando erros de construção de rede ou então requeriam mais memória RAM do que aquela que o hardware tinha disponível.

6. Também se tentou usar escalogramas, outro tipo de representação no domínio frequencial calculados a partir da transformada de *Wavelet* e equivalentes a espetrogramas, mas não foi possível introduzir-los nas redes existentes, dando erros de entrada na rede. Tentativas posteriores em redes mais recentes também não foram bem sucedidas.

7. Dadas todas as dificuldades até aqui, decidiu-se construir as redes passo a passo, isto é, começar por passar um sinal ou espetrograma pela rede sem que esta fizesse grandes alterações, comprovando assim que é possível obter um audio inteligível após inserção numa NN. O objetivo final era obter um sinal ou espetrograma de tamanho semelhante ao de entrada (`https://drive.google.com/file/d/1pAXrhkBkLu7jxenT9RzIED7dvAceMil6/view?usp=drive_link`). Por isso criou-se uma rede com duas camadas, uma *Flatten* e uma *Reshape*. Essencialmente, a entrada é achatada e depois retornada à sua forma original. Aqui houve sucesso, o sinal de fala reconstruido apresentou grandes semelhanças ao sinal original (sinal de fala alvo - `https://drive.google.com/file/d/1xAFt2-Ms4KzGTBFW8OLTknU$_$sAYjOayS/view?usp=drive_link`), com uma pequena distorção no som já esperada do mesmo devido à sua passagem pela rede e pela transformação de sinal para espetrograma e depois o calculo do seu inverso (voltando à escala temporal) como se pode ver por comparação da Figura A.5 (referência) com a Figura A.9 (`https://drive.google.com/file/d/1akW2s_5TsLW1ADA2579HT2Hm221Lyxyi/view?usp=drive_link`).

8. Os próximos modelos foram de construção simples, tendo apenas camadas de Convolução, Ativação, Normalização e uma Densificação, sendo que o sinais de fala de entrada (`https://drive.google.com/file/d/1G618MPznpmAwvTWCddW_pPckckp6ve5y/view?usp=drive_link`) e alvo (`https://drive.google.com/file/d/1EW470vvgLOml735PJjBGN1fSp6Ci9LNX/view?usp=drive_link)` são os mesmos que foram utilizados na tentativa 4).

    - O primeiro destes modelos foi construído com 3 camadas, uma Convolução, uma Ativação e uma Densificação (Figura A.10) (`https://drive.google.com/file/d/1hkNTunj3SuNpl81R-ll93z_1C_GVmsqk/view?usp=drive_link`).
    - O segundo modelo acrescentou uma Normalização em *Batch* a seguir à camada de Ativação e antes da Densificação Figura A.11) (`https://drive.google.com/file/d/1YRy110Vmda6Rt3MbY4Yu_rQPyxcdG1tW/view?usp=drive_link`).
    - O terceiro e último modelo repete a combinação de Convolução, Ativação e Normalização descrita no modelo anterior, ficando assim com 7 camadas no total Figura A.12) (`https://drive.google.com/file/d/1Nkmw6VHyWGUdBHKfsJhC0t4OwpOJnsBv/view?usp=drive_link`).

    Os resultados obtidos a partir destes modelos não se aproximam do que era desejado em nenhum dos 3 casos. Dito isto, foi necessário mudar de estratégia mais uma vez.

9. Como as CNN puras não estavam a resultar, foi necessário encontrar outro tipo de rede. Feita nova pesquisa, chegou-se aos *Autoencoders*. Este tipo de rede codifica e descodifica os dados inseridos, mantendo o tamanho dos dados de entrada à sua saída da rede. Dado o seu uso na remoção de ruído no exemplo *"Intro to Autoencoders"* da *TensorFlow*, teorizou-se que seria

possível usar este tipo de rede para atenuar eco. Antes de começar a desenhar *Autoencoders* próprios, testou-se as redes fornecidas pelo tutorial nas bases de dados de imagem para as quais foram trabalhadas com o intuito de aprender mais sobre o seu funcionamento.

10. Para garantir que as redes tinham dados suficientes para poderem atuar, aumentou-se o número de amostras de cada sinal. Isto é feito concatenando 3 sinais escolhidos ao acaso a partir da mesma pasta de ficheiros a cada sinal individual, mantendo o número de ficheiros de áudio, mas aumentando o tamanho de todos.

11. Tendo os dados preparados, passou-se à criação das NN. Foram criados 4 *Autoencoders*, treinados e testados de acordo com a metodologia descrita posteriormente.

2.2 Base de dados

Neste trabalho foi usada a TIMIT, uma base de dados pertencente à agência americana DARPA desenvolvida para uso na aquisição de conhecimento acústico-fonético e no desenvolvimento de sistemas de reconhecimento automático de fala. Esta contém 6300 sinais, gravados por 630 interlocutores masculinos e femininos (10 sinais cada um) que podem ser agrupados pelos seus sotaques que originam de 8 regiões americanas diferentes. Destes sinais de fala, foram escolhidas as 4 primeiras regiões o que totaliza 2580 sinais para treino e 950 para teste.

Para além de usar esta base de dados, também foram usados 5 sinais RIR fornecidos pelos orientadores desta tese.

Tanto os sinais TIMIT como os sinais RIR foram amostrados a 16kHZ.

2.3 Pré-Processamento

Antes de começar a trabalhar com os sinais de fala, é necessário criar pastas para os guardar para uso nas tarefas propostas. A sua organização será feita como apresentado no esquema seguinte (Figura 2.1):

Após descarregar os sinais de fala através de *scripts* escritos em *Jupyter*, uma ferramenta de programação em *Python*, estes são convertidos para o formato *wav* padrão para permitir trabalhá-los e ouvi-los no ambiente *Visual Studio Code* (VS Code). Como os ficheiros de áudio fornecidos são de pouca duração, serão concatenados 4 sinais de fala da mesma pasta escolhidos ao acaso. Isto será feito para todos os sinais de fala, mantendo assim o número total dos mesmos. Estes sinais são depois guardados nas pastas com o desígnio *raw*, significando que estes são os sinais originais. A partir daqui são feitas as operações de transformação necessárias para obter os sinais com ruído, reverberação ou eco para discriminar, sendo que os sinais com eco também serão usados na atenuação do mesmo.

Para criar os sinais com ruído, é gerado um sinal de ruído gaussiano de média 0 e desvio-padrão 0,1. Este sinal de ruído será depois somado aos sinais de fala *Raw* e guardado na pasta correspondente.

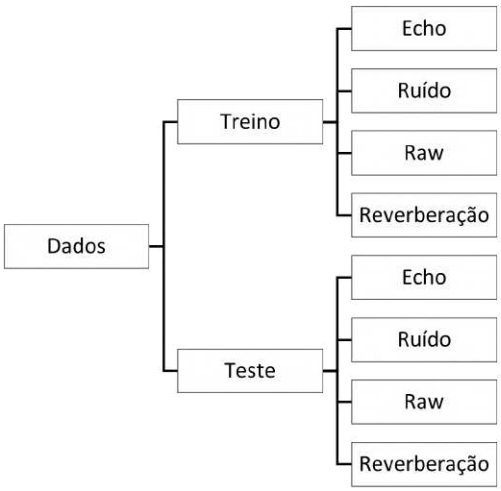


Figura 2.1: Esquema da organização dos dados em pastas.

Os sinais com reverberação são gerados usando funções especializadas da biblioteca *Pedalboard*, criada pela empresa *Spotify* para criar efeitos de áudio. Depois de gerado a reverberação, esta é adicionada ao sinal de fala original e guardada na respetiva pasta.

Por último, os sinais com eco são criados pela sobreposição dos sinais de fala *Raw* com os ecos gerados pela convolução dos sinais originais com os sinais RIR fornecidos. Destes sinais RIR, 4 são usados para criar os sinais de treino e o último para os sinais de teste. Aquando da sobreposição dos sinais, os secos são sobrepostos aos sinais originais com um atraso de 300ms para garantir uma boa perceção dos mesmos, tanto pelo ouvinte como pela NN. No final, os sinais de fala com eco são guardados nas pastas designadas de *Echo*.

A Figura 2.2 mostra um exemplo de sinal para cada uma das categorias explicitadas.

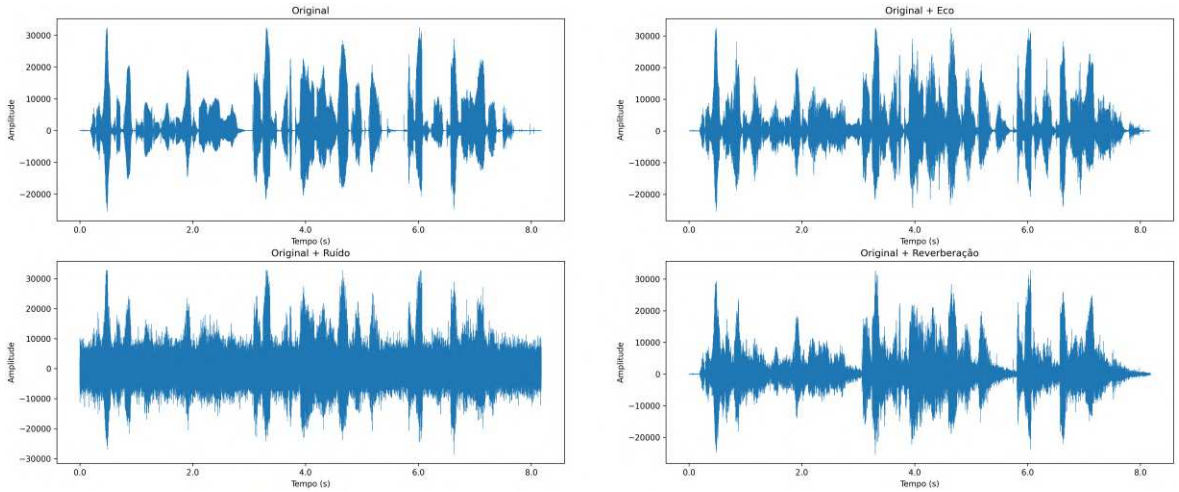


Figura 2.2: Representação gráfica das 4 classes de sinal de fala gerada em *Python*.

2.4 Processamento

Após o pré-processamento, é criado um *dataset* de tensores para treino e outro para teste. Os *datasets* para discriminação de sinais de fala, contêm 10320 e 3800 tensores, respetivamente. Já os *datasets* para atenuação de eco contêm 5160 e 950 tensores, respetivamente. Tensores são uma estrutura de dados usada para guardar os sinais aqui usados num formato fácil de usar como entrada nos modelos. Aqui há uma divergência entre as duas tarefas. Enquanto que na discriminação dos sinais de fala são usadas as 4 categorias de sinais apresentadas, na tarefa de atenuação de eco acústico são usadas apenas as categorias *Raw* e *Echo*. Como a função usada faz uso do nome das pastas como etiquetas e não permite selecionar quais as subpastas a usar, os sinais de fala *Raw* e *Echo* são copiados para outra pasta de dados para permitir o uso da função criadora de *datasets*. Para além de agrupar os sinais e etiquetá-los, a função permite organizá-los em *batches*, isto é, conjuntos de dados que são analisados ao mesmo tempo, e permite ainda determinar quantas amostras dos sinais são usadas. Dado que as NN requerem entradas com tamanhos consistentes, esta opção é usada para garantir que os sinais no *dataset* têm exatamente 2^{17} amostras. Esta escolha garante que todos os sinais tenham, aproximadamente, 8 segundos de duração. Neste trabalho recorreu-se muito a potências de 2, uma escolha justificada pela sua standardização em programação. Inicialmente, as potências de 2 eram usadas devido ao facto dos computadores trabalharem em potências de 2, desde a representação binária que é em base 2, a memória dos computadores também é medida em potências de 2, até a arquitetura dos processadores está desenhada para funcionar em potências de 2 e, como tal, os algoritmos tinham de ser adaptados a esta realidade. Atualmente, os algoritmos tornaram-se tão eficientes que já não é necessário recorrer a potências de 2 para garantir o bom funcionamento dos mesmos. Neste passo também é criado um *dataset* de validação a partir do *dataset* de treino, sendo que 20% dos dados são usados para o efeito. A validação permite avaliar o desempenho do modelo para garantir a sua capacidade para generalizar para novos dados, impedindo o modelo de se sobreajustar aos dados de treino e as consequentes perdas de desempenho e de capacidade de generalização, aumento de complexidade desnecessária do modelo e incapacidade de realizar previsões precisas.

Depois de criados os *datasets*, são calculados os espetrogramas de cada sinal com janelas de 64ms com um deslizamento de 32ms, dando origem a espetrogramas de tamanho 128x1024 em que a primeira dimensão representa o eixo do tempo do espetrograma e a segunda o número de intervalos de frequência. É adicionada uma terceira dimensão representativa dos canais cujo valor é 1, para permitir a ação das camadas de Convolução. Com o tensor a apresentar três elementos nas dimensões, as camadas de Convolução consideram o tensor de entrada como uma imagem bidimensional, exatamente o que se pretende. A intensidade das cores do espetrograma, representam a potência de cada intervalo de frequência no tempo. A camada de entrada das NN requer o preenchimento do hiperparâmetro *Input shape*, por isso usará as dimensões dos espetrogramas para o efeito. Este hiperparâmetro não é único, cada camada tem os seus hiperparâmetros e a alteração desta permite controlar o

comportamento da rede, sendo que estes são definidos durante a construção da mesma.

2.5 Modelo

Por fim, são criados os modelos de NN para as tarefas em causa. Este é um processo iterativo no qual diversos modelos foram testados e a sua parametrização ajustada. Os modelos finais são de aprendizagem supervisionada, recebendo duas entradas, sendo a primeira os dados a processar e a segunda o alvo da rede. Em ambas as tarefas, recorreu-se a modelos sequenciais, isto é, a sua arquitetura é linear e a sua parametrização foi otimizada com recurso a um algoritmo que itera várias combinações de parâmetros para obter o melhor resultado possível. Os resultados desta otimização são enviados para a ferramenta *Tensor Board*, da biblioteca *Tensor Flow*, que permite visualizar um sumário do processo.

2.5.1 Discriminação de perturbações sonoras em sinais de fala

O modelo inicial começa com uma camada de *Resizing* cuja saída tem tamanho (32, 32) para fazer um *downsampling* de modo a reduzir o tempo que demora a treinar o modelo, seguido de uma camada de Normalização em *Batch* para normalizar todos os pixéis da imagem resultante de acordo com a média e desvio-padrão de cada um. De seguida, seguem-se 2 camadas de Convolução 2D, com 64 e 128 filtros respetivamente, ambas com kernel de tamanho (3, 3), deslocamento (1, 1) e função de ativação *ReLU* para extrair caraterísticas de cada imagem que serão usadas pela rede para identificar cada uma. Segue-se uma camada *Max Pooling* de dimensões 2 por 2 juntamente com uma camada *Dropout* com uma taxa de 0,25 para realizar outro *downsmapling* e reduzir a possibilidade de sobreajuste. A camada de *Flatten* que se segue reduz a sua entrada a um vetor unidimensional para facilitar o processamento dos dados na camada que se segue, uma camada de Densificação com 128 neurónios e ativação ReLU que irá aprender as diferentes caraterísticas extraídas anteriormente. Para terminar, há mais uma camada *Dropout* com uma taxa de 0,5 e depois outra Densificação de 4 neurónios, o número de classes de sinais em estudo, que usa os dados recolhidos para classificar os dados de entrada, dando valores para probabilidade pertencer a cada classe. Este modelo foi retirado de um tutorial no site da biblioteca *Tensor Flow* que pretende demonstrar como pré-processar ficheiros *WAV* e treinar uma rede para reconhecimento de palavras [51]. Os modelos subsequentes, apenas diferem do modelo original no número de camadas de convolução e no número de filtros que cada uma contem, somando 64 filtros em cada camada sucessiva, demonstrado pelo esquema seguinte (Figura 2.3):

Todos os modelos usaram o algoritmo de otimização *Adam* com uma taxa de aprendizagem de 1^{-5}, o cálculo das perdas foi feito através da função de *Sparse Categorical Crossentropy* e a métrica usada para avaliar os modelos foi a Precisão com 50 épocas de treino. A tabela 2.1 resume o treino das 4 redes.

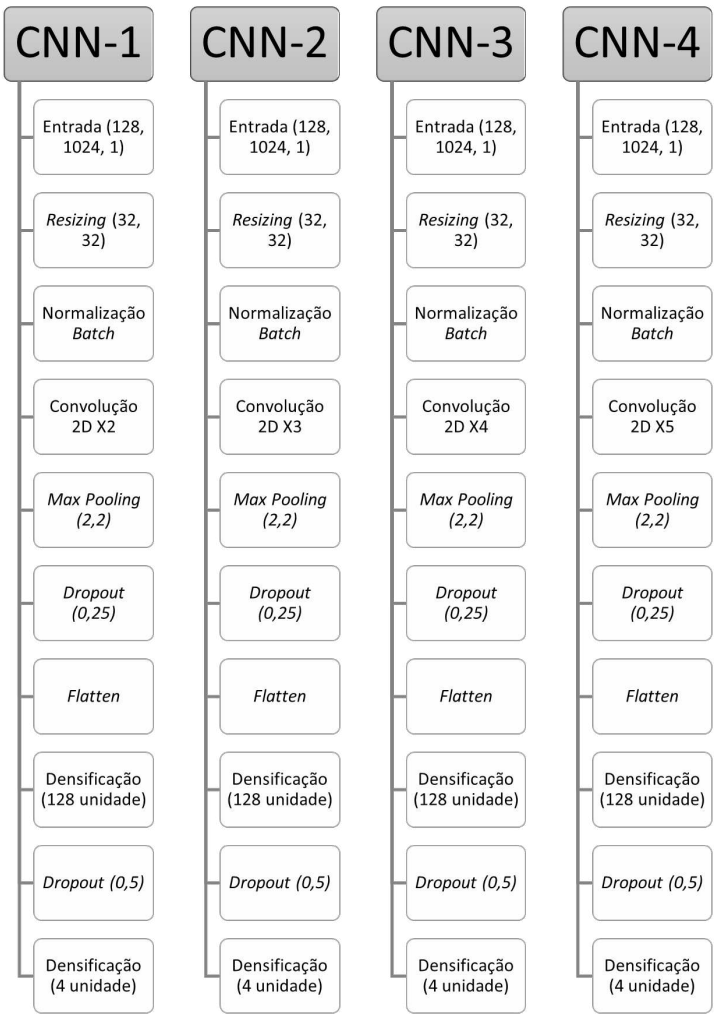


Figura 2.3: Esquematização das 4 CNNs testadas para a Classificação de sinais.

Tabela 2.1: Resumo do treino das redes de classificação.

	CNN-1	**CNN-2**	**CNN-3**	**CNN-4**
Perdas	0.0376	0.0204	0.0145	0.0250
Precisão	0.9898	0.9949	0.9964	0.9924
Tempo Médio de treino por época (s)	17,60	18,08	19,74	20,14

Para a fase de teste, o modelo escolhido foi o CNN-3 e este irá receber como entrada o *dataset* de teste previamente preparado e devolver uma lista com as classes com as quais ele classificou os sinais de teste. No final, criou-se uma matriz de confusão, que mostra a percentagem de sinais de fala corretamente discriminados e a percentagem de dados incorretamente identificados como outra classe, e um relatório de classificação, composto pelas 4 métricas já apresentadas.

Para além de usar os *datasets* com as 4 classes de sinais, também foram criados *datasets* binários

para avaliar a capacidade da rede de classificar com menos classes e confrontar com os resultados obtidos da classificação com todas as classes. Tal como para o *dataset* original, as métricas calculadas são organizadas num relatório de classificação. A composição dos grupos é a seguinte:

1. Eco vs. Ruído
2. Eco vs. Original
3. Eco vs. Reverberação
4. Ruído vs. Original
5. Ruído vs. Reverberação
6. Original vs. Reverberação

Todo o processo de treino e teste é realizado para cada um destes grupos, usando o mesmo modelo, mas com uma modificação na camada de Densificação de saída passando de 4 neurónios para 2 neurónios, a mesma redução no número de classes.

2.5.2 Atenuação

Nesta tarefa foram testados 4 modelos de um tipo diferente dos modelos usados na discriminação de distúrbios sonoros em sinais de fala. Enquanto que estes eram CNNs puras, os modelos de atenuação são CNNs e *Autoencoders*, ou seja, são *Autoencoders* Convolucionais. Os quatro apresentam uma arquitetura semelhante de crescente complexidade do primeiro modelo ao último. O primeiro modelo de atenuação, denominado por CNN-AC1, tem apenas 5 camadas. Após a camada de entrada, vem uma camada de Normalização com média 0 e desvio-padrão 1 para normalizar os pixéis da imagem que entra nesta camada. Segue-se uma camada de Convolução 2D (codificador) e outra de Desconvolução 2D (descodificador) com parâmetros semelhantes, kernel de tamanho (6, 6), deslocamento (1, 1) e função de ativação *ReLU*. A camada de saída da rede é uma Densificação de 1 neurónio, para garantir que a terceira dimensão do tensor de saída, os canais, tem valor 1. O segundo modelo, CNN-AC2, tem 6 camadas e em tudo idêntico ao modelo anterior, apresentando uma camada de Densificação de 64 neurónios entre as camadas de Convolução 2D (codificador) e Desconvolução 2D (descodificador) como a única diferença. O terceiro modelo, CNN-AC3, tem 7 camadas e é baseado no modelo CNN-AC1, acrescentando uma camada de Convolução 2D ao codificador e uma Desconvolução 2D ao descodificador, em tudo semelhantes às camadas já presente, mas com o dobro dos filtros. O quarto e último modelo, CNN-AC4, tem 8 camadas e, tal como na CNN-AC2, apresenta uma camada de Densificação de 64 neurónios entre as camadas de Convolução 2D e Desconvolução 2D. A esquematização das redes está feita na Figura 2.4.

Nestas redes, as camadas de Normalização são usadas para melhorar a convergência da rede, as camadas de Convolução para extrair caraterísticas a partir dos dados de entrada, reduzir as

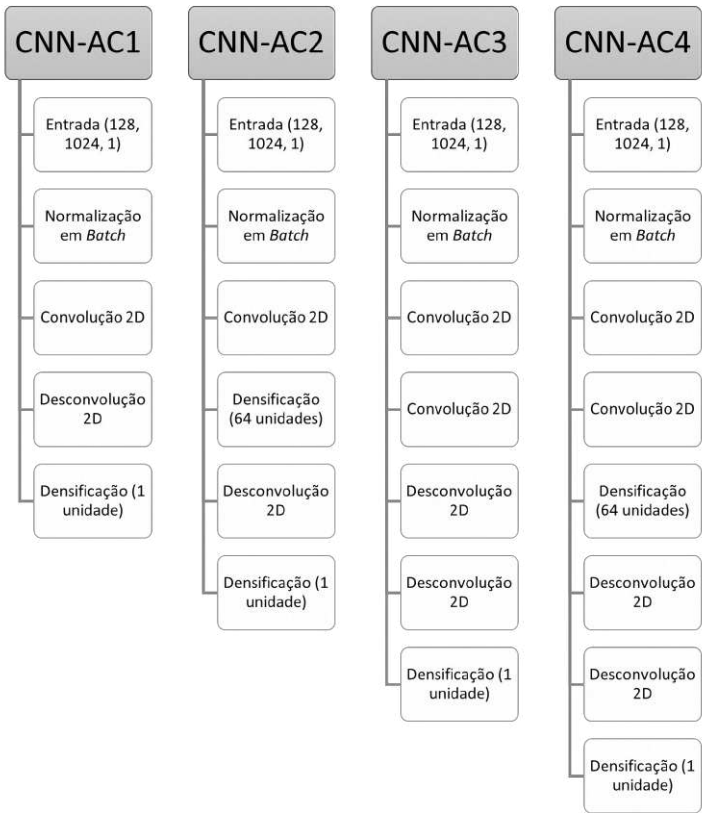


Figura 2.4: Esquematização dos modelos finais usados na tarefa de atenuação de eco.

dimensões dos mesmos (retendo as caraterísticas importante e descartando o resto, idealmente o eco acústico), as camadas de Desconvolução reconstroem os dados a partir da informação extraída anteriormente, as camadas de Densificação intermédias facilitam a transição entre as camadas de codificação (Convolução) e de descodificação (Desconvolução) e as camadas de Densificação finais servem para igualar a dimensionalidade dos dados de saída com a dos dados iniciais.

Todos os modelos recorrem ao algoritmo otimizador *Adam* e a função de cálculo das perdas dos modelos foi a EQM. A Precisão não foi calculada aqui, estando o interesse mais na minimização das perdas da rede. Todas as redes foram treinadas com uma taxa de aprendizagem de 1-5, com validação, e durante 50 épocas com uma função de paragem. A tabela 2.2 resume o treino das redes em questão:

As imagens seguintes (Figuras 2.5 (`https://drive.google.com/file/d/1tqIeUYK56gsrwpd4_C4RjMRx4CJY5dfV/view?usp=drive_link`) e 2.6 (`https://drive.google.com/file/d/1B_imhhjDYmOhKLkJcQooEFZlbvTjZpRn/view?usp=drive_link`)), representam um sinal escolhido como exemplo, e usado por estas redes.

Na fase de teste e tal como foi feito durante a classificação, os modelos recebem o *dataset* de teste previamente preparado para fazer a atenuação de eco. Aqui os sinais de fala são passados um a um por cada modelo, seguido do cálculo de uma métrica objetiva sobre os sinais resultantes para

Tabela 2.2: Resumo do treino das redes de atenuação.

	CNN-AC1	**CNN-AC2**	**CNN-AC3**	**CNN-AC4**
Perdas	0.2440	0.2432	0.2138	0.2115
Épocas Treinadas	21	24	32	32
Tempo Médio de treino por época (s)	95	111	485	472

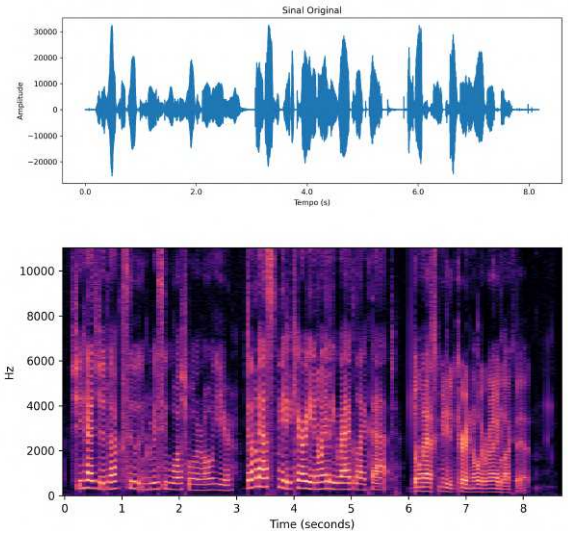


Figura 2.5: Representação gráfica do sinal original e o seu espetrograma que serve como alvo das NN.

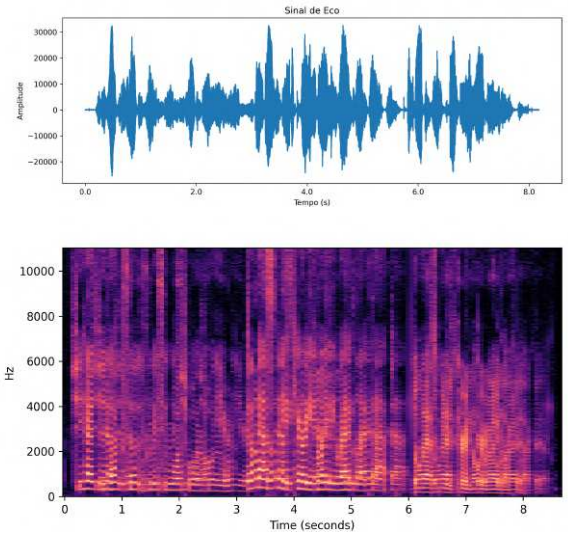


Figura 2.6: Representação gráfica do sinal com eco e o seu espetrograma que serve como entrada das NN.

avaliar o desempenho do sistema.

2.6 Fluxo de dados

Para sumariar o fluxo de dados de cada tarefa, foram construídos os seguintes fluxogramas(Figuras 2.7 e 2.8):

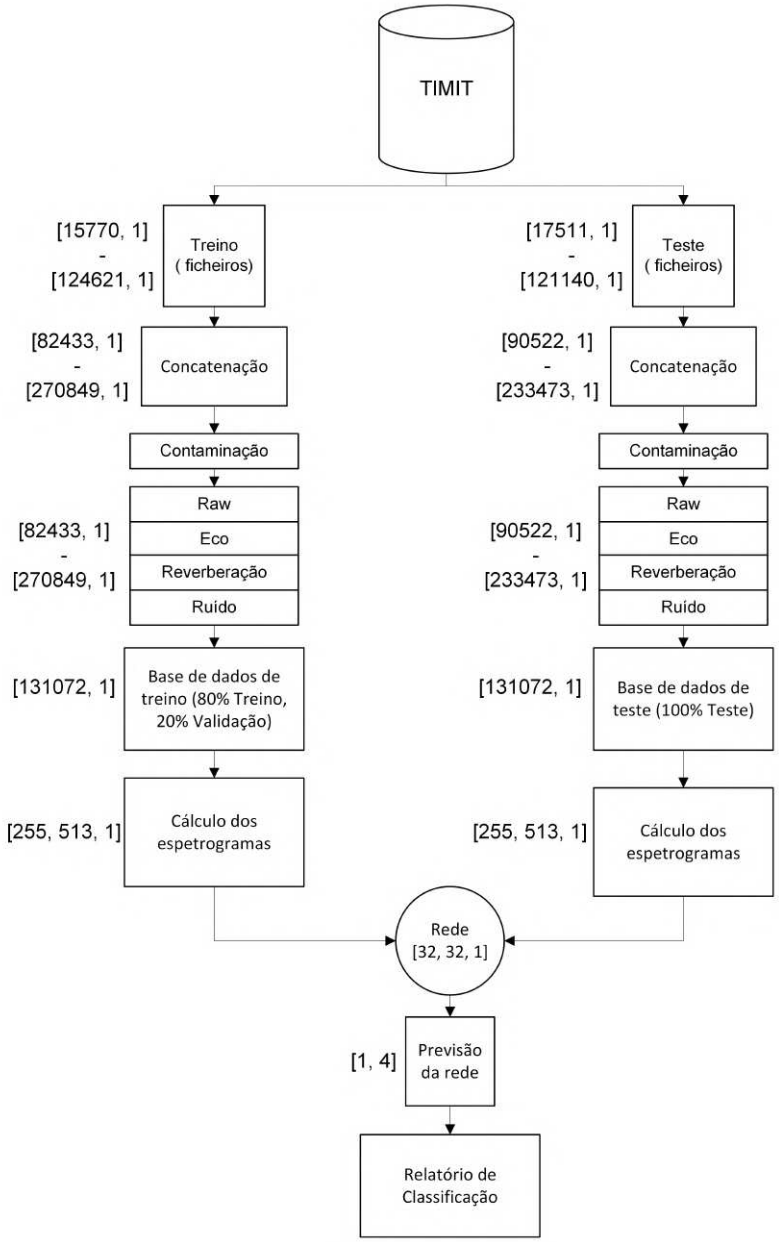


Figura 2.7: Fluxograma de dados para Classificação.

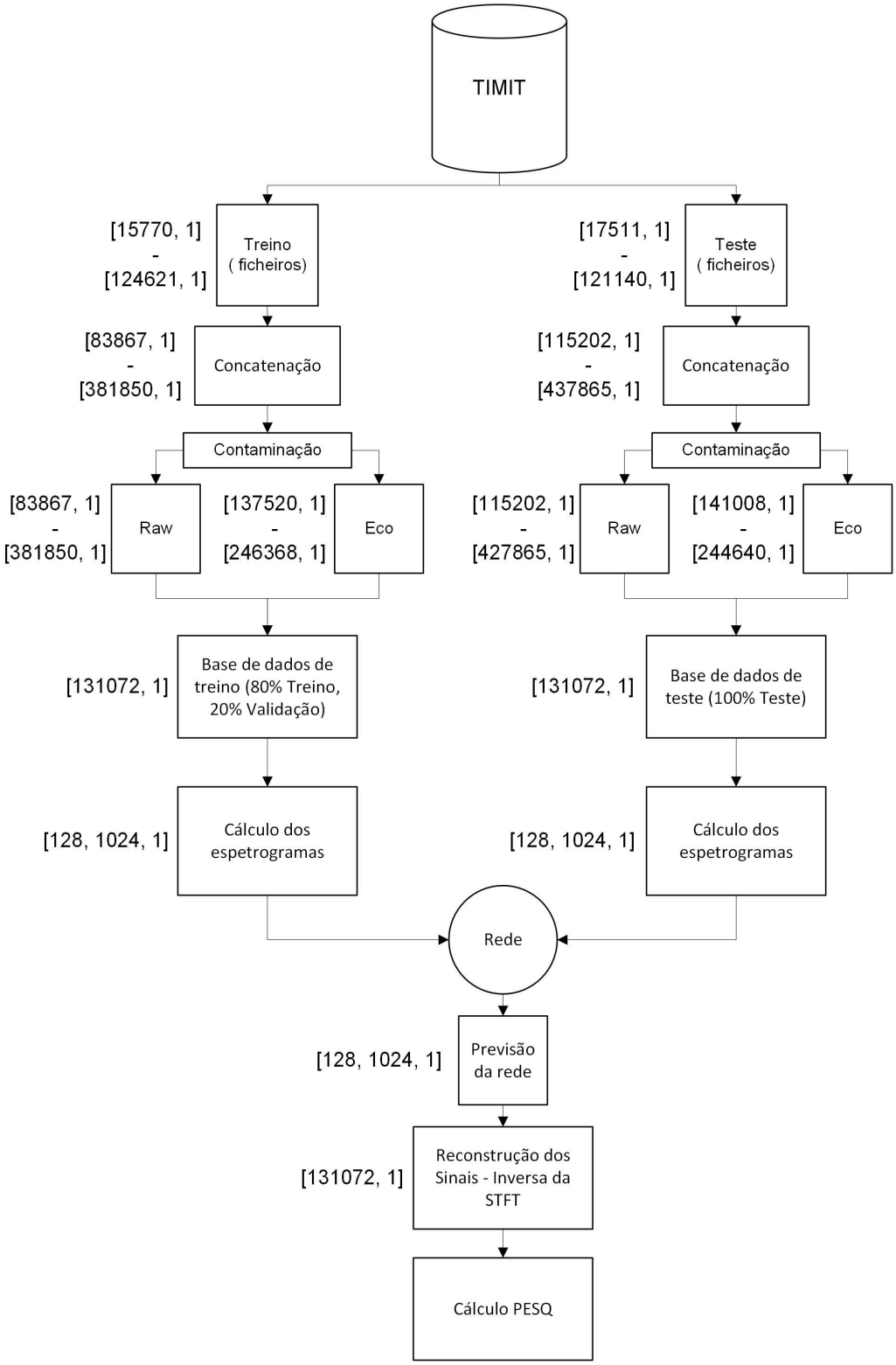


Figura 2.8: Fluxograma de dados para Atenuação.

CAPÍTULO 3

Resultados obtidos e discussão - Discriminação entre diferentes perturbações do sinal de fala

Neste capítulo apresentam-se os resultados obtidos durante a discriminação entre diferentes perturbações do sinal de fala e é feita uma análise crítica dos mesmos.

3.1 Resultados obtidos

O modelo escolhido para avançar para a fase de teste foi o CNN-3, pois apresenta o valor de Exatidão superior. No entanto, qualquer um dos outros modelos seria válido, porque todos apresentam valores de Exatidão muito elevados.

Para melhor compreensão dos dados, estes foram organizados sob a forma de uma tabela com *heatmap* que sumaria os relatórios de classificação obtidos. Todas as métricas calculadas podem ser obtidas a partir da seguinte matriz de confusão (Figura 3.1).

Como podemos ver por esta matriz de confusão, o modelo identificou corretamente 91,75% dos sinais com eco, sendo 7,63% foram erroneamente identificados como sinal original e 0,62% como sinal com reverberação. Já nos sinais com ruído, o modelo acertou na totalidade dos casos, enquanto que durante a discriminação dos sinais de fala originais acertou em 99,36% dos casos, identificando erradamente 0,64% como sinais com eco. Por último, o modelo acertou em 99,17% dos sinais com reverberação e identificou 0,83% como sinais com eco. Os relatórios de classificação, foram sumariados na seguinte tabela com *heatmap* (Figura 3.2), mostrando os valores de cada uma das 4 métricas para cada grupo de sinais de fala.

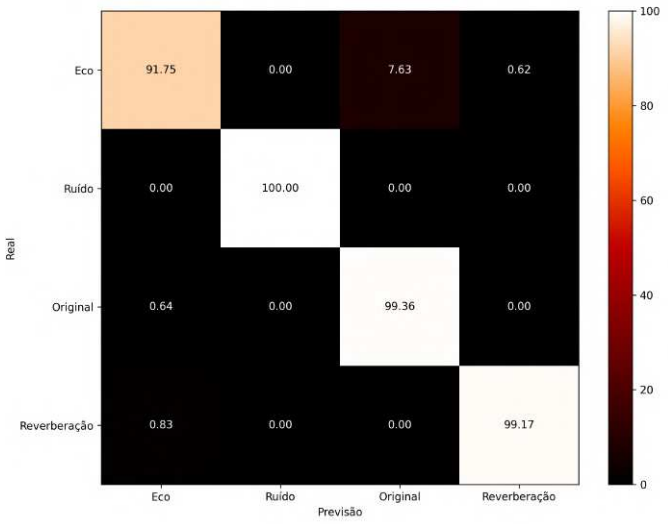


Figura 3.1: Matriz de confusão em valores percentuais.

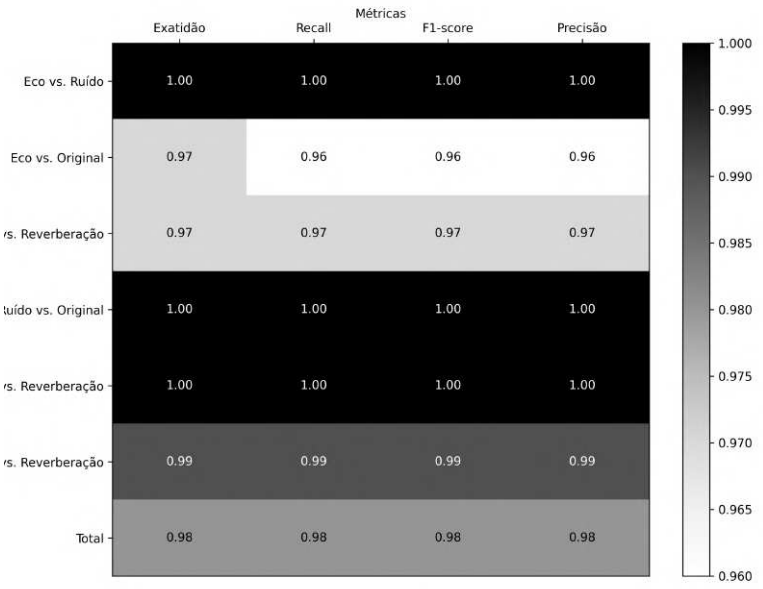


Figura 3.2: Valores obtidos para as métricas para cada grupo binário e para o conjunto completo de classes em formato *heatmap*

.

3.2 Discussão dos resultados

Com os resultados apresentados no *heatmap*, podemos verificar que, o modelo CNN-3 é apresenta elevada capacidade em distinguir entre as 4 classes de sinal de fala. Consegue classificar sem erros, os grupos 1, 4 e 5, isto é, distingue bem eco de ruído, ruído de sinal original ou de reverberação e original de reverberação. Interpretando as métricas, pode-se dizer que, nestes 4 grupos, o modelo não apresentou nem Falsos Positivos, (de acordo com a Exatidão) nem Falsos Negativos (pelo valor do *Recall*) e, consequentemente, previu corretamente todos os Positivos Reais (*F1-score* e Exatidão) e também os Negativos Reais (apenas Precisão).

A maior dificuldade para o modelo está em distinguir um sinal com eco de um sinal original, fácil de perceber olhando para a Figura 3.1, onde se verifica que, de todos os sinais, os mais semelhantes são os sinais original e com eco. Apesar disto, o modelo teve uma pontuação de 96% em todas as métricas exceto na Exatidão onde teve 97%, revelando um desempenho forte mesmo no seu ponto mais fraco. Isto quer dizer que teve 4% de Falsos Positivos, 4% de Falsos Negativos, 96% de Positivos Reais corretos e 96% de Negativos Reais certos neste grupo.

O grupo 3 (Eco vs. Reverberação) é o segundo grupo com as pontuações mais baixas (ainda que elevadas) com 97% em todas as suas métricas. Traduzindo valores por resultados concretos, neste grupo, o modelo teve 3% de Falsos Positivos, 3% de Falsos Negativos e 97% de Positivos e Negativos Reais corretos.

Por último, no grupo que engloba as 4 classes, o modelo teve uma pontuação de 98%. Com 2% de Falsos Positivos e Negativos e 98% de Positivos e Negativos Reais, pode-se concluir que o modelo é capaz de distinguir muito bem os 4 tipos de sinal quando estão misturados uns com os outros.

No global, o modelo usado teve um ótimo desempenho, mas apenas como prova de conceito. Para se puder aplicar este modelo em situações reais, é necessário diversificar o tipo de distúrbios usados no treino. Na tratamento dos sinais de fala com eco, diferentes atrasos produziriam áudios diferentes e em situação real os ecos terão atrasos variados e os RIR também serão muito mais variados. Já no caso de ruído, há muitos mais tipos de ruído do que aquele aqui usado que podem ser adquiridos por microfone, como ruído branco, ruído castanho, ruído ambiental, isto é, sons de fundo do dia-a-dia como cães a ladrar ou carros a passar, ou simplesmente conversas cruzadas, nas quais se ouve mais do que uma pessoa a falar ao mesmo tempo. Por último, a reverberação não é sempre igual como nos sinais de fala com reverberação gerados para este trabalho. Ao variar os parâmetros de geração, é possível obter reverberações diferentes. Em suma, há muito que se pode fazer para robustecer os modelos usados, podendo mesmo iterar sobre eles e fazer alterações aos mesmos para lidar com mais situações descritas.

CAPÍTULO 4

Resultados e discussão - Atenuação de Eco Acústico

Neste capítulo apresentam-se os resultados obtidos durante a Atenuação de eco acústico e é feita uma análise crítica dos mesmos.

4.1 Resultados obtidos

Para efeitos de comparação visual, as imagens 4.1, 4.2, 4.3 e 4.4 representam o mesmo sinal das imagens 2.5 e 2.6.

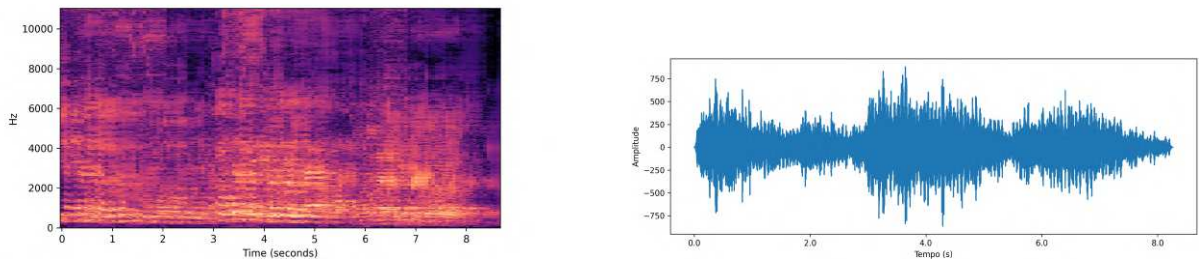


Figura 4.1: Resultados obtidos pela rede CNN-AC1.

A seguinte lista apresenta os links de *Google Drive* que contêm os ficheiros de áudio resultantes para consulta e comparação auditiva:

- Figura 4.1: `https://drive.google.com/file/d/19aMksUb_EZZ_dkh4YCNaRWyEaIstADQp/view?usp=drive_link`
- Figura 4.2: `https://drive.google.com/file/d/1a-W5bOfJd8rPGiW705SH-kkqUqeydIwa/view?usp=drive_link`

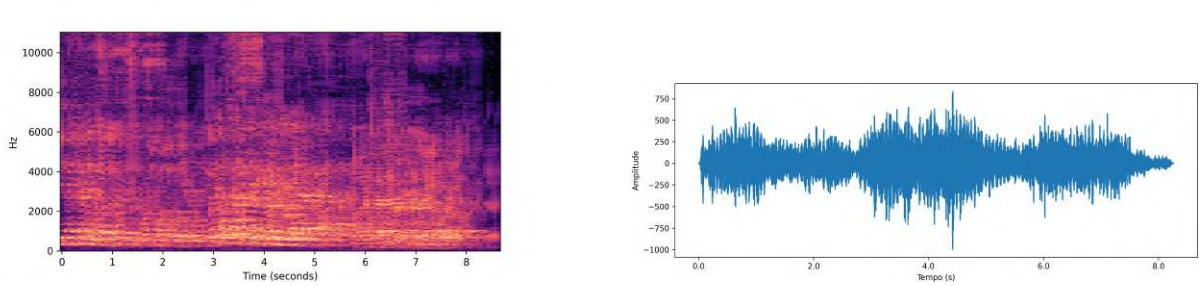


Figura 4.2: Resultados obtidos pela rede CNN-AC2.

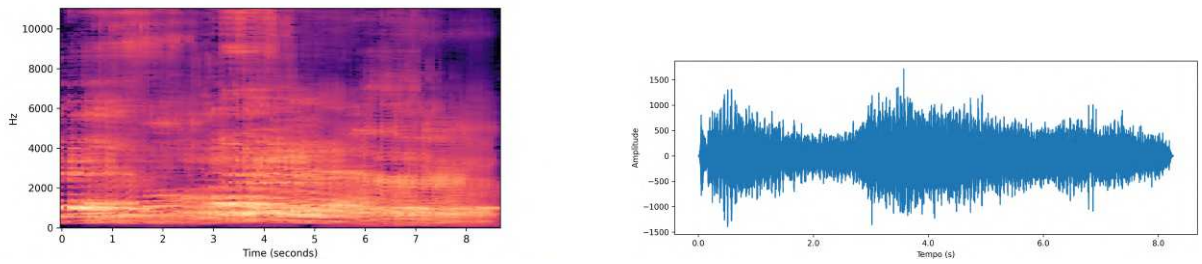


Figura 4.3: Resultados obtidos pela rede CNN-AC3.

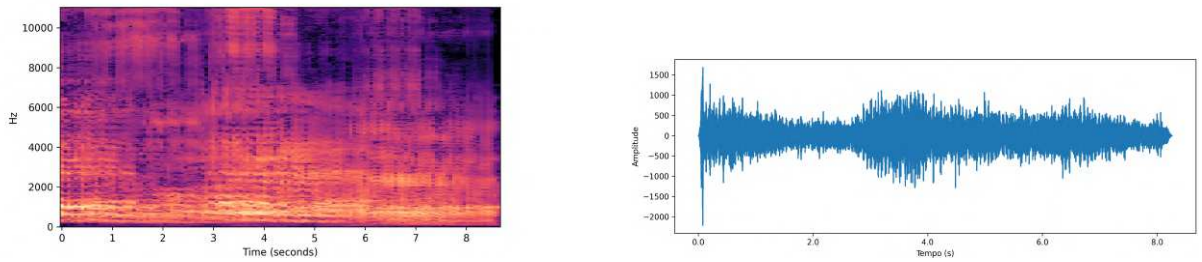


Figura 4.4: Resultados obtidos pela rede CNN-AC4.

- Figura 4.3: `https://drive.google.com/file/d/1GMFIavgPxmt5GSrtpd0W3qhVZj7G-kGJ/view?usp=drive_link`
- Figura 4.4: `https://drive.google.com/file/d/1_F_2BwBN8dF2NXcIX8OupygmH-ih_fxi/view?usp=drive_link`

Como se pode visualizar, os resultados obtidos diferem bastante do pretendido. Das 4 redes, a CNN-AC1 foi a que obteve melhores resultados, embora seja difícil de o visualizar. Os ficheiros de áudio permitem uma melhor análise dos resultados em termos da sua proximidade com o sinal de fala alvo.

Antes de analisar a métrica PESQ, pode-se fazer uma comparação dos resultados de Perdas e do tempo despendido por época de treino entre as redes de Discriminação de distúrbios sonoros e as de

	CNN-AC1	CNN-AC2	CNN-AC3	CNN-AC4
Mínimo	1,02	1,02	1,02	1,02
Máximo	4,31	4,29	3,80	4,48
Média	1,09	1,12	1,12	1,11
Mediana	1,04	1,04	1,05	1,05

Tabela 4.1: Tabela sumário dos valores PESQ obtidos para os 4 modelos em estudo.

Atenuação. Pelas tabelas 2.1 e 2.2, as redes de Atenuação tiveram perdas de maior magnitude que as redes de Classificação e o tempo médio que cada época demorou a treinar as redes de Atenuação também foi bastante superior.

A tabela 4.1 mostra várias estatísticas calculadas para os valores de PESQ obtidos para cada uma das NN explicitadas. Cada uma destas métricas servem para melhor analisar os dados, sendo que os mínimos e máximos dão os limites dos valores de PESQ obtidos e a média e a mediana indicam a tendência dos dados, isto é, para que lado dos limites estes estão mais inclinados. A mediana é mais relevante que a média aqui pois não é influenciada pelos *outliers* presentes nos dados em análise.

4.2 Discussão dos resultados

Dado que foram usados 950 sinais no teste de cada um dos modelos, obteve-se no final 950 valores de PESQ, um para cada instância. Como tal, foram calculadas algumas estatísticas para sumariar os resultados obtidos, como se pode ver pela tabela 4.1.

Todos os modelos apresentam o mesmo valor de PESQ mínimo, variando em tudo o resto com semelhanças em alguns casos.

Apesar de os valores máximos de PESQ serem bastante altos (o limite superior da escala é 4,5), estes são *outliers* e possivelmente incorretos, pois se se ouvir os ficheiros de áudio respetivos, percebe-se que são tão impercétiveis como os que apresentam PESQ mais baixo, o que justifica a importância de confrontar as pontuações de PESQ com a perceção humana dos sinais de fala.

Analisando os dados como um todo, verifica-se que a pontuação PESQ é bastante baixa, apoiado pela audição dos sinais em causa. Apesar de se perceber que há um interlocutor que está de a dizer alguma coisa, os sinais de fala apresentam-se demasiado degradados para afirmar que os resultados correspondem às expectativas.

É grande a probabilidade de o problema estar na estrutura dos modelos usados como se pode verificar, não só pelas pontuações PESQ, mas também pelas estatísticas de treino das redes. Estas mostraram-se lentas em comparação com as redes de discriminação e com uma capacidade preditiva menor e maiores perdas.

Esta tentativa de usar *Autoencoders* Convolucionais para tentar fazer a atenuação do eco como se este fosse um ruído para reduzir demonstrou que é necessário desenvolver mais a arquitetura das NN aqui usadas ou mesmo substituir por outro tipo de rede.

CAPÍTULO 5

Conclusões e Trabalho Futuro

Neste capítulo pretende-se expor as conclusões a que se chegou neste trabalho. Essas conclusões têm por base os resultados obtidos. São ainda abordadas futuras linhas de investigação.

5.1 Conclusões

É inegável que a Inteligência Artificial é amplamente utilizada na atualidade nas mais diversas áreas, tornando-se por isso cada vez mais importante compreender o que ela é e como fazer uso das diversas técnicas que ela engloba. Nesta Tese, o trabalho desenvolveu-se usando a IA, focando-se em *Deep Learning* e aplicando técnicas de Redes Neuronais no estudo da fala.

A consecução deste trabalho implicou todo um processo evolutivo, feito por tentativas, para criar um sistema capaz de atenuar o eco acústico, que exigiu mudanças desde tipologia de rede e do processamento dos sinais de fala até uma mudança de linguagem de programação. Durante a pesquisa por um método que se pudesse adaptar para atenuar eco, encontrou-se uma rede de classificação de ficheiros de áudio. Após algum desenvolvimento desta rede, ficou-se com um método de discriminação de sinais de fala completo que se explicitou nesta dissertação.

As redes de Classificação criadas pretendem diferenciar sinais de fala de 4 tipos, sendo eles os sinais originais, sinais com eco, sinais com ruído e sinais com reverberação. Para facilitar a sua análise recorreu-se à STFT e aos espetrogramas calculados a partir desta, permitindo adaptar técnicas de classificação de imagem, como a CNN, um assunto muito estudado e, por consequência, bem conhecido em termos de estratégia e resultados. A adaptação destes métodos permitiu criar um modelo de CNN muito capaz nas circunstâncias para as quais foi desenhado. Claro está, que este é uma prova de conceito, bastante boa, mas com capacidade para crescer, tanto em robustez como em desempenho.

A Atenuação de Eco Acústico não correspondeu às expectativas, não tendo atingido os resultados esperados. O objetivo seria a criação de um ou mais modelos capazes de atenuar eco acústico. Houve um grande foco em modelos convolucionais que se mostraram todos eles incapazes de devolver um sinal bem audível, distorcendo-o para além da presença do eco acústico. Os modelos finais eram do tipo *Autoencoder* Convolucional. A ideia era tratar este distúrbio como ruído, e como uma das aplicações de Autoencoders é a redução de ruído e como os espetrogramas são, essencialmente, uma imagem de som, escolheu-se esta arquitetura de NN. Como se pode ver pelos resultados, não é possível afirmar que este é o caminho para realizar esta tarefa. É possível também que, com um trabalho de desenvolvimento maior dos modelos já criados, haja possibilidade de atingir o objetivo pretendido.

Em suma, esta dissertação permitiu expandir o conhecimento sobre inteligência artificial, desde a sua definição e composição até às suas aplicações, criando oportunidades para trabalhar mais na área.

5.2 Trabalho Futuro

De futuro, pretende-se elevar de patamar o sistema de discriminação de sinais de fala, introduzindo mais variáveis como já foi proposto. O desempenho deste sistema também pode ser melhorado para fornecer uma resposta mais instantânea. Há também a possibilidade de encadear este sistema de discriminação com sistemas de atenuação dos distúrbios sonoros identificados. Por exemplo, o sistema identifica um dado sinal de fala como tendo eco e envia este para um sistema de atenuação de eco. O mesmo pode-se fazer para sinais de fala com ruído ou reverberação.

Já o sistema de atenuação, o que foi criado durante o decorrer deste trabalho tem de ser desenvolvido muito mais, para que seja capaz de atenuar eco sem distorcer irrevogavelmente o sinal de entrada. Também é preciso ter em atenção, que os sinais de fala adquiridos não terão apenas eco, havendo a possibilidade de conterem ruído ou reverberação também. Por isso, o sistema terá de aprender, ou a ignorar o ruído e a reverberação e focar-se no eco, deixando as outras duas perturbações para outros sistemas desenhados para lidar com elas, ou terá de ser desenvolvido de tal forma que seja capaz de atenuar as três perturbações ao mesmo tempo sem comprometer a inteligibilidade do sinal de entrada. No caso da criação de sistemas individuais para lidar com cada um dos distúrbios sonoros aqui referidos, poder-se-ia encadear três sistemas diferentes, um para cada distúrbio, e essencialmente atenuar cada um em cadeia. Claro está, sempre mantendo a inteligibilidade do som.

Mais do que desenvolver os algoritmos, é preciso aplicá-los, introduzindo-os em dispositivos como um aparelho auditivo para melhorar as suas funções. Para tal é necessário estudar como introduzir estes sistemas nos aparelhos e como os melhorar para que o desempenho conjunto seja suficiente para comercialização.

ANEXO A

Evolução do Método - Representação Gráfica

São aqui apresentadas as representações gráficas referentes às várias tentativas explicitadas no Capítulo 2, Secção 1. Apenas as tentativas mais importantes estarão aqui representadas.

Tentativa 3

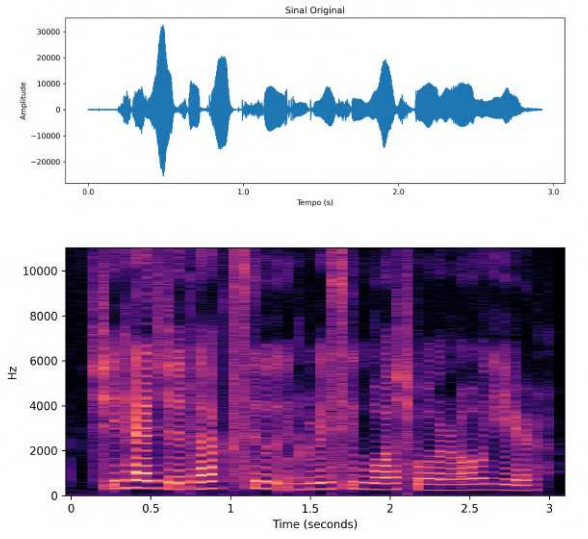


Figura A.1: Representação gráfica do sinal original inalterado (com períodos de silêncios ou concatenado com outros sinais) e o seu espetrograma que serve como alvo das NN.

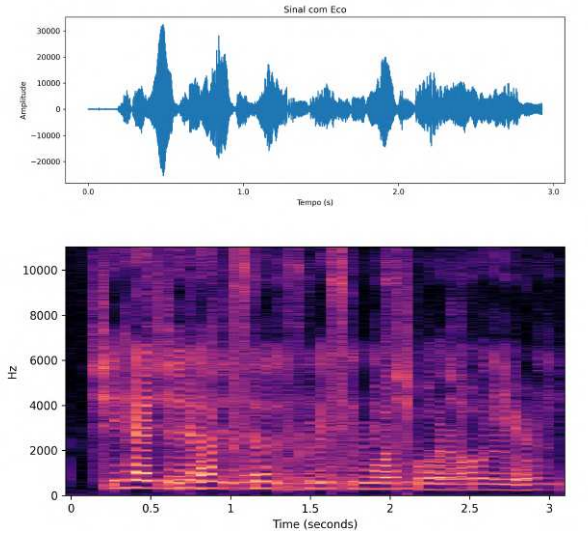


Figura A.2: Representação gráfica do sinal com eco e o seu espetrograma que serve como entrada das NN.

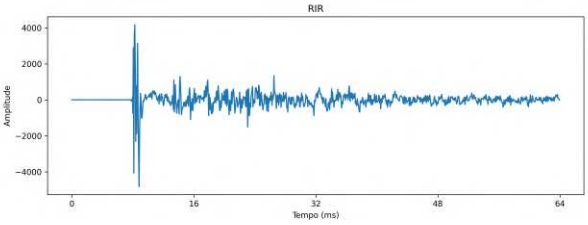


Figura A.3: Representação gráfica de um RIR.

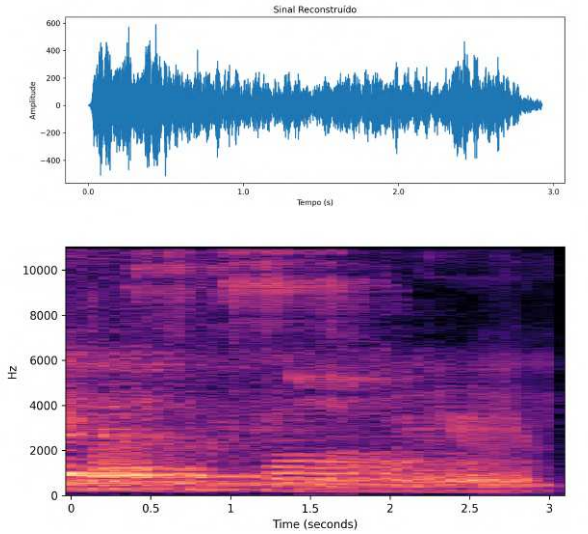


Figura A.4: Representação gráfica do sinal reconstruido e o seu espetrograma.

Tentativa 4

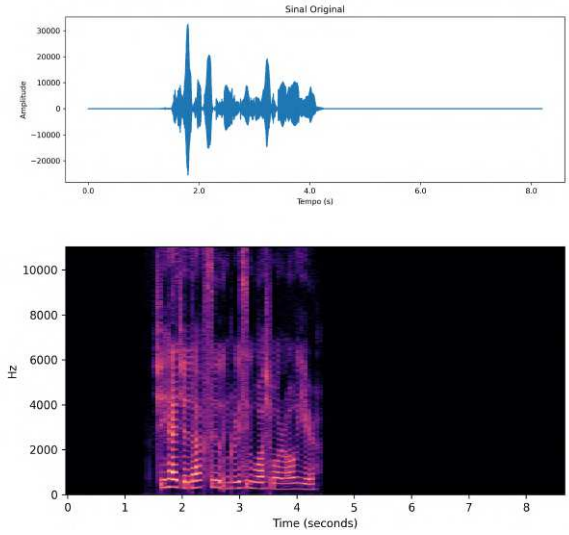


Figura A.5: Sinal original com períodos de silêncios adicionados no início e no fim e o seu espetrograma que serve como alvo das NN.

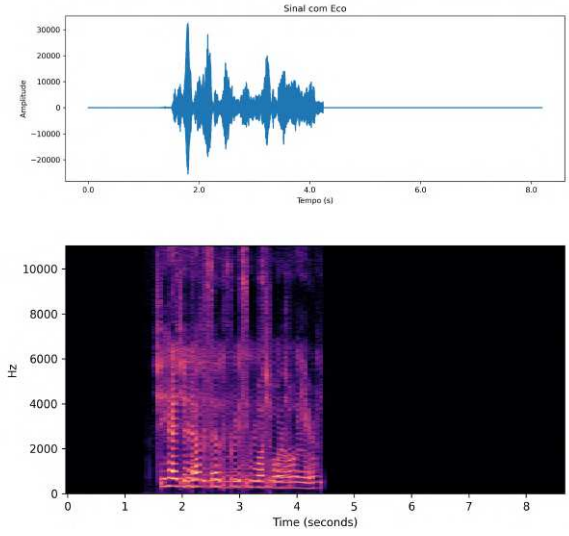


Figura A.6: Sinal com eco e o seu espetrograma que serve como entrada das NN.

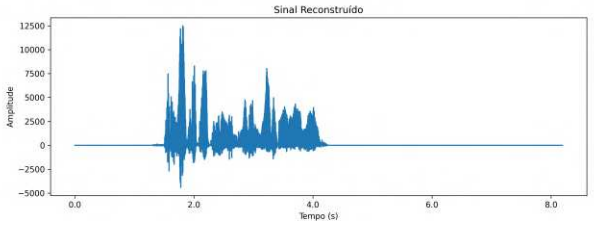


Figura A.7: Representação gráfica do sinal obtido trabalhando com o sinal de entrada diretamente.

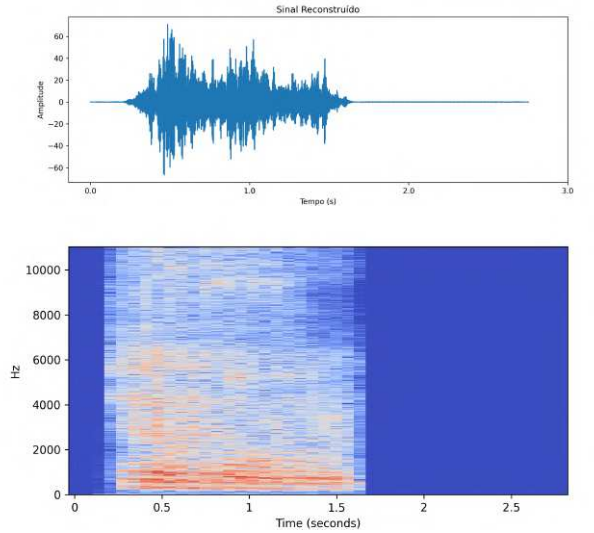


Figura A.8: Representação gráfica do espetrograma obtido e do sinal reconstruido a partir dele. Aqui foram usados espetrogramas nas entradas para obter este espetrograma na saída.

Tentativa 7

O sinais de referência são os mesmos da tentativa 4.

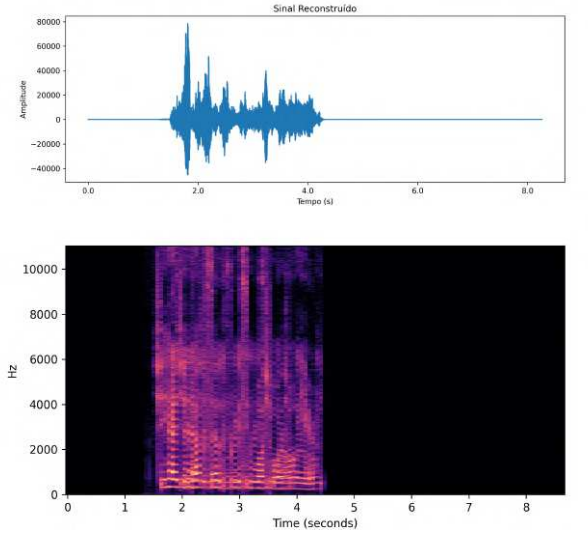


Figura A.9: Resultado da tentativa 7, o sinal reconstruido e o espetrograma obtido na saída da rede.

Tentativa 8

Tal como na tentativa 7, os sinais de referência são os mesmos que foram usados na tentativa 4.

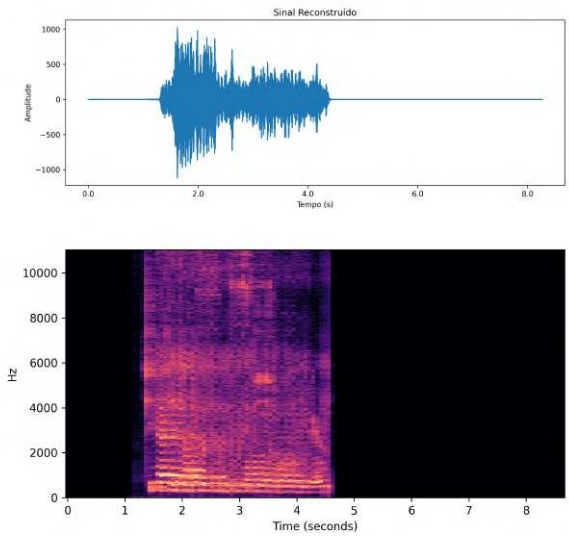


Figura A.10: Resultados do primeiro modelo.

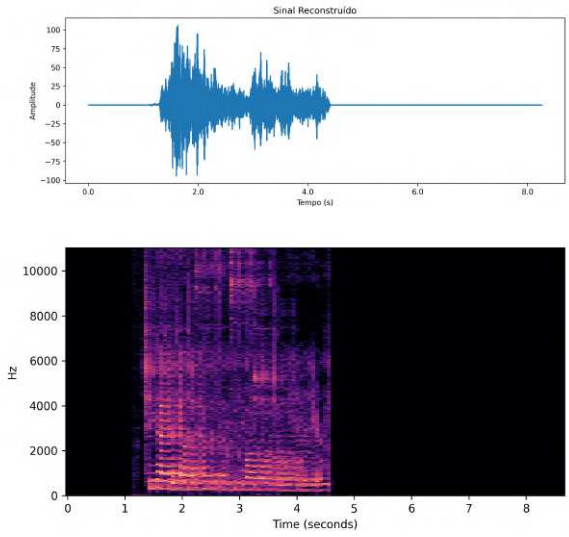


Figura A.11: Resultados do segundo modelo.

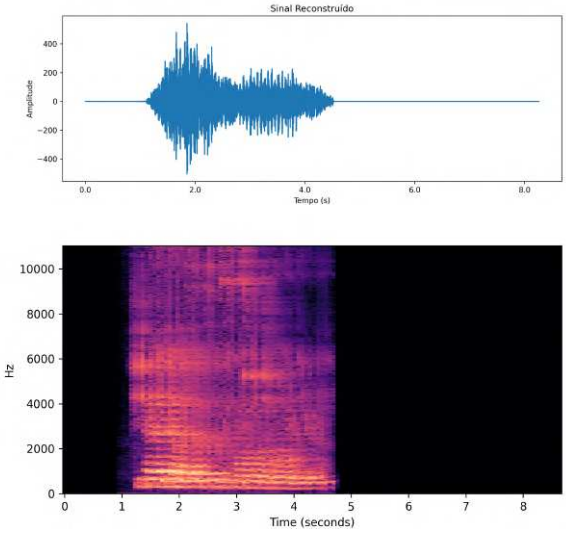


Figura A.12: Resultados do terceiro modelo.

Bibliografia

[1] X. Hu, A.-Q. Hu, Q. Luo e T.-Y. Cai. "A novel adaptive acoustic echo cancellation for teleconferencing systems". Em: *Proceedings. International Conference on Machine Learning and Cybernetics*. Vol. 2. 2002, 1005–1009 vol.2. DOI: 10.1109/ICMLC.2002.1174534.

[2] Y. Feng, G. Tang, R. Liang e Q. Wang. "An improved echo cancellation algorithm for hearing aids". Em: *Electronics, Electrical Engineering and Information Science*. 2016, pp. 362–369. DOI: 10.1142/9789814740135_0038.

[3] Google Cloud. *What is Artificial Intelligence?* https://cloud.google.com/learn/what-is-artificial-intelligence. data de consulta: 30/07/23. 2023.

[4] Google Cloud. *AI vs. Machine Learning: How Do They Differ?* https://cloud.google.com/learn/artificial-intelligence-vs-machine-learning. data de consulta: 30/07/23. 2022.

[5] IBM. *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks*. https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/. data de consulta: 30/07/23. 2017.

[6] E. Gomede. *Understanding Loss Functions in Deep Learning*. https://medium.com/@evertongomede/understanding-loss-functions-in-deep-learning-9f06e5090f20. data de consulta: 01/15/2024. 2023.

[7] IBM. *Neural Networks*. https://www.ibm.com/topics/neural-networks. data de consulta: 30/07/23. 2017.

[8] F. Chollet. *Deep Learning with Python*. 2ª ed. Shelter Island, New York: Manning Publications, 2021, p. 504. ISBN: 978-1617296864.

[9] A. Burkov. *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019, p. 160. ISBN: 978-1777005474.

[10] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza e E. A. Chavez-Urbiola. *Loss Functions and Metrics in Deep Learning*. 2023. arXiv: 2307.02694 [cs.LG].

[11] J. Brownlee. *A Gentle Introduction to the ADAM Optimization Algorithm*. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. 2015.

[12] D. P. Kingma e J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[13] S. Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[14] V. Efimov. *Understanding Deep Learning Optimizers: Momentum, AdaGrad, RMSProp & Adam*. https://towardsdatascience.com/understanding-deep-learning-optimizers-momentum-adagrad-rmsprop-adam-e311e377e9c2. data de consulta: 15/01/2024. 2023.

[15] I. Goodfellow, Y. Bengio e A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016, p. 800. ISBN: 978-0262035613.

[16] D. T. Chang. *Probabilistic Deep Learning with Probabilistic Neural Networks and Deep Probabilistic Models*. 2021. arXiv: 2106.00120 [cs.LG].

[17] M. Nielsen. *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com/. 2019.

[18] O. Theobald. *Machine Learning for Absolute Beginners: A Plain English Introduction*. 2ª ed. Independently Published, 2017, p. 168. ISBN: 978-1520951409.

[19] A. S. Rathore. *Layers in Neural network*. https://medium.com/nerd-for-tech/layers-in-neural-network-90d48a5a42fb. data de consulta: 16/01/2024. 2021.

[20] A. Géron. *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. 2ª ed. Sebastopol: O'Reilly Media, 2019, p. 856. ISBN: 978-1492032649.

[21] Baeldung. *How ReLU and Dropout Layers work in CNNs*. https://www.baeldung.com/cs/ml-ReLU-dropout-layers. data de consulta: 16/01/2024. 2023.

[22] *Flatten Layer*. https://keras.io/api/layers/reshaping_layers/flatten/. data de consulta: 16/01/2024. 2023.

[23] Kaggle. *L24 Convolutional neural networks*. https://www.kaggle.com/code/jhskaggle/l24-convolutional-neural-networks. data de consulta: 16/01/2024. 2022.

[24] Keras. *Reshape Layer*. https://keras.io/api/layers/reshaping_layers/reshape/. data de consulta: 16/01/2024. 2023.

[25] Keras. *Resizing Layer*. https://keras.io/api/layers/preprocessing_layers/image_preprocessing/resizing/. data de consulta: 17/01/204. 2023.

[26] SaffronEdge. *Feedforward vs Backpropagation in ANN*. https://www.linkedin.com/pulse/feedforward-vs-backpropagation-ann-saffronedge1. 2021.

[27] Y. Bengio, P. Simard e P. Frasconi. "Learning long-term dependencies with gradient descent is difficult". Em: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[28] S. Hochreiter e J. Schmidhuber. "Long Short-term Memory". Em: *Neural computation* 9 (1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[29] S. Dobilas. *LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past*. https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e. data de consulta: 15/01/2024. 2022.

[30] J. Chung, C. Gulcehre, K. Cho e Y. Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[31] L. Ke, W. Pei, R. Li, X. Shen e Y.-W. Tai. *Reflective Decoding Network for Image Captioning*. 2019. arXiv: 1908.11824 [cs.CV].

[32] S. Shah. *Convolutional Neural Network: An Overview*. data de consulta: 01/08/23. 2022.

[33] Y. Song, S. Hyun e Y.-G. Cheong. "Analysis of Autoencoders for Network Intrusion Detection". Em: *Sensors* 21.13 (2021). ISSN: 1424-8220. URL: \url{https://www.mdpi.com/1424-8220/21/13/4294}.

[34] D. Snover, C. W. Johnson, M. J. Bianco e P. Gerstoft. "Deep Clustering to Identify Sources of Urban Seismic Noise in Long Beach, California". Em: *Seismological Research Letters* 92.2A (2020), pp. 1011–1022. ISSN: 0895-0695. DOI: 10.1785/0220200164. URL: \url{https://doi.org/10.1785/0220200164}.

[35] T. Wong e Z. Luo. *Recurrent Auto-Encoder Model for Multidimensional Time Series Representation*. https://openreview.net/forum?id=r1cLblgCZ. 2018.

[36] D. M. W. Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2020. arXiv: 2010.16061 [cs.LG].

[37] ITU-T. *Perceptual Evaluation of Speech Quality (PESQ): An Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs*. Rel. téc. Geneva: International Telecommunications Union (ITU), 2001.

[38] C. Zhang e X. Zhang. "A Robust and Cascaded Acoustic Echo Cancellation Based on Deep Learning". Em: *Proc. Interspeech 2020*. 2020, pp. 3940–3944. DOI: 10.21437/Interspeech.2020-1260.

[39] J. Field. *Psycholinguistics: The Key Concepts*. Routledge, 2004, p. 386. ISBN: 978-0415258906.

[40] J. V. Vyas e A. C. Suthar. "Study of Speech Recognition Technology and its Significance in Human-Machine Interface". Em: *International Journal of Science Technology & Engineering* 3.10 (2017), pp. 416–422. DOI: 10.13140/RF.2.2.27403.44327.

[41] J. S. Sobolewski. "Data Transmission Media". Em: *Encyclopedia of Physical Science and Technology (Third Edition)*. Ed. por R. A. Meyers. Third Edition. New York: Academic Press, 2003, pp. 277–303. ISBN: 978-0-12-227410-7. DOI: \url{https://doi.org/10.1016/B0-12-227410-5/00165-4}.

[42] S. S. Haykin e B. V. Veen. *Signals and systems*. 2ª ed. John Wiley & Sons, 2002, p. 862. ISBN: 978-0471164746.

[43] P. M. Rodrigues. "Diagnóstico da doença de Alzheimer com base no electroencefalograma". Tese de mestrado. Instituto Politécnico de Bragança, Escola Superior de Tecnologia e Gestão, 2011, p. 117.

[44] J. O. Smith. *Spectral Audio Signal Processing*. Stanford, CA: W3K Publishing, 2011, p. 674. ISBN: 978-0-9745607-3-1.

[45] G. Ballou. *Handbook for Sound Engineers*. 5ª ed. Abingdon: Routledge, 2015, p. 1780. ISBN: 978-0415842938.

[46] F. Walden, S. Dasgupta, M. Rahman e M. Islam. *Improving the Environmental Perception of Autonomous Vehicles using Deep Learning-based Audio Classification*. 2022. arXiv: 2209.04075 [cs.SD].

[47] K. Palanisamy, D. Singhania e A. Yao. *Rethinking CNN Models for Audio Classification*. 2020. arXiv: 2007.11154 [cs.CV].

[48] H. Zhang e D. Wang. "Deep Learning for Acoustic Echo Cancellation in Noisy and Double-Talk Scenarios". Em: *Proc. Interspeech 2018*. 2018, pp. 3239–3243. DOI: 10.21437/Interspeech.2018-1484.

[49] A. Fazel, M. El-Khamy e J. Lee. "Deep Multitask Acoustic Echo Cancellation". Em: *Interspeech 2019*. 2019, pp. 4250–4254. DOI: 10.21437/Interspeech.2019-2908.

[50] H. Zhang e D. Wang. "A Deep Learning Approach to Multi-Channel and Multi-Microphone Acoustic Echo Cancellation". Em: *Proc. Interspeech 2021*. 2021, pp. 1139–1143. DOI: 10.21437/Interspeech.2021-1508.

[51] TensorFlow. *Simple audio recognition: Recognizing keywords*. https://www.tensorflow.org/tutorials/audio/simple_audio. data de consulta: 17/01/2024. 2023.