ESTIMAÇÃO CEGA DO TEMPO DE REVERBERAÇÃO ATRAVÉS DE REDES NEURAIS

Rodrigo Leite Prates

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Mariane Rembold Petraglia

Rio de Janeiro
Março de 2019

ESTIMAÇÃO CEGA DO TEMPO DE REVERBERAÇÃO ATRAVÉS DE REDES NEURAIS

Rodrigo Leite Prates

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Mariane Rembold Petraglia, Ph.D.

Prof. Julio Cesar Boscher Torres, D.Sc.

Prof. Marcio Nogueira de Souza, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2019

Prates, Rodrigo Leite

Estimação Cega do Tempo de Reverberação através de Redes Neurais/Rodrigo Leite Prates. – Rio de Janeiro: UFRJ/COPPE, 2019.

XII, 63 p.: il.; 29, 7cm.

Orientador: Mariane Rembold Petraglia

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 60 – 63.

1. Tempo de Reverberação. 2. Redes Neurais. 3. Curva de Decaimento. I. Petraglia, Mariane Rembold. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Dedico esse trabalho a todos que acreditaram e torceram por mim.

Agradecimentos

Gostaria de agradecer primeiramente a Deus por ter me permitir chegar até aqui. A minha família, cujo apoio foi indispensável para que pudesse completar mais essa etapa em minha vida. Também agradeço a orientação da professora Mariane Petraglia, cujo conhecimento e paciência fizeram desse Mestrado uma realidade. Por fim, agradeço a todos que direta ou indiretamente contribuiram com essa dissertação.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESTIMAÇÃO CEGA DO TEMPO DE REVERBERAÇÃO ATRAVÉS DE REDES NEURAIS

Rodrigo Leite Prates

Março/2019

Orientador: Mariane Rembold Petraglia

Programa: Engenharia Elétrica

Nesta dissertação são investigados os desempenhos das principais técnicas de estimação do tempo de reverberação de um ambiente de forma cega, ou seja, sem o conhecimento a priori da resposta ao impulso da sala. Essas técnicas modelam o sinal reverberante para estimar o decaimento da resposta ao impulso do ambiente, de forma a obter o tempo que o som leva para decair 60 db após a interrupção da fonte sonora, denominado RT_{60}. Entre elas, destaca-se o método baseado em redes neurais, que realiza a predição do RT_{60} a partir de um trecho do sinal reverberante. Esse destaque se deve à sua capacidade de estimação robusta a ruído e a variações abruptas do sinal ao longo do tempo. Os cálculos são feitos no domínio da frequência, onde as características das amostras de treinamento são geradas na escala mel, antes de serem utilizadas pela rede. São apresentados resultados de estimação do RT_{60} para diferentes ambientes, comparando-se os erros médios absolutos obtidos com diferentes técnicas. Pode-se concluir que os erros de estimação obtidos pelos métodos que utilizam redes neurais são semelhantes aos obtidos pelas demais técnicas, sendo esses erros inferiores em algumas configurações de ambiente.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

BLIND ESTIMATION OF REVERBERATION TIME BY NEURAL NETWORKS

Rodrigo Leite Prates

March/2019

Advisor: Mariane Rembold Petraglia

Department: Electrical Engineering

In this dissertation we investigate the performances of the main techniques for blind estimation of the reverberation time of an environment, that is, without the a priori knowledge of the impulse response of the room. These techniques model the reverberant signal to estimate the decay of the ambient impulse response in order to obtain the time that the sound takes to decay by 60 db after the interruption of the sound source, termed RT_{60}. Among them, we highlight the method based on neural networks, which realizes the prediction of the RT_{60} from a stretch of the reverberant signal. This distinction is due to its ability to estimate the RT_{60} robustly to noise and abrupt signal variations over time. Calculations are made in the frequency domain, with the features of the training samples generated on the mel scale, before being used by the network. Results of the RT_{60} estimation are presented for different environments, comparing the average absolute errors obtained with different techniques. It can be concluded that the estimation errors obtained by the methods that use neural networks are similar to those obtained by the other techniques, being these errors lower in some environment configurations.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A estimativa do tempo de reverberação (RT_{60}) é uma tarefa importante na caracterização da qualidade sonora de espaços fechados. Ela contribui para aferição da qualidade e inteligibilidade dos sinais de áudio obtidos de ambientes reverberantes, e para a melhoria dos algoritmos de processamento de áudio e reconhecimento de voz. O parâmetro RT_{60} de um ambiente é definido como o intervalo de tempo que a energia de um som emitido leva para decair 60 dB após a interrupção da fonte sonora. A Figura 1.1 apresenta uma ilustração da estimação deste parâmetro em um ambiente no qual o seu valor é igual a 1.9 s.

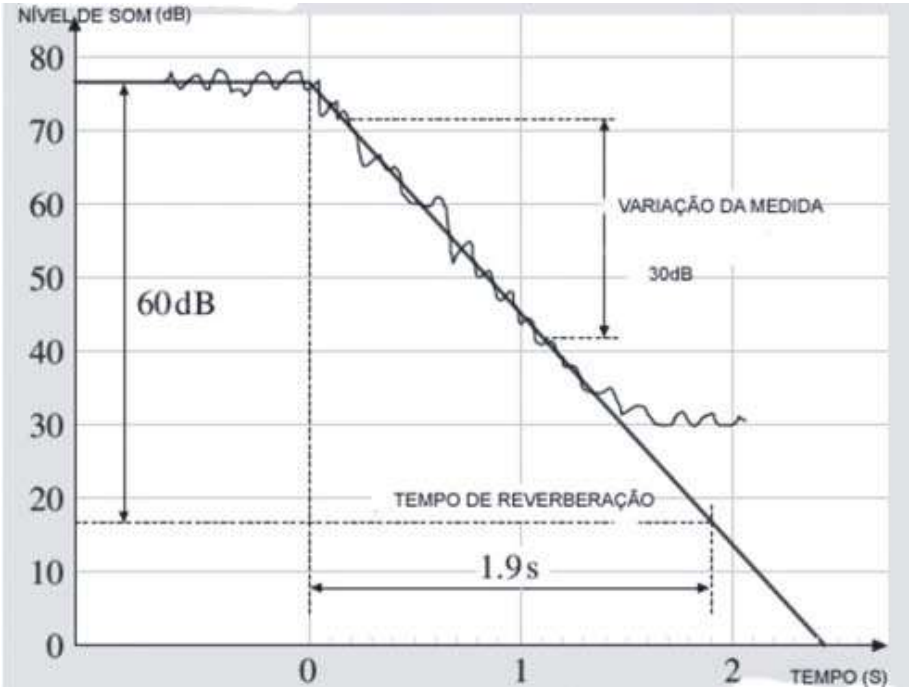

Figura 1.1: Estimação do RT_{60} [1]

O cálculo do RT_{60} de uma sala pode ser realizado através de medidas sobre o nível de pressão gerado por uma fonte sonora. Conforme ilustrado na Figura 1.1,

[1] https://docplayer.com.br/39675644-Acustica-e-educacao-em-musica.html

o cálculo inicia-se quando a queda do nível de pressão sonora é de 5 dB após a interrupção da emissão sonora. Os parâmetros RT_{20} e RT_{30} são obtidos medindo-se os intervalos de tempo em que o nível de pressão sonora leva para diminuir de 20 dB e 30 dB, respectivamente. Assim, o RT_{60} pode ser calculado multiplicando-se por 3 e por 2 os valores de RT_{20} e RT_{30}, respectivamente.

O RT_{60} foi inicialmente calculado por Sabine [1], através das informações de geometria do ambiente e coeficientes de absorção das superfícies que compõem o ambiente. Quando essas informações não estão disponíveis, um sinal de teste controlado pode ser gerado para estimação do RT_{60} baseado no método de interrupção do ruído [2], ou pelo método da curva de integração de Schoreder [3], que utiliza a resposta ao impulso da sala (*Room Impulse Response* - RIR) medida.

Considerando que a RIR pode não estar disponível, e que o sinal de teste pode ser de difícil aplicação, outras forma de cálculo do RT_{60} precisaram ser exploradas. Com isso, em muitas aplicações se faz necessária a determinação do RT_{60} unicamente a partir do sinal de voz gravado. Diversos métodos foram propostos para estimativa do RT_{60}, dentre eles encontram-se os métodos que usam máxima verossimilhança (*Maximum Likelihood* - ML) para modelar o decaimento do sinal reverberante [4], [5] e os métodos que usam a distribuição de decaimento do sinal reverberante (*Spectral Decay Distribution* - SDD), calculada através do envelope de energia da RIR [6]. Há ainda os métodos que usam o domínio da frequência modelando a densidade espectral de potência (*Power Spectral Density* - PSD) através de um filtro IIR, cujo pólo está relacionado ao tempo de reverberação [7]. Por fim, alguns trabalhos utilizam técnicas de aprendizagem de máquinas (*Machine Learning* - ML), tais como redes neurais profundas (*Deep Neural Networks* - DNN) [8], e uma grande quantidade de dados para obter estimativas mais precisas, ou seja, de menor error em relação ao valor original. Essas técnicas já demostraram conseguir significativa melhora na solução de problemas como reconhecimento de voz, reconhecimento de padrões em imagens e estimativa de direção de chegada [2]. O objetivo dessa dissertação é o de tratar o problema de estimação do tempo de reverberação de salas como um problema de mapeamento do sinal reverberante para uma estimativa do RT_{60} usando DNN, de forma semelhante à apresentada em [2]. Neste trabalho, uma rede de classificação é usada para selecionar o RT_{60} mais provável dado um vetor de entrada formado pela concatenação de coeficientes mel [9] de amostras (*frames*) do sinal reverberante. Essa rede foi treinada a partir de uma grande quantidade de sinais de voz reverberantes, gerados através de RIRs simuladas pelo método das imagens [10], e com a adição de ruido branco gaussiano com diferentes razões sinal-ruído (*Signal to Noise Ratio* - SNR). Os resultados obtidos com a implementação do método baseado em DNN são comparados aos alcançados pelos algoritmos considerados "estado da arte". É também realizada a implementação do método PSD, uma abordagem não-cega, ou

seja, que depende do conhecimeto *a priori* do sinal limpo, cujos resultados também são apresentados para comparação. Nesse trabalho o RT_{60} é o mesmo para todas as faixas de frequência, o que é uma simplificação adotada para todos os métodos implementados.

Essa dissertação está organizada como segue. No restante deste capítulo será apresentado o problema de estimação do RT_{60}. No Capítulo 2 será descrita a solução deste problema obtida a partir da densidade espectral de potência (PSD), enquanto que no Capítulo 3 será apresentada a solução gerada via redes neurais profundas (DNN). A metodologia emprega na comparação entre os métodos é descrita no Capítulo 4. No Capítulos 5 é apresentado os resultados obtidos com os diferentes métodos. Por fim, conclusões e trabalhos futuros são discutidos no Capítulo 6.

1.1 Formulação do Problema

Nessa seção será apresentada uma formulação matemática do problema de estimação do RT_{60}, a partir de um sinal de voz gravado de um microfone em um ambiente reverberante. Sejam s, x e d os sinais de voz original, sinal de voz reverberante e sinal de ruído, respectivamente. O sinal y gravado no microfone, a uma certa distância da fonte, pode ser expresso por:

$$y(n) = \alpha s(n - n_0) + x(n) + d(n), \tag{1.1}$$

onde n é o índice de tempo discreto, α é o fator de atenuação e n_0 é o atraso de propagação entre fonte e receptor. Assumindo uma distância específica entre fonte e receptor, n_0 pode ser removido para simplificar o modelo da Eq. (1.1). Com isso, o sinal reverberante x pode ser reescrito como

$$x(n) = h(n - 1) * s(n), \tag{1.2}$$

onde $*$ é o operador convolução e $h(n)$ representa a resposta ao impulso da sala (RIR). De acordo com o modelo de *Polack* [2],[7], a RIR é gerada como uma realização do processo estocástico

$$h(n) = b(n)e^{-\eta n} n \geq 0, \tag{1.3}$$

onde $b(n)$ é um processo estocástico de média zero e distribuição normal, com variância ν^2, que define a estrutura da RIR modulada por uma função exponencial com taxa de decaimento η. Essa taxa é defina como [7]

$$\eta = \frac{3ln(10)}{RT_{60}f_s} \tag{1.4}$$

onde RT_{60} e f_s são o tempo de reverberação e a frequência de amostragem, respectivamente. Para o i-ésimo *frame* e k-ésimo *bin* da transformada discreta de Fourier (*Discrete Fourier Transform* - DFT), podemos reescrever a Eq. (1.1) no domínio da frequência como:

$$Y(i,k) = \alpha S(i,k) + X(i,k) + D(i,k). \tag{1.5}$$

Supondo α e s conhecidos, podemos definir a PSD $Z(i,k)$ da seguinte forma:

$$Z(i,k) = Y(i,k) - \alpha S(i,k) = X(i,k) + D(i,k). \tag{1.6}$$

O método PSD visa calcular $Z(i,k)$ modelando a mesma por um filtro IIR de primeira ordem. A localização do pólo desse filtro está relacionada à taxa de decaimento do sinal reverberante, ou seja, pode ser usado para obter uma estimativa do RT_{60}.

Já o método que utiliza redes neurais também separa o sinal em trechos (*frames*), mas ao invés de usar diretamente essa informação, os trechos são tranformados através de coeficientes da escala de frequência mel. Esses coeficientes foram escolhidos pela referência, em [2], em detrimento de outras escalas, tais como Bark e a *ERB* (*Equivalent Rectangular Bandwidth*). Os coeficientes mel são usados como dados de entrada para o treinamento da rede neural, que irá aprender a estimar o tempo de reverberação associado ao frame. Para isso diversos frames são concatenados até formarem um trecho de sinal suficientemente grande para que a estimativa possa ser realizada.

Serão apresentados mais detalhes de cada método investigado nesta dissertação nos próximos capítulos.

Capítulo 2

Estimação do Tempo de Reverberação pela PSD

Neste capítulo será apresentado o método baseado na densidade espectral de potência (PSD), cuja abordagem requer o conhecimento prévio do sinal sem reverberação, ou seja, é uma metodologia não-cega. Nesse método, a PSD do sinal de voz reverberante pode ser descrita por um modelo IIR de primeira ordem com pólo relacionado ao decaimento da resposta ao impulso do ambiente. Utilizando esse modelo, é possível estimar o tempo de reverberação associado ao sinal de voz reverberante. Além disso, por realizar a estimativa do RT_{60} no domínio da frequência em sub-bandas, esta técnica apresenta maior robustez a ruído quando comparada a técnicas anteriormente propostas. A seguir apresentaremos o algoritmo baseado na PSD, o qual foi descrito em [7].

A energia de um campo sonoro reverberante, em uma sala, decresce de forma exponencial com uma taxa relacionada com o tempo de reverberação (RT_{60}). Conhecimentos a cerca do RT_{60} são bastante usados em técnicas de desreverberação de sinais de áudio e no projeto da arquitetura de salas acústicas, como auditórios e grandes igrejas. Alguns métodos determinam o parâmetro RT_{60} de um ambiente fechado através de equações relacionadas à geometria e à capacidade de absorção de sons pelos objetos. Além disso, o mesmo parâmetro pode ser estimado a partir da envoltória da resposta ao impulso da sala (*Room Impulse Response* - RIR). Nesses métodos, a RIR é obtida através de cuidadosos experimentos e com sinais específicos de excitação, tais como pulsos de ruído ou pulsos que possam se aproximar de um impulso de Dirac [3]. Os métodos acima mencionados são de difícil aplicação prática e implementação em tempo real, pois dependem do conhecimento a prior de diversas informações sobre o sinal de excitação e da sala. Dessa forma, normalmente opta-se pela utilização de métodos cegos ou de métodos que não dependam de tantas informações como as mencionadas.

A técnica baseada na PSD é uma proposta de estimação do tempo de rever-

beração não-cega e *online*, que pode ser usada quando o sinal de voz sem reverberação (limpo) encontra-se disponível. Para o cálculo em tempo real do RT_{60}, a PSD estimada pode ser obtida a partir de um segmento arbitrário do sinal de voz reverberante. Comparando-se a PSD do sinal limpo com a estimada, um valor de RT_{60} pode ser obtido para cada segmento. Dessa forma, não são necessários longos segmentos do sinal reverberante para um cálculo mais preciso do tempo de reverberação.

A organização deste capítulo está dividida da seguinte forma:

1. obtenção da PSD do sinal reverberante;
2. modelagem da PSD através de um filtro IIR;
3. método de estimação da taxa de decaimento do sinal reverberante sem a presença de ruído;
4. método de estimação da taxa de decaimento considerando a presença de ruído.

2.1 Obtenção da PSD do sinal reverberante

Considere o ambiente descrito na Seção 1.1, onde $S(i,k)$, $X(i,k)$, $D(i,k)$ e $Y(i,k)$ são as DFTs do sinal limpo, sinal reverberante, sinal de ruído e sinal gravado, respectivamente, no i-ésimo *frame* e no k-ésimo *bin*(raia da transformada). A PSD de um sinal de voz reverberante, denotada por σ_X^2, pode ser determinada por [7]:

$$\sigma_X^2(i,k) = \sum_{p=0}^{\infty} \nu_{i+\frac{N}{2}}^2 e^{-2\eta_{i+\frac{N}{2}}p} S^2(i-p-1,k), \tag{2.1}$$

onde $S(i-p-1,.)$ é a DFT de um janela do sinal de voz limpo com início na $(i-p-1)$-ésima amostra. Os fatores $\nu_{i+\frac{N}{2}}$ e $\eta_{i+\frac{N}{2}}$ são parâmetros variantes no tempo da RIR, sendo η a taxa de decaimento, de acordo com o modelo de Polack, e N é o comprimento da janela. Por (2.1), pode-se perceber a dependencia não linear com a taxa de decaimento η. A seguir será apresentada uma simplificação do modelo acima, de forma a reduzir o custo computacional na estimativa do tempo de reverberação.

2.2 Modelagem da PSD através de um filtro IIR

A Equação (2.1) pode ser re-escrida da seguinte forma:

$$\sigma_X^2(i,k) = \nu_{i+\frac{N}{2}}^2 S^2(i-1,k) + \sum_{p=1}^{\infty} \nu_{i+\frac{N}{2}}^2 e^{-2\eta_{i+\frac{N}{2}}p} S^2(i-p-1,k). \tag{2.2}$$

Realizando uma mudança de variável, $q = p - 1$, a Equação (2.2) é simplificada para:

$$\sigma_X^2(i,k) = \nu_{i+\frac{N}{2}}^2 S^2(i-1,k) + \sum_{q=0}^{\infty} \nu_{i+\frac{N}{2}}^2 e^{-2\eta_{i+\frac{N}{2}}\, q+1} S^2(i-q-2,k). \tag{2.3}$$

Assumindo que os parâmetros de uma RIR não são variantes no tempo, pode-se remover o sub-escrito $i + \frac{N}{2}$ dos termos $\nu_{i+\frac{N}{2}}^2$ e $\eta_{i+\frac{N}{2}}$. Por fim a Equação (2.3) pode ser re-escrita como segue:

$$\begin{aligned} \sigma_X^2(i,k) &= \nu^2 S^2(i-1,k) \\ &\quad + e^{-2\eta} \sum_{q=0}^{\infty} \nu^2 e^{-2\eta q} S^2(i-q-2,k) \\ &= e^{-2\eta}\sigma_X^2(i-1,k) + \nu^2 S^2(i-1,k). \end{aligned} \tag{2.4}$$

Para que o cálculo seja possível na prática, o somatório na equação (2.1) deve ser limitado, considerando uma RIR de comprimento finito. Para um comprimento Q, temos:

$$\begin{aligned} \sigma_X^2(i,k) = e^{-2\eta}\sigma_X^2(i-1,k) + \nu^2 S^2(i-1,k) \\ - \nu^2 e^{-2\eta(Q+1)} S^2(i-Q-2,k). \end{aligned} \tag{2.5}$$

Para um valor de Q suficientemente grande, pode-se discartar o último termo da Equação (2.5). Dessa forma, a PSD fica representada, para cada instante de tempo e para cada *bin* da DFT, como um filtro IIR de primeira-ordem com polo em $e^{-2\eta}$, ou seja,

$$\sigma_X^2(i,k) = e^{-2\eta}\sigma_X^2(i-1,k) + \nu^2 S^2(i-1,k). \tag{2.6}$$

2.3 Método de estimação da taxa de decaimento do sinal reverberante sem a presença de ruído

Considere um segmento curto de um sinal de voz reverberante, que começa no n-ésimo índice de tempo discreto com comprimento de N amostras, isto é, $x(n), x(n+1), \cdots, x(n+N-1)$. Esse segmento é dividido em janelas de L amostras ($L < N$), com deslocamento de 1 amostra entre janelas. Seja $\sigma_X^2(m,k)$ a PSD da m-ésima janela, para $1 \leq m \leq M$. De acordo com a Equação (2.4), a estimativa

da taxa de decaimento pode ser obtida por

$$\hat{\eta}(j) = \min_{\eta} f(\eta, j), \tag{2.7}$$

onde $f(\eta, j)$ é a função objetivo a ser minimizada, dada por

$$f(\eta, j) = \sum_{k=1}^{N_F/2+1} \sum_{m=2}^{M} e^2(m, k), \tag{2.8a}$$

$$e(m, k) = \sigma_X^2(m, k) - e^{-2\eta}\sigma_X^2(m-1, k) - \nu^2 S^2(m-1, k), \tag{2.8b}$$

onde N_F é o número de coeficientes da DFT e j é o número do segmento. Igualando-se a derivada da função objetivo, dada na Equação (2.8a), em relação a η a zero, obtém-se:

$$\hat{\eta}(j) = -0.5log \left\{ \frac{\sum_{k=1}^{\frac{N_F}{2}+1} \sum_{m=2}^{M} \sigma_X^2(m-1, k)[\sigma_X^2(m, k) - \nu^2 S^2(m-1, k)]}{\sum_{k=1}^{\frac{N_F}{2}+1} \sum_{m=2}^{M} \sigma_X^4(m-1, k)} \right\} \tag{2.9}$$

Utilizando-se $\sigma_X^2(i, k)$ na Equação (2.9), pode-se obter diretamente o valor da taxa de decaimento. Entretanto, normalmente o que temos é uma estimativa da PSD do sinal reverberante e ruidoso $\hat{\sigma}_X^2(i, k)$. Assim, para o cálculo do decaimento a partir da estimativa da PSD, os seguintes passos são realizados:

1. uma estimativa inicial é calculada a partir de um número J de segmentos consecutivos do sinal reverberante, usando a Equação (2.9);

2. a próxima estimativa para o j-ésimo segmento é obtida por alguma medida estatística, como a média ou a mediana, do histograma gerado pelas estimativas iniciais, isto é, $\hat{\eta}_{pri}(j), \hat{\eta}_{pri}(j-1), \cdots, \hat{\eta}_{pri}(j-J+1)$;

3. a estimativa final é obtida a partir da estimativa anterior, suavizada através de um filtro de média recursivo de primeira ordem, com constante de tempo igual a 0.996, ou seja,

$$\hat{\eta}_{fin}(j) = 0.996\hat{\eta}_{fin}(j-1) + 0.004\hat{\eta}_{sec}(j). \tag{2.10}$$

Ainda que o procedimento acima possa remover estimativas espúrias, o uso de filtragem adaptativa permite uma redução considerável no viés das estimativas, além de gerar maior estabilidade. Aplicando-se o método de gradiente descendente na minimização da função objetivo da Equação (2.8a), a atualização da j-ésima estimativa

da taxa de decaimento η é dada por:

$$\hat{\eta}(j) = \hat{\eta}(j-1) - 2\mu \sum_{m=2}^{m=M} \sum_{k=1}^{N_F/2+1} e(m,k) \frac{\partial e(m,k)}{\partial \eta}, \tag{2.11}$$

sendo $\frac{\partial e(m,k)}{\partial \eta} = 2e^{-2\eta}\hat{\sigma}_X^2(m-1,k) \geq 0$ e μ a taxa de aprendizado. A seguinte modificação da Equação (2.11) foi proposta em [7]:

$$\hat{\eta}(j) = \hat{\eta}(j-1) - \frac{\mu}{\nu^2 Var\{s(n)\}} \sum_{m=2}^{m=M} \sum_{k=1}^{N_F/2+1} e(m,k). \tag{2.12}$$

Essa modificação, que normaliza a taxa de aprendizado μ por $\nu^2 Var\{s(n)\}$, é similar a utilizada no método *Normalized Least Mean Squares* (NLMS) para ter a taxa de aprendizado constante para todos os níveis de potência do sinal de entrada [11]. Além disso, pode-se observar que utilizar o sinal (sempre positivo) da derivada ($\frac{\partial e}{\partial \eta}$) ao invés do seu valor causa uma maior estabilidade no algoritmo adaptativo da Equação (2.11). Por isso, é feita essa segunda modificação, que substitui a derivada do erro pelo seu sinal. Essa alteração também reduz a complexidade computacional da Equação (2.11). As estimativas finais das taxas de decaimento são, então, obtidas seguindos os três passos mencionados acima.

2.4 Método de estimação da taxa de decaimento considerando a presença de ruído

Para um sinal de voz ruidoso reverberante, $\hat{\sigma}_X^2$ é substituido por $\hat{\sigma}_Z^2$ na Equação (2.8b), conforme notação usada na Seção 1.1. Quanto mais ruidoso for o sinal reverberante, mais ruidosa é a sua PSD estimada, e maiores serão os erros na estimativa de η. Para melhorar a acurácia dessas estimativas, pode-se incorporar um limiar da SNR *a priori* sobre os pontos da PSD. Dessa forma, somente os pontos com SNRs maiores que um certo limiar (SNR_{th}) são utilizados no procedimento de estimativa. A SNR *a priori* do ponto (m,k) da PSD é definida como

$$snr(m,k) = 10log_{10}\frac{\hat{\sigma}_X^2(m,k)}{\hat{\sigma}_D^2(m,k)}, \tag{2.13}$$

onde $\hat{\sigma}_D^2(m,k)$ é uma estimativa da PSD ruidosa e $\hat{\sigma}_X^2(m,k)$ é definida na Equação (2.1). Entretanto, o exato valor de $\hat{\sigma}_X^2(m,k)$ é desconhecido, visto que depende da taxa de decaimento η a ser estimada. Assim, usando uma estimativa aproximada

de $\hat{\sigma}_X^2(m,k)$, a SNR *a priori* aproximada ($asnr$) é definida como:

$$asnr(m,k) = 10log_{10}\frac{\sum_{p=0}^{P-1}\nu^2 S^2(m-p-1)}{\hat{\sigma}_D^2(m,k)}, \tag{2.14}$$

onde P é o comprimento da resposta ao impulso da sala (RIR). O numerador da Equação (2.14) é $\hat{\sigma}_X^2(m,k)$ excluindo o termo exponencial $e^{-2\eta p}$, o que torna a aproximação $asnr$ sempre maior que o valor da snr. Desta forma, a função objetivo da Equação (2.8a) é modificada para:

$$f_{asnr}(\eta,j) = \sum_{(m,k)\in\{(m_a,k_a)|asnr(m_a,k_a)>SNR_{th}\}} e^2(m,k) \tag{2.15}$$

Nessa nova função objetivo, somente os pontos com valores de $asnr$ acima do limiar SNR_{th} são considerados no cálculo.

Capítulo 3

Estimação do Tempo de Reverberação por Redes Neurais

Neste capítulo são apresentados os métodos de estimação cego e supervisionado do tempo de reverberação RT_{60} baseados em redes neurais profundas (DNN). Uma das principais características desse tipo de rede é a grande quantidade de camadas escondidas (intermediárias) que compõem a rede, podendo ser três ou mais camadas. O método mapeia trechos de um sinal de entrada em um valor correspondente à estimativa do tempo de reverberação. É usada uma rede de classificação com várias saídas, onde cada uma representa um possível valor de RT_{60}. Dessa forma, dadas as amostras de entrada, a rede seleciona uma das saídas como a mais provável de ser a classe associada à faixa de valores do tempo de reverberação do ambiente em que o som reverberante foi adquirido, e a partir do índice dessa saída fornece o valor estimado do parâmetro RT_{60}.

A motivação principal para o uso de redes neurais profundas está na sua capacidade de generalização do processo de aprendizagem, modelando de forma eficiente problemas complexos, ao contrário de redes de uma única camada [12]. O método de estimação proposto é ilustrado na Figura 3.1.

Na parte inferior da figura encontram-se os dados de entrada, e na parte superior as possíveis classes associadas às faixas do parâmetro RT_{60}. Entre as camadas de entrada e de saída, existem camadas intermediárias (*Hidden layers*) que são responsáveis por transformações não-lineares das saídas de uma camada para as entradas da camada seguinte.

O detalhamento do método de estimação do RT_{60} usando DNN está dividido da seguinte forma neste capítulo:

1. breve introdução sobre redes neurais e redes profundas;

2. apresentação dos dados de entrada;

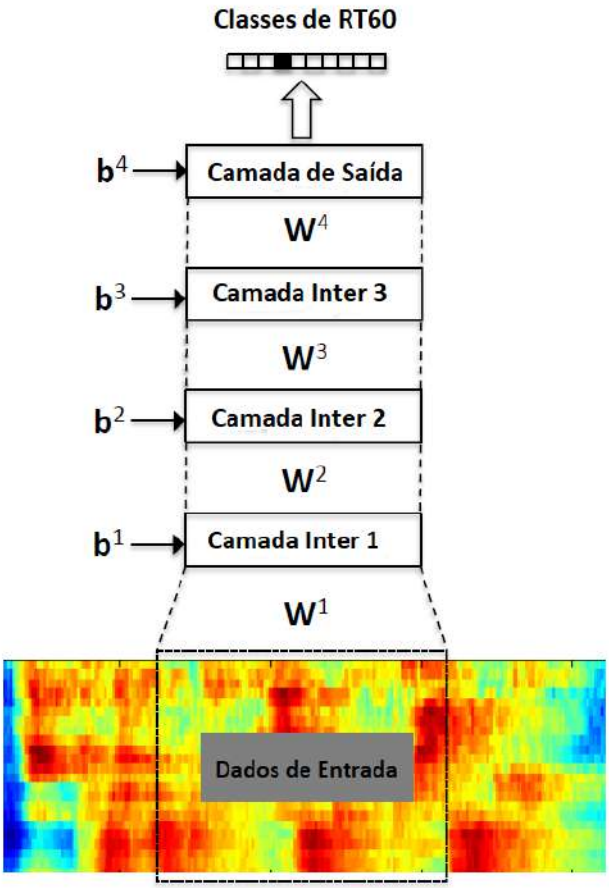

Figura 3.1: Estimação do RT_{60} via DNN (adaptado de [2])

3. apresentação dos dados de saída;

4. descrição do treinamento da rede.

3.1 Introdução a Redes Neurais

O objetivo desta seção é descrever brevemente os principais conceitos relacionados a redes neurais, tais como suas estruturas fundamentais, tipos de treinamentos e técnicas de otimização, finalizando com o arranjo em redes profundas.

3.1.1 Definição

Redes neurais são modelos matemáticos que tentam mimetizar o funcionamento de parte do cérebro quanto ao processamento de informações. O cérebro humano é um sistema altamente complexo, formado por estruturas chamadas neurônios, capazes de processamentos em uma velocidade muito maior que os computadores digitais atuais. Essa capacidade motiva os estudos e avanços na área, na tentativa de modelar tal capacidade.

3.1.2 Neurônio Artificial

Os neurônios artificiais, ou simplesmente neurônios, são as unidades de processamento de informação fundamentais das redes neurais. Tais unidades nada mais são do que modelos artificiais de neurônios biológicos. O primeiro modelo de um neurônio biológico foi proposto pelo fisiologista *Warren McCulloch* juntamente com *Walter Pits*, em 1943, e tratava-se de um modelo simples que descrevia um funcionamento booleano da célula [13]. Este modelo consistia em m terminais de entrada (dentritos) que recebem m valores $x_1, x_2, \ldots, x_m$, que representam as ativações dos neurônios anteriores, e apenas um terminal de saída y (axônio). As sinapses podem ser representadas por pesos $w_1, w_2, \ldots, w_m$, acoplados aos terminais de entrada e de saída. Tais pesos podem assumir valores positivos ou negativos. Assim, o efeito da sinapse de um neurônio i, que recebe um sinal (excitação) x_i, é $w_i x_i$. A excitação total de tal modelo é feito somando todas as sinapses, que corresponde a $\sum w_i x_i$. A ativação ou não do neurônio se dá comparando o resultado das sinapses com o limiar de ativação, a partir de uma função de ativação. A Figura 3.2 ilustra o modelo proposto por *McCulloch* e *Pitts*.

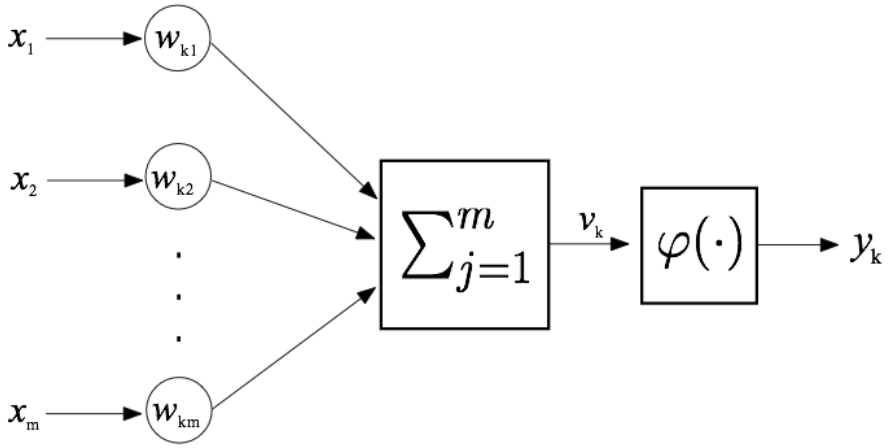

Figura 3.2: Modelo simplificado do neurônio de *McCulloch* e *Pitts*

A função de ativação $\varphi(\cdot)$ tem por objetivo limitar a amplitude do sinal de saída do neurônio. Para o modelo acima, uma função de ativação comumente utilizada é a fiunção limiar, representada matematicamente como:

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases} \tag{3.1}$$

Esse modelo pode ser esquematizado em uma forma mais geral, inserindo-se um polarizador ou *bias* na entrada de cada neurônio. O *bias* tem o efeito de aumentar ou diminuir a entrada da função de ativação. Dessa forma, 4 elementos podem ser identificados nesse modelo geral:

1. Sinapses;

2. Somador;

3. Função de Ativação;

4. *Bias* (polarização).

O neurônio, acrescido do *bias*, pode ser descrito pelas equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.2}$$

$$y_k = \varphi(u_k + b_k), \tag{3.3}$$

em que $x_1, x_2, \ldots, x_m$ são sinais de entrada; $w_{k1}, w_{k2}, \ldots, w_{km}$ são os pesos sinápticos do neurônio k; u_k é a saída do combinador linear; b_k é o *bias*; $\varphi(.)$ é a função de ativação; e y_k é o sinal de saída do neurônio.

A função de ativação pode apresentar diversas topologias de acordo com as funções que a definem. Em geral, as funções restrigem a amplitude de saída do neurônio em um intervalo unitário fechado [0, 1] ou [-1, 1]. Abaixo seguem algumas funções de ativação presentes em [13]:

1. Função Limiar: Função com 2 limiares conforme a Equação (3.1);

2. Função Linear por Partes: função com uma transição entre os limiares, dada por:

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 1/2 \\ v + 1/2, & \text{se } 1/2 > v > -1/2 \\ 0, & \text{se } v \leq -1/2 \end{cases} \tag{3.4}$$

3. Função Sigmóide: função contínua e suave, sendo também diferenciável. Ela pode ser representada pela função logística ou pela função tangente hiperbólica. Fundamentalmente, a diferença entre elas está nos intervalos nos quais são delimitadas. As expressões da função logística e da tangente hiperbólica são apresentadas abaixo, respectivamente:

$$\varphi_l(v) = a \cdot \frac{1}{1 + exp(-bv)} \tag{3.5}$$

$$\varphi_{th}(v) = a \cdot \frac{exp(bv) - exp(-bv)}{exp(bv) + exp(-bv)} \tag{3.6}$$

Uma função comumente utilizada na etapa de pós-processamento de uma rede de classificação é a função do tipo sigmóide *Softmax*. Ela transforma as saídas para cada classe em valores entre 0 e 1, e os divide pela soma das saídas, ou seja,

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \text{ para } j = 1, .., K \tag{3.7}$$

O resultado corresponde essencialmente à probabilidade da entrada estar em uma determinada classe. Por isso, é idealmente usada na camada de saída do classificador. Por exemplo, para $z = [1.2, 0.9, 0.75]$, aplicando a função softmax, obtemos $\sigma(z) = [0.42, 0.31, 0.27]$, que correspondem aos valores de probabilidade das entradas estarem em cada classe.

3.1.3 Arquiteturas de Rede

A arquitetura de uma rede neural é a maneira como seus neurônios estão organizados. Uma forma comum de organização de redes é em camadas, ou seja, os neurônios estão dispostos em uma ou várias camadas. No caso mais simples, tem-se uma única camada, em que os nós de entrada se conectam a uma camada de neurônios de saída. Pode haver ou não realimentação da saída para a entrada. Essa organização está ligada ao algoritmo de aprendizagem usado para o treinamento da rede. Das diversas arquiteturas existentes, podemos destacar:

1. **Redes Neurais *Feedforward* com Camada Única**: é de camada única, pois somente a camada de saída realiza cálculos sobre os dados, ou seja, a camada de entrada somente reproduz os sinais de entrada para cada neurônio da camada de saída. A Figura 3.3 ilustra esse tipo de arquitetura.

2. **Redes Neurais *Feedforward* com Múltiplas Camadas**: neste tipo de arquitetura temos uma ou mais camadas intermediárias entre as camadas de entrada e de saída. Essas camadas intermediárias são referidas como ocultas, ou escondidas. Nesse arranjo, os nós da camada de entrada distribuem o vetor de entradas (sinal de entrada da rede neural) para os neurônios da segunda camada. Os sinais de saída da segunda camada são passados para a terceira camada como sinais de entrada, e assim se repetirá até a camada de saída da rede, sendo que o conjunto de sinais dos neurônios desta última camada constituirá a resposta global da rede para o padrão de ativação fornecido pela camada de entrada. A Figura 3.4 ilustra uma rede com uma camada oculta.

3. **Redes Recorrentes**: uma rede recorrente se distingue das redes anteriores por ter pelo menos um laço de realimentação. Ela pode apresentar neurônios ocultos ou não. A presença de laços de realimentação tem grande impacto

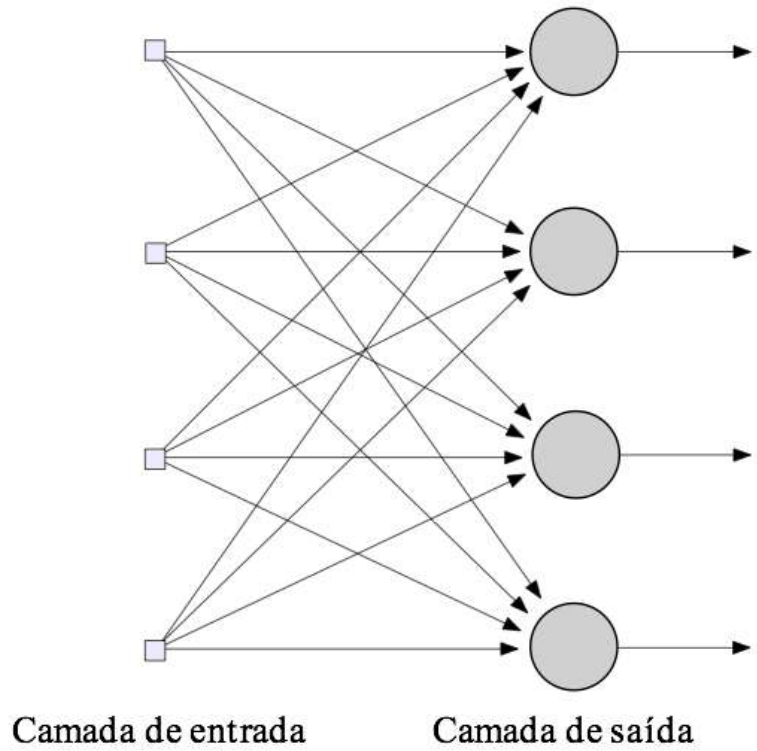

Figura 3.3: Rede alimentada textitfeedforward com uma camada de entrada e uma de saída

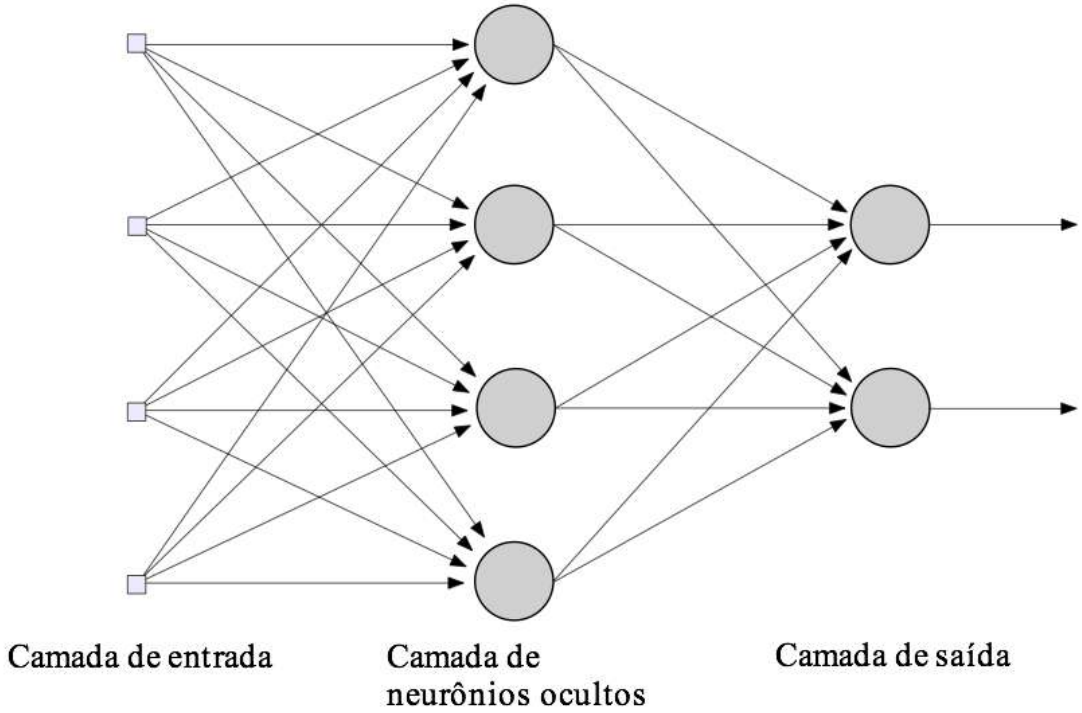

Figura 3.4: Rede alimentada adiante com uma camada oculta de neurônios

na capacidade de aprendizado da rede e no seu desempenho. Além disso, esses laços envolvem o uso de ramos particulares compostos de elementos de

atraso unitário, os quais realimentam alguns neurônios da rede com saídas processadas no passo de tempo anterior. Essas realimentações conferem à rede um comportamento dinâmico não-linear. A Figura 3.5 mostra uma rede com laço de realimentação.

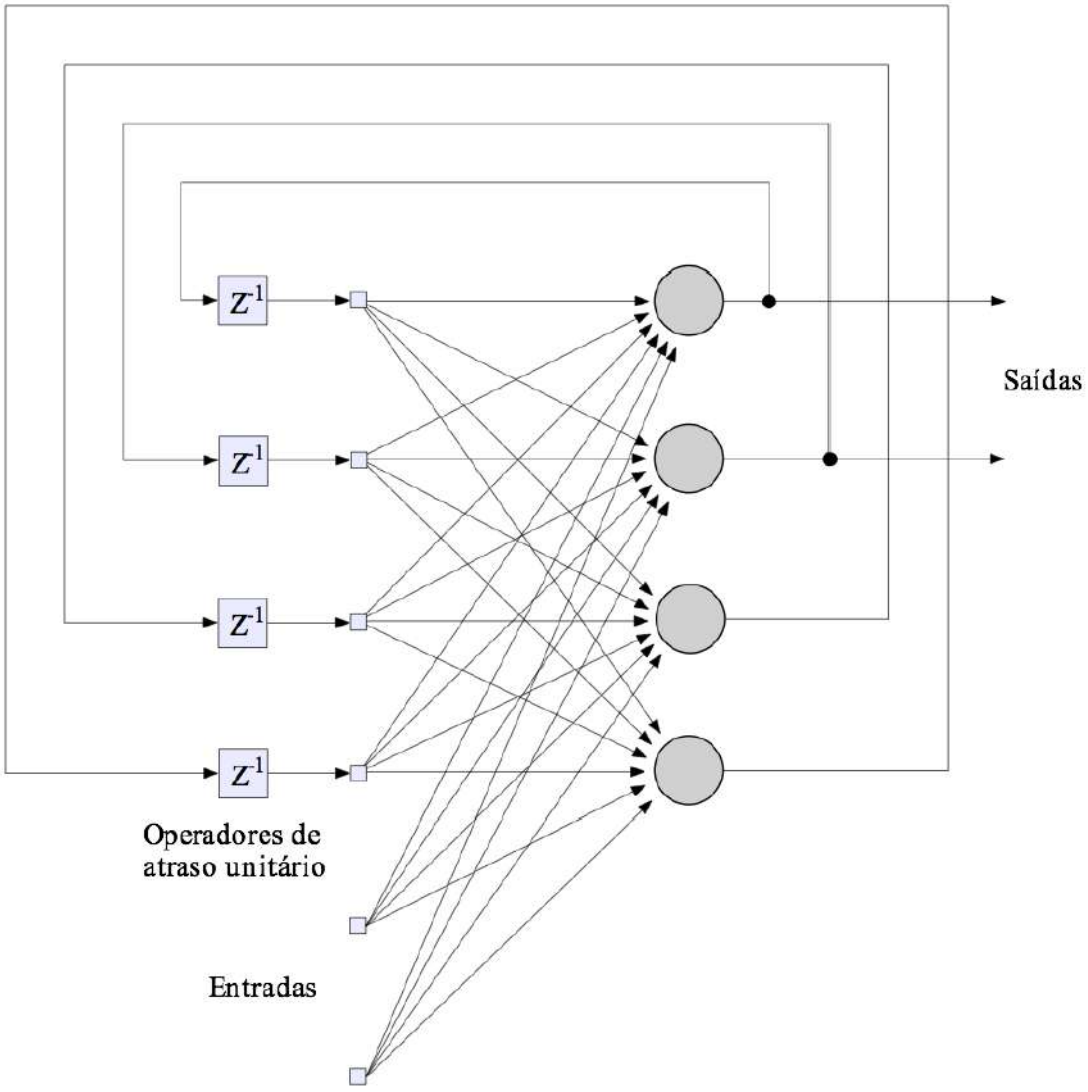

Figura 3.5: Rede recorrente com apenas camadas de entrada e de saída

4. **Redes LSTM**: a LSTM (*Long Short-Term Memory*) é um tipo de rede recorrente usada para aprendizado de longo prazo. Ela foi introduzida por *Hochreiter & Schmidhuber* em (1997) [14], com o objetivo de oferecer um desempenho melhor por resolver o problema de desaparecimento de gradiente que redes neurais recorrentes naturalmente sofrem quando lidam com grandes sequências de dados. De forma geral, toda rede recorrente apresenta um módulo de repetição com uma única camada. Já as redes LSTM possuem mais camadas, e suas unidades são formadas, normalmente, pelos seguintes componentes:

   (a) Célula: armazena uma entrada por um período de tempo;

(b) Portão de Entrada: controla quando um novo valor chega à Célula;

(c) Portão de Saída: controla quando o valor na Célula será usado para calcular a saída da unidade LSTM;

(d) Portão de Esquecer: controla quanto tempo o valor deve permanecer na Célula.

A Figura 3.6 representa uma unidade LSTM.

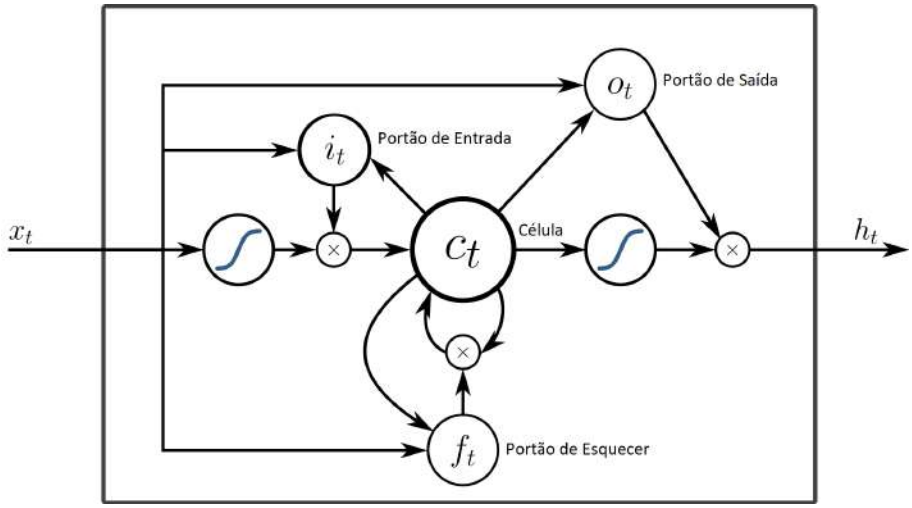

Figura 3.6: Unidade LSTM (obtida de [14])

5. **Redes Convolucionais**: é uma classe de redes *feedforward* muito usadas no processamento e análise de imagens digitais. Essas redes também são conhecidas como redes neurais invariantes ao deslocamento ou invariantes ao espaço. As redes convolucionais (CNNs) exploram a correlação espacial local, impondo um padrão de conectividade entre os neurônios das camadas adjacentes [15]. Em outras palavras, as entradas de neurônios intermediários na camada m são de um subconjunto de unidades da camada $m-1$, conforme ilustrado na Figura 3.7. As redes convolucionais são inspiradas nos processos biológicos.

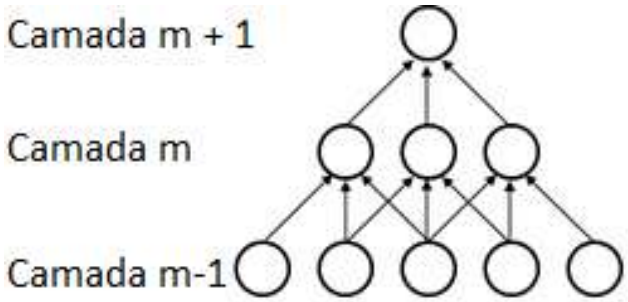

Figura 3.7: Esparsidade na arquitetura de uma rede convolucional

Nelas o padrão de conectividade entre os neurônios é inspirado na organização do córtex visual dos animais. Imagine que a camada $m-1$ é a retina de entrada. Na Figura 3.7, as unidades na camada m têm campos receptivos de largura 3 na retina de entrada e, portanto, são conectados apenas a 3 neurônios adjacentes na camada de retina. As unidades na camada $m+1$ têm uma conectividade semelhante com a camada abaixo. Dizemos que seu campo receptivo em relação à camada abaixo também é 3, mas seu campo receptivo em relação à camada $m-1$ é maior (5). Cada unidade não responde a variações fora de seu campo receptivo em relação à retina. A arquitetura garante, assim, que os "filtros" aprendidos produzam a resposta mais forte a um padrão de entrada espacialmente local.

3.1.4 Processo de Aprendizagem

O processo de aprendizado é uma propriedade fundamental para uma rede neural. Em geral, uma rede aprende acerca dos seus dados de entrada a partir de um processo de ajustes sobre seus pesos sinápticos e *biases*. Idealmente, a rede se torna mais instruída sobre os seus dados de entrada após cada iteração do processo de aprendizado [13].

Os processos de aprendizado podem ser agrupados em dois paradigmas principais: aprendizado supervisionado e aprendizado não-supervisionado [16]. Aprendizado supervisionado implica a existência de um supervisor, o qual é responsável por estimular as entradas da rede por meio de padrões de entrada e observar a saída da rede calculada pela mesma, comparando-a com a saída desejada. Esse tipo de aprendizado é aplicado em problemas em que se deseja mapear padrões de entrada e saída. No aprendizado não-supervisionado, não existe a figura do supervisor e, em geral, apenas os padrões de entrada estão disponíveis. O processo de aprendizado se dá pela existência de regularidades e redundâncias nos padrões de entrada apresentados à rede. Esse processo se aplica em problemas que visam à descoberta de características estatísticas nos dados.

Abaixo segue uma lista com alguns processos de aprendizagem, separando-os dentro dos dois paradigmas, de acordo com [16].

1. Aprendizado supervisionado:
   (a) Aprendizado por correção de erro;
   (b) Aprendizado por reforço.
2. Aprendizado não supervisionado:
   (a) Aprendizado *Hebbiano*;

(b) Aprendizado competitivo.

Existem diversos algoritmos de aprendizado que, em geral, diferem um do outro pela forma como ajustam os pesos sinápticos dos neurônios. No presente trabalho, foi utilizado um algoritmo conhecido como retropropagação (*backpropagation*), o qual faz parte do processo de aprendizado supervisionado por correção de erro. A seguir são apresentados os processos de aprendizado por correção de erro e o algoritmo de *backpropagation*.

Aprendizagem por correção de erro

Este processo baseia-se em alterar os valores dos pesos sinápticos dos neurônios em função da diferença entre os sinais de saída da rede $\mathbf{y}(n,k) = (y_1(n,k), \ldots, y_i(n,k), \ldots, y_m(n,k))$ e os valores esperados $\mathbf{d}(k) = (d_1(k), \ldots, d_i(k), \ldots, d_m(k))$, em que o índice n indica a n-ésima iteração no processo de aprendizagem, k é o índice da k-ésima amostra de conjunto de treinamento e m é o número de neurônios na camada de saída. Essa diferença recebe o nome de sinal de erro $\mathbf{e}(n,k) = (e_1(n,k), \ldots, e_i(n,k), \ldots, e_m(n,k))$, definido como:

$$\mathbf{e}(n,k) = \mathbf{d}(k) - \mathbf{y}(n,k) \tag{3.8}$$

O sinal de erro $\mathbf{e}(n,k)$ é usado para aplicar uma sequência de ajustes corretivos aos pesos sinápticos de cada neurônio. Os ajustes são projetados para aproximar passo a passo o sinal de saída $\mathbf{y}(n,k)$ da resposta desejada $\mathbf{d}(k)$. Isto é realizado através da minimização de uma função de custo $\xi(n,k)$, definida como:

$$\xi(n,k) = \frac{1}{2}(\mathbf{e}(n,k) \cdot \mathbf{e}(n,k)) \tag{3.9}$$

sendo $\xi(n,k)$ o valor instantâneo da energia do erro na n-ésima iteração (ou época) para a k-ésima amostra do conjunto de treinamento. Destaca-se que para contabilizar uma iteração é necessário que todas as amostras do conjunto de treinamento sejam apresentadas à rede no processo de treinamento. Assim, na n-ésima iteração, todas as amostras do conjunto de treinamento foram apresentadas $n-1$ vezes à rede. Ao final da n-ésima iteração, todas as amostras foram apresentadas n vezes.

A energia média do erro quadrado é obtida somando-se os $\xi(n,k)$ de todo o conjunto de treinamento. Considerando T como sendo a dimensão da amostra de treinamento, tem-se a energia média do erro quadrático na n-ésima iteração:

$$\xi_{med}(n) = \frac{1}{T}\sum_{k=1}^{T}\xi(n,k) \tag{3.10}$$

Os ajustes dos pesos de cada neurônio continuam até o sistema atingir um estado estável, isto é, os pesos estão de tal forma estabilizados que o valor da energia do erro situa-se num ponto de mínimo (local ou global) na superfície de erro. Idealmente, o processo de treinamento deveria se encerrar quando a energia do erro atingisse seu mínimo global. Porém, isso nem sempre é possível devido às não-linearidades presentes na rede. Apesar disso, existem técnicas de treinamento que buscam levar a rede a se aproximar do ponto de mínimo global.

A minimização da função de custo é dependende do processo ou algoritmo de aprendizado utilizado. Na sequência, será descrito maiores detalhes dessa minimização, de acordo com o algoritmo de *backpropagation*.

Aprendizagem de retropropagação (*backpropagation*)

As energias instantânea $\xi(n,k)$ e média $\xi_{med}(n)$ do erro são funções de todos os pesos sinápticos e *biases* dos neurônios da rede. Para um dado conjunto de treinamento, $\xi_{med}(n)$ representa a função custo como uma medida do desempenho de aprendizagem. O objetivo do processo de aprendizagem é ajustar os parâmetros livres (pesos sinápticos e *biases*) para minimizar $\xi_{med}(n)$. A média aritmética destas alterações individuais dos pesos sobre o conjunto de treinamento é, portanto, uma estimativa da alteração real que resultaria da modificação dos pesos baseada na minimização da função de custo ξ_{med} sobre o conjunto de treinamento inteiro. A seguir será descrita a expressão de correção dos pesos sinápticos e a sequência de passos para realizá-la. A correção $\Delta w_{ji}(n)$ aplicada ao peso sináptico $w_{ji}(n)$ é definida pela regra delta:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{ji}(n)} \tag{3.11}$$

em que η é o parâmetro taxa de aprendizagem e j representa o índice do j-ésimo neurônio da camada de saída cujo i-ésimo peso sináptico está sendo corrigido. O sinal negativo indica o decréscimo do gradiente no espaço de pesos. O ajuste do peso w_{ji} é realizado como:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) \tag{3.12}$$

A Equação (3.12) pode ser escrita como:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \tag{3.13}$$

em que $\delta_j(n)$ é chamado gradiente local e $y_i(n)$ é o sinal de saída do i-ésimo neurônio

pré-sináptico ao neurônio j. O gradiente local $\delta_j(n)$ é definido como:

$$\delta_j(n) = e_j(n)\varphi_j^{'}(v_j(n)) \tag{3.14}$$

em que $e_j(n)$ é o sinal de erro do neurônio j e $\varphi_j^{'}(v_j(n))$ é a derivada da função de ativação em relação à $v_j(n)$.

O gradiente local é um estimador que aponta para uma possível direção de modificações necessárias a serem efetuadas sobre os pesos sinápticos. Conforme a Equação (3.14), o gradiente local $\delta_j(n)$ para o neurônio de saída j é igual ao produto do sinal de erro $e_j(n)$ pela derivada $\varphi_j^{'}(v_j(n))$ da função de ativação associada a esse neurônio.

Pelas Equações (3.13) e (3.14) temos que um fator chave no cálculo do ajuste de peso $\Delta_{ji}(n)$ é o sinal de erro $e_j(n)$ na saída do neurônio j. Neste contexto, identificam-se dois casos distintos dependendo de onde o neurônio j está localizado. No primeiro caso, o neurônio j é um nó de saída. No segundo caso, o neurônio j é um nó escondido ou oculto. Para cada caso, o cálculo de $\Delta w_{ji}(n)$ assume formas diferentes, conforme são descritas abaixo:

1. Primeiro Caso. Quando o neurônio j está localizado na camada de saída da rede, ele recebe um resposta desejada particular. Podemos utilizar a Equação (3.8) para calcular o sinal de erro $e_j(n)$ associado a esse neurônio. Com isso, calcula-se diretamente o gradiente local $\delta_j(n)$, usando a Equação (3.14).

2. Segundo Caso. Quando o neurônio j está localizado em uma camada oculta da rede, não existe uma resposta desejada para aquele neurônio. Sendo assim, para calcular o gradiente local, é necessário retropropagar os sinais de erros da rede. O cálculo do gradiente local $\delta_j(n)$, partindo da função de custo definida na Equação (3.9), é dado por:

$$\delta_j(n) = \varphi_j^{'}(v_j(n)) \sum_{k=1}^{m} \delta_k(n) w_{kj}(n) \tag{3.15}$$

   em que k é o índice dos neurônios da camada subsequente à camada do neurônio j, isto é, são os neurônios que recebem o sinal de saída do neurônio j. O valor m é o número total de neurônios presentes nessa camada, os quais estão ligados ao neurônio j por meio dos pesos $w_{kj}(n)$.

Em suma, o algoritmo de retropropagação é definido como a sequência de dois passos principais:

1. Propagação: nessa etapa os pesos sinápticos se mantêm inalterados em toda a rede e os sinais funcionais são calculados individualmente, neurônio a neurônio.

Ou seja, a propagação começa desde a camada de entrada, passando pelas camadas ocultas, e terminando na camada de saída, onde são calculados os sinais de erro;

2. Retropropagação: essa etapa começa na camada de saída, a partir da qual os ajustes dos pesos são enviados para as camadas ocultas através dos cálculos dos gradientes locais $\delta_j(n)$ para cada neurônio oculto. Todos os pesos sinápticos dos neurônios são ajustados de acordo com as expressões definidas pela regra delta. Após o término do passo de retropropagação, inicializa-se o passo de propagação, recomeçando assim o algoritmo. Esse processo recursivo permite que os pesos sinápticos sofram modificações de acordo com a regra delta. Os passos de propagação e de retropropagação podem ser realizados amostra por amostra do conjunto de treinamento. Uma outra forma de aplicar o algoritmo é realizar o passo de propagação sobre todas as amostras do conjunto de treinamento e somente depois realizar o passo de retropropagação. Essa forma é conhecida como modo por lote de treinamento e apresenta poucas diferenças no cálculo da correção $\Delta w_{ji}(n)$ em relação ao modo anterior, conhecido como modo sequencial.

Um termo que pode ser acrescentado na Equação (3.12), com o intuito de aumentar a taxa de aprendizagem e evitar instabilidade na convergência, é um termo de momento. A equação de atualização dos pesos é dada por:

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) + \alpha w_{ji}(n-1) \tag{3.16}$$

onde α é normalmente um número positivo chamado de constante de momento. A Equação (3.16) é conhecida como regra delta generalizada.

3.1.5 Função Custo

A função custo tem como objetivo calcular uma métrica do erro cometido pela rede, permitindo obter informações sobre seu desempenho e ajustar os seus coeficientes. Gera um valor que informa quanto adequada está a rede frente às amostras usadas no seu treinamento ou na sua avaliação. Essas funções costumam depender da natureza do problema a ser resolvido, tais como: problemas de classificação, regressão, e redução de dimensionalidade. Além disso, para o cálculo dessas funções, é necessário o uso dos seguintes parâmetros: pesos da rede, *bias*, amostras de entrada e de saída.

Algumas funções comumente utilizadas são:

1. Erro médio quadrático (MSE);

2. Erro médio absoluto (MAE);

3. Entropia cruzada (*Cross-Entropy*).

A função custo MSE, apresentada na Equação (3.9), é uma função quadrática normalmente usada em problemas de regressão e classificação. Sua principal característica é a simetria, isto é, erros positivos ou negativos geram o mesmo resultado.

Já a função por erro médio absoluto tem a seguinte forma:

$$\xi(n,k) = \frac{1}{2}|\mathbf{e}(n,k)|, \tag{3.17}$$

onde o termo quadrático foi substituído pelo valor absoluto do erro. Esse função, tal como a anterior, não faz distinção se o erro foi sub-estimado ou sobre-estimado.

Por fim, para o caso da entropia cruzada temos:

$$\xi(n,k) = -\frac{1}{2}\left[y(n,k)\ln(d(k)) + (1-y(n,k))\ln(1-d(k))\right]. \tag{3.18}$$

A Equação (3.18) apresenta duas propriedades relevantes para uma função custo. A primeira é que ela é sempre positiva, e a segunda é que se a saída real do neurônio estiver próxima da saída desejada para todas as k entradas de treinamento, então a entropia cruzada será próxima de zero. Por exemplo, suponha que $y(n,k) = 0$ e $d(k) \approx 0$ para alguma entrada $x(n)$. Nesse caso, vemos que o primeiro termo da Equação (3.18) desaparece, enquanto o segundo termo é $ln(1 - d(k)) \approx 0$. Uma análise semelhante é valida quando $y(n,k) = 1$ e $d(k) \approx 1$. Dessa forma, a contribuição para a função custo é baixa, desde que a saída real esteja próxima da saída desejada. Em suma, a entropia cruzada é positiva e tende a zero, à medida que o neurônio melhora a computação da saída desejada, y, para todas as entradas de treinamento x. Essas propriedades são também observadas pelas funções custo quadráticas, como a função *MSE*. A diferença está na capacidade da função de entropia cruzada de evitar o problema de desaceleração do aprendizado [17].

Calculando a derivada da função custo entropia-cruzada C, dada na Equação (3.18), em relação a um peso j qualquer, temos:

$$\frac{\partial C}{\partial w_J} = -\frac{1}{n}\sum_x \left(\frac{y}{\phi(z)} - \frac{(1-y)}{1-\phi(z)}\right)\frac{\partial \phi}{\partial w_j} \tag{3.19}$$

$$= -\frac{1}{n}\sum_x \left(\frac{y}{\phi(z)} - \frac{(1-y)}{1-\phi(z)}\right)\phi'(z)x_j. \tag{3.20}$$

onde $d(k)$ na Equação (3.18) foi substituído pela função de ativação $\phi(z)$, n é o número total de amostras de entrada e $\sum_x$ é o somatório sobre todas as amostras

de entrada. A Equação (3.19) pode ser reescrita como

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x \frac{\phi'(z) x_j}{\phi(z)(1-\phi(z))} (\phi(z) - y) \tag{3.21}$$

Usando a definição da função sigmóide, $\phi(z) = 1/(1+e^z)$, pode-se mostrar que a Equação (3.21) pode ser simplificada para:

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j (\phi(z) - y) \tag{3.22}$$

Pela Equação (3.22), a taxa na qual o peso j é alterado é dependente do erro na saída do neurônio correspondente, ou seja, $\phi(z) - y$. Assim, quanto maior o erro, mais rápido é o aprendizado. Isto é uma característica muito útil para uma função custo. Em particular, evita o efeito da desaceleração do aprendizado, causada pelo pequeno valor que pode assumir o termo $\phi'(z)$ que aparece na função custo quadrática para atualização dos pesos. Isso pode acontecer, por exemplo, quando o erro obtido para atualização dos pesos é muito alto. Por outro lado, o termo $\phi'(z)$ é cancelado no caso da entropia cruzada. Existe uma equação análoga à Equação (3.22) para o caso de atualização do *bias*. Essa última propriedade da função custo entropia cruzada só é observada caso a função de ativação usada seja a função sigmóide.

Ainda com relação à função custo entropia cruzada, em [18] é enfatizado o melhor resultado obtido quando se utiliza essa função custo ao invés do erro de classificação, para treinamento de redes de classificação. Por exemplo, consideremos uma rede para classificar dentre 3 classes possíveis, com neurônios de saída pós-processados pela função *softmax*. Supondo ainda somente 3 amostras para treinamento, cujos valores de saída correspondentes são dados na Tabela 3.1. Conforme indicado nesta tabela, essa rede teve erro de classificação igual a $1/3 = 0.33$, ou acurácia de $2/3 = 0.67$. Pode-se notar que os resultados de classificação dessa rede foram corretos para as duas primeiras amostras, mas com saídas próximas dos limiares de classificação, e que o resultado para a terceira amostra foi incorreto e com saídas distantes dos valores esperados.

Tabela 3.1: Resultados de classificação da Rede 1

calculados	esperados	correto?
0.3 0.3 0.4	0 0 1 (democrata)	sim
0.3 0.4 0.3	0 1 0 (republicano)	sim
0.1 0.2 0.7	1 0 0 (outro)	não

Na Tabela 3.2 é mostrado o resultado de outra rede aplicada às mesmas 3 amostras do exemplo anterior. Essa outra rede também obteve erro de classificação igual a $1/3 = 0.33$. Por outro lado, os acertos foram por valores bem mais próximos dos valores esperados.

Tabela 3.2: Resultado de classificação da Rede 2

calculado	esperado	correto?
0.1 0.2 0.7	0 0 1 (democrata)	sim
0.1 0.7 0.2	0 1 0 (republicano)	sim
0.3 0.4 0.3	1 0 0 (outro)	não

Para as duas redes acima, considerando o cálculo do erro por entropia cruzada, temos:

$$ACE_1 = \frac{-\left(ln(0.4) + ln(0.4) + ln(0.1)\right)}{3} = 1.38 \tag{3.23}$$

$$ACE_2 = \frac{-\left(ln(0.7) + ln(0.7) + ln(0.3)\right)}{3} = 0.64 \tag{3.24}$$

onde ACE_i significa erro médio de entropia cruzada (*average cross-entropy error*), e i o índice referente à rede. Pelos resultados acima, percebe-se que o erro é menor para a segunda rede neural, o que era de se esperar.

Podemos realizar o mesmo cálculo usando a função custo erro médio quadrático *MSE*:

$$MSE_1 = \frac{(0.54 + 0.54 + 1.34)}{3} = 0.81 \tag{3.25}$$

$$MSE_2 = \frac{(0.14 + 0.14 + 0.74)}{3} = 0.34 \tag{3.26}$$

Pode-se observar que o erro médio quadrático também corresponde a uma métrica de avaliação de desempenho melhor que o simples erro de classificação. Por outro lado, comparando com o erro de entropia cruzada, esse último gera uma maior diferença entre os resultados das redes, o que é preferível para a seleção das mesmas. Dessa forma, para problemas de classificação, é recomendado o uso da função custo entropia cruzada, juntamente com a função sigmóide nos neurônios de saída da rede.

Existem diversas outras funções custos, não avaliadas neste trabalho, que podem ser utilizadas, tais como: função custo exponencial, máxima verossimilhança, distância de *Hellinger*, divergente de *Kullback–Leibler* e distancia de *Itakura–Saito*.

3.2 Dados de Entrada

Sabe-se que a percepção humana ao conteúdo de frequência dos sons não segue uma escala linear [19]. Estudos a respeito dessa percepção geraram diversas escalas psicoacústicas, tais como: mel, bark, erb. A escala mel é uma escala de percepção de tons (*pitchs*), que reflete a percepção relativa de dois sons de alturas distintas. É uma das escalas mais usadas em sistemas de identificação de usuário por voz. O mapeamento usado para converter uma frequência em *Hertz* (f_{lin}) na escala mel é dado por:

$$F_{mel} = 2595 \times log_{10}(1 + \frac{f_{lin}}{700})$$

A escala bark é outra escala psicoacústica, baseada em medidas subjetivas de intensidade sonora. É dividida em 24 bandas, que correspondem às bandas críticas da audição [20]. O mapeamento das frequências é obtido através da seguinte equação:

$$F_{bark} = 13 \arctan(0.00076 f_{lin}) + 3.5 \arctan((\frac{f_{lin}}{7500})^2)$$

Por último, a escala erb (*Equivalent Rectangular Bandwidth*) é uma escala que fornece uma aproximação das larguras de banda dos filtros observados no sistema de audição humana, os quais são modelandos como filtros passa-banda retangulares [21]. A equação a seguir define um dos possíveis mapeamentos das frequências:

$$F_{erb} = 6.23 \times 10^{-6} f_{lin}^2 + 9.339 \times 10^{-2} f_{lin} + 28.52$$

Na Figura 3.8 são apresentadas as respostas em frequência dos bancos de filtros das três escalas. Essas escalas costumam ser usadas no pré-processamento de sinais, para geração dos dados de entrada para os algoritmos de processamento de áudio, tais como os que fazem a estimação do tempo de reverberação. Como o presente trabalho não tem como objetivo avaliar o impacto no uso das diferentes escalas, optou-se por usar unicamente a escala mel, tal como definido em [2].

Assim, como parâmetros de entrada, foram utilizados os coeficientes do produto entre o sinal de entrada e um banco de filtros na escala mel (*Log Mel Filterbank Energies* - LMFBE) [2, 6, 8].

Os passos para geração das *features*, ou dados de entrada, segue abaixo conforme [2, 9]. Essa proposta assume que os sinais estão todos amostrados na taxa de 16 kHz.

1. Inicialmente é aplicado um filtro de pré-enfase no sinal de voz reverberante, com coeficiente $\alpha = 0.97$, ou seja,

$$y(n) = x(n) - \alpha x(n-1);$$

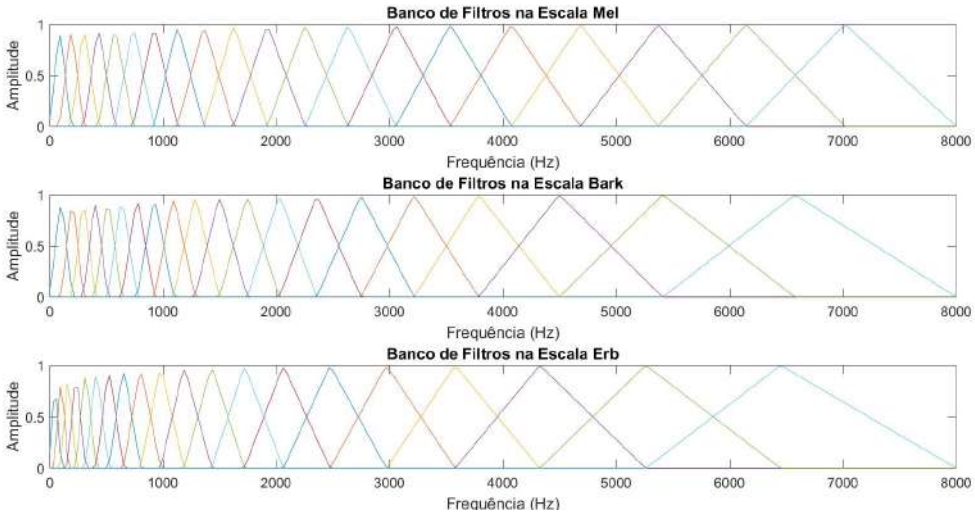

Figura 3.8: Respostas em Frequência dos Bancos de Filtros das Escalas Psicoacústicas Mel, Bark e Erb

2. Em seguida o sinal é dividido em frames de 25ms, com overlap de 40%;

3. A cada frame é aplicada a janela de Hamming,

$$w(n) = 0.54 - 0.46cos(\frac{2\pi n}{N-1}),$$

sendo N o tamanho da janela;

4. Aplica-se a transformada rápida de Fourier (*Fast Fourier Transform* - FFT) de comprimento 512 a cada frame, também chamada de STFT (*Short Time Fourier Transform*). Feito isso, calcula-se o espectro de potência pela equação:

$$P = \frac{|FFT(x_i)|^2}{N},$$

onde x_i é o i-ésimo frame do sinal x;

5. Na parte final, aplica-se um banco de 23 filtros triangulares T_{256x23}, de 256 coeficientes, ao espectro de potência de cada frame. Tornando cada frame F_{1x23} um array de 23 coeficientes, ou seja,

$$F_{1x23} = f_{1x256} * T_{256x23};$$

6. Ainda a cada frame, aplica-se o logaritmo de base 10 e remove-se a média para o balanceamento do espectro e melhoria na relação sinal ruído [9].

Na Figura 3.9 são apresentados os coeficientes LMFBE de um sinal de voz limpo e de um reverberante, este último gerado a partir da filtragem do sinal limpo pela res-

posta ao impulso simulada de uma sala considerada grande, de dimensão 10 m × 7 m, e com distância entre a fonte e o receptor considerada pequena, de 1, 5 m (próximo). Essa denominação para a configuração de um experimento (grande/próximo) a partir dos parâmetros "tamanho e distância" será empregada no próximo capítulo.

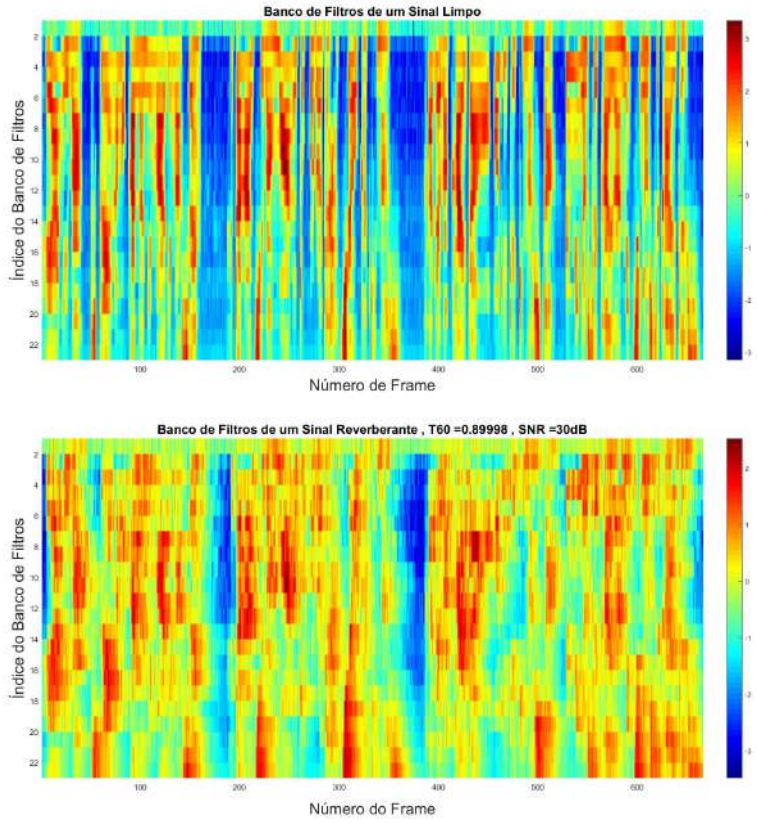

Figura 3.9: Coeficientes LMFBE de 10 s de sinais de voz limpo e reverberante

Das Figuras 3.9 e 3.10, pode-se observar que a reverberação causa um espalhamento no tempo e na frequência dos padrões do sinal de voz. Para realizar o mapeamento entre os coeficientes LMFBE e o tempo de reverberação RT_{60}, é necessário concatenar uma quantidade suficiente de frames. Dessa forma, é criada uma sequência de 51 frames contínuos, como vetor de entrada para a DNN. Essa quantidade de frames concatenados é proporcional ao maior tempo de reverberação que se deseja estimar.

3.3 Dados de saída

Com os dados de entrada selecionados, o próximo passo é escolher os dados de saída para o treinamento da DNN. É natural tratar a estimação com um problema de regressão. A DNN pode ser treinada para minimizar o erro médio quadrático (MSE) entre o RT_{60} predito e o verdadeiro RT_{60} do conjunto de treinamento. Entretanto, conforme [2], essa solução por regressão tende a gerar altos erros de estimação nos

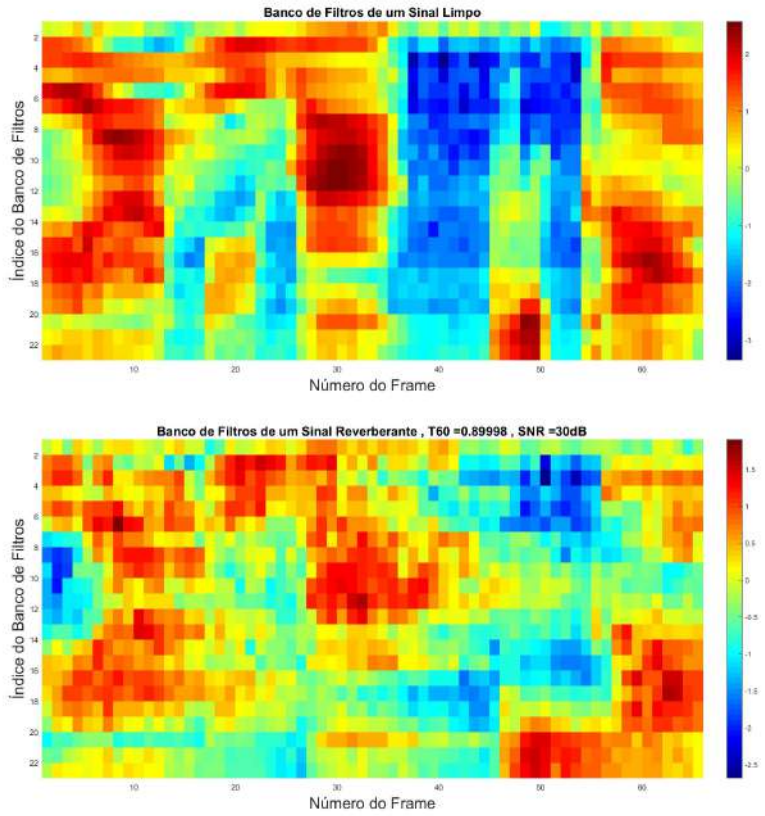

Figura 3.10: Coeficientes LMFBE de 1 s de sinais de voz limpo e reverberante

limites da escala do RT_{60} a ser estimado. Por exemplo, quando uma DNN é treinada para estimar tempos variando entre 0.1s a 1.0s, a mesma tende a super-estimar o RT_{60} quando o valor verdadeiro é baixo, e sub-estimar quando o valor verdadeiro é alto. Assim, o valor predito é enviesado ao redor do centro da escala do tempo de reverberação de treinamento. Esse problema é resultado direto da escolha pela regressão como forma de tratar o problema, e do uso do MSE como função custo de treinamento da DNN. Para evitar isso, podemos tratar a estimação como um problema de classificação, onde dividimos os valores de RT_{60}, na escala de 0.1 s a 1.0 s, em 19 bins, um por classe para cada bin. Assim a DNN é treinada para estimar a classe na qual o RT_{60} pertença. Temos então, por exemplo, o seguinte vetor de saída para o caso do valor 0.1 s ser o RT_{60} verdadeiro:

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T \tag{3.27}$$

3.4 Treinamento

Com os parâmetros de entrada e saída da rede definidos, a DNN é treinada para aprender o melhor mapeamento dos valores da entrada para a saída. A arquitetura da rede é tal como ilustrado na Figura 3.1. Nela, o vetor de entrada passa

por uma transformação afim através da matriz de pesos $\mathbf{W}^1$ e o vetor de bias $\mathbf{b}^1$ primeiramente, e em seguida passa pela função de ativação sigmóide para gerar a saída da primeira camada escondida. Essa saída se torna a entrada para mais uma transformação afim por $\mathbf{W}^2$ e $\mathbf{b}^2$, e passa pela função de ativação para gerar a saída da segunda camada escondida. Esse processo se repete igualmente para a terceira camada escondida. Finalmente, no topo do modelo, a camada de saída gera o vetor de saída da DNN usando a função softmax. Dessa forma, a saída é um vetor de probabilidades *a posteriori*:

$$\mathbf{p_t} = [p(1|\mathbf{o_t}), \ldots, p(N|\mathbf{o_t})]^T \tag{3.28}$$

onde o N é o número de RT_{60} classes/bins considerado no sistema, $p(i|\mathbf{o_t})$ é a probabilidade *a posteriori* do i-ésimo bin dada a feature de entrada $\mathbf{o_t}$, sendo t o índice do frame. O valor do RT_{60} é estimado por

$$\hat{RT}_{60} = c_j \tag{3.29}$$

$$j = \arg\max_i p(i|\mathbf{o_t}), i \in [1, N] \tag{3.30}$$

onde c_i é o centro do i-ésimo bin. Na prática, a DNN gera um vetor de probabilidade *a posteriori* para cada frame de voz. Como o RT_{60} é calculado para segmento (N frames) de voz, a estimativa da Equação (3.29) é uma média de probabilidades sobre um segmento.

Existe um compromisso entre a largura do bin e o número de classes. Quanto maior a largura do bin, mais acurada é a classificação, mas pior é a resolução da estimativa do RT_{60}. Em [2] foram usados 19 bins para cobrir o intervalo de 0.1 s a 1.0 s, com bin de largura 0.05 s. Conforme será visto no próximo capítulo, foi necessário realizar uma pequena alteração nessa escala para trabalhar no intervalo de 0.2 s a 1.0 s, ainda usando 19 bins. Isso se deve a problemas na geração da resposta ao impulso (RIR), pelo metodo das imagens [10], para o valor de RT_{60} de 0.1 s. Assim, para a nova escala, a largura do bin passou a ser de 0.042 s. Uma largura ainda menor foi usada em [8], sem que isso inviabiliza-se a estimação.

Para melhorar a resolução da estimação, a estimativa final do RT_{60} é obtida da seguinte forma:

$$\hat{RT}_{60} = \frac{\sum_{i=j-1}^{j+1} p(i|\mathbf{o_t}) c_i}{\sum_{i=j-1}^{j+1} p(i|\mathbf{o_t})}, \tag{3.31}$$

que corresponde a uma soma ponderada dos centros dos bins do RT_{60} ao redor da classe estimada.

Capítulo 4

Metodologia e Simulações

Nesse capítulo será apresentada a metodologia usada e os experimentos simulados para a avaliação dos algoritmos. Inicialmente, entretanto, serão detalhados o ambiente de desenvolvimento usado e os procedimentos para geração dos dados de treinamento e de teste.

4.1 Ambiente de desenvolvimento

Para esse trabalho foi utilizado um notebook Dell com a seguinte configuração:

1. Windows 10 Home Basic de 64 bits;
2. Processador I7-6700HQ 2.6 GHz;
3. 16 GB de memória RAM;
4. 1 TB de HD híbrido com 8 GB de memória flash;
5. Placa de Video NVIDIA GeForce GTX 960M 4 GB.

Essa configuração permitiu a execução das simulações e armazenamento dos dados, sem travamentos ou gargalos.

Todas as simulações foram criadas e executadas através do software de simulação Matlab R2016b e compilador C++ MinGW 4.9.2. O Matlab disponibiliza uma vasta quantidade de ferramentas para *Machine Learning* e *Signal Processing*. Para a geração das RIRs, foi necessária a utilização de uma função chamada *rir-generator* [22], desenvolvida em C++. Logo, para sua utilização no Matlab, foi necessário a integração de um compilador C++ ao mesmo [23, 24]. Os códigos desenvolvidos estão na forma de funções e scripts.

4.2 Geração do Banco de Dados

Nesta seção será detahada a geração do banco de dados usado nas simulações.

4.2.1 Seleção do Banco de Sinais de Voz

Foi gerado um conjunto de dados para treinamento e teste dos algoritmos de estimação do RT_{60}. Em [2] foram extraídas 7.861 sentenças sem reverberação, emitidas por 92 pessoas da base WSJCAM0 [25]. Como essa base teria que ser adquirida, não foi possível o seu uso neste trabalho. As seguintes bases puderam ser obtidas gratuitamente com todos os seus dados:

1. VCTK-Corpus: base do *Centre for Speech Technology Voice Cloning Toolkit* [26], formada por áudios gravados de 109 pessoas, falando 400 frases cada, totalizando 43.600 arquivos. Os arquivos estão no formato wav, amostrados a 96 KHz e com duração variando de 3 a 10 segundos;

2. TIMIT-Database: base da *DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus* [27], formada por 6.300 arquivos de áudio de sentenças gravadas por 630 pessoas, 10 sentenças por pessoa. Os arquivos estão no formato wav, amostrados a 16 KHz, e com duração variando de 4 a 6 segundos. Pode ser obtida em versão reduzida com somente 460 sentenças [28].

Devido à superioridade em quantidade de arquivos, duração dos áudios e simplicidade na organização dos arquivos por usuário, optou-se por usar unicamente a base VCTK-Corpus.

Após a seleção da base de dados de sinais de voz, foi necessário definir como seria criado o banco de respostas ao impulso (RIRs) que gerariam os sinais reverberantes a partir dos sinais limpos do banco de dados, considerando valores variados de tempo de reverberação. Para isso, optou-se por gerar essas RIRs através do conhecido método das imagens [22]. A descrição dos parâmetros desse método e da metodologia criada para seu uso segue abaixo.

4.2.2 Geração das Respostas ao Impulso

Para a geração das Respostas ao Impulso das salas (RIRs), foi utilizada uma função, em C++, que usa o método das imagens para esse fim [22]. Essa função disponibiliza uma série de parâmetros:

1. Velocidade do som: c em m/s;

2. Frequência de amostragem: f_s em $samples/s$;

3. Posição do receptor: r $[x\ y\ z]$ em m;
4. Posição da fonte: s $[x\ y\ z]$ em m.
5. Dimensões da sala: d $[x\ y\ z]$ em m;
6. Tempo de reverberação: RT_{60} em s;
7. Número de amostras: n;
8. Coeficientes de reflexão [29];
9. Tipo de microfone;
10. Orientação do microfone;
11. Aplicação ou não de um filtro passa-altas.

Desses parâmetros, optou-se por utilizar somente os sete primeiros, visto que além dessas informações serem as únicas disponíveis no artigo de referência [2], a utilização destes foi suficiente para permitir a geração de RIRs com tempos de reverberação muito próximos dos valores desejados. Para essa validação, a RIR gerada foi pós-processada através do método da curva de integração de Schoreder, o que permitiu verificar a proximidade do tempo desejado (t_{60}) com um tempo de referência, o que também foi usado em [6, 7].

Foi utilizada uma metodologia para variar os parâmetros do método das imagens, e assim gerar diferentes RIRs a serem convoluidas com os sinais de áudio. Nesse procedimento foram definidas três faixas de valores de áreas distintas de salas (pequena, média e grande) e duas distâncias possíveis entre fonte e receptor (próximo e distante).

Para a velocidade do som e frequência de amostragem, fixaram-se os valores $c = 340$ m/s e $f_s = 16$ kHz, respectivamente. As RIRs foram geradas para uma sala retangular, cujo coeficiente de reflexão é dado pela equação [22]:

$$r = 1 - \frac{24Vlog(10)}{cSRT_{60}}, \tag{4.1}$$

onde V é o volume da sala em m^3 e S a superfície da sala em m^2. A Tabela 4.1 apresenta os valores dos coeficientes de reflexão, para cada valor de RT_{60} e tamanho de sala.

Tabela 4.1: Coeficientes de Reflexão das Salas para cada RT_{60}

RT_{60}	Pequena	Média	Grande
0.2	0.5124	0.3905	0.1755
0.2444	0.6010	0.5013	0.3254
0.2889	0.6624	0.5780	0.4292
0.3333	0.7074	0.6343	0.5053
0.3778	0.7419	0.6773	0.5635
0.4222	0.7690	0.7113	0.6095
0.4667	0.7910	0.7388	0.6467
0.5111	0.8092	0.7615	0.6774
0.5556	0.8245	0.7806	0.7032
0.6	0.8375	0.7968	0.7252
0.6444	0.8487	0.8108	0.7441
0.6889	0.8584	0.8230	0.7606
0.7333	0.8670	0.8338	0.7751
0.7778	0.8746	0.8433	0.7880
0.8222	0.8814	0.8517	0.8097
0.8667	0.8875	0.8593	0.8097
0.9111	0.8930	0.8662	0.8190
0.9556	0.8979	0.8724	0.8274
1.0	0.9025	0.8781	0.8351

Para dimensão dessa sala, foram utilizadas 3 configurações possíveis: sala pequena $d = [4, 3, 4]$ m, sala média $d = [6, 4, 4]$ m e sala grande $d = [10, 7, 4]$ m. Já para as posições de fonte s e receptor r, partiu-se de uma posição inicial $[2, 2, 2]$ m para ambas. Em seguida, foi escolhida aleatoriamente uma orientação a ser alterada, podendo ser a x ou a y. Também foi escolhida, de forma aleatória, a distância a ser usada entre fonte e receptor, podendo ser de $1,5$ m (próxima) ou 3 m (distante). Por fim, esse deslocamento foi aplicado à posição do receptor, verificando-se a dimensão da sala, de forma a que o receptor não fique em uma posição inválida. Para isso, a fonte também pode ser deslocada, o que garante que ambos estejam dentro da sala, independente do tamanho da mesma.

Por esse procedimento, verificamos que as alterações podem ocorrer tanto na coordenada x quanto na y, enquanto z, que corresponde à altura da sala, é sempre mantida constante. Isso gerou uma grande variabilidade nas RIRs geradas, o que beneficia o treinamento e teste dos algoritmos.

4.2.3 Geração dos Dados de Treinamento e Teste

Em seguida, para a geração dos dados de treinamento e teste, convencionou-se trabalhar com trechos de voz de 3 segundos de duração de cada arquivo de áudio. Para evitar possíveis inconsistências no banco de dados, foi gerada uma lista com os nomes de todos os arquivos de áudio válidos, isto é, os que atendiam aos seguintes critérios:

1. sinais com extensão correta '.wav';
2. sinais com duração mínima de 4 segundos;
3. sinais que, ao se remover o trecho inical de silêncio, tinham duração de pelo menos 3 segundos.

Essa lista permitiu avaliar a quantidade de dados disponíveis para a criação dos conjuntos de treinamento e teste dos algoritmos.

Com a aplicação dos filtros acima, obteve-se um total de 9.586 arquivos de voz de 152 pessoas distintas. Dessa quantidade de arquivos, separou-se aproximadamente 5% (479 arquivos) para compor o conjunto de teste. Esse conjunto foi montado tanto com sinais de voz de pessoas que apareceram no treinamento como com pessoas que não apareceram. Isso contribuiu para a avaliação da capacidade de generalização dos algoritmos.

Dessa forma, temos um total de 9.107 arquivos para treinamento das redes. A implementação do treinamento utilizada realiza esse procedimento separando os dados, por padrão, em 70% para treinamento, 15% para validação e 15% para teste [30]. Isso contribui para evitar um viés no treinamento das redes.

Assim, a partir da lista com os nomes dos arquivos de áudio, as seguintes etapas foram realizadas para a geração dos dados de entrada e saída dos algoritmos para cada arquivo de voz:

1. reamostrar o sinal de voz para a taxa de 16 kHz;
2. obter um trecho contínuo de 3 segundos de sinal de voz. Para isso, verificou-se a maior amplitude do sinal, e calculou-se 10% desse valor, o que consideramos como limiar de silêncio. Excluiu-se todo o trecho de voz inicial, cujo valor é inferior ao limiar calculado. O trecho subsequente de 3 segundos é então extraído do sinal original;
3. realizar a convolução do trecho de sinal anterior com a RIR gerada pelo método das imagens, obtendo o sinal reverberante. Nessa etapa os seguintes parâmetros são utilizados pela função geradora da RIR, escolhidos de forma aleatória para cada sinal de voz:

(a) RT_{60}, cuja escolha deve respeitar a escala de valores possíveis, de 0.2 a 1.0 segundo, com intervalo de 0.042 s entre cada bin/classe;

(b) dimensão da sala, podendo ser pequena, média ou grande;

(c) distância entre fonte e receptor, podendo ser próxima (1,5m) ou distante (3 m).

4. dividir o sinal reverberante pelo seu valor absoluto máximo, de forma a escalar o sinal para a faixa de $[-1, 1]$;

5. adicionar ruído branco gaussiano ao sinal reverberante, cujo nível em dB é escolhido também de forma aleatória, mas de acordo com os seguintes valores: $[0, 5, 10, 20, 30]$ dB para sinais usados no treinamento, e $[-10, 0, 10, 20, 30]$ dB para sinais usados no teste;

6. aplicar o procedimento explicado nas Seções 3.2 e 3.3 a cada sinal reverberante e ruidoso. Essa etapa é realizada 5 vezes, sobre o mesmo sinal, de forma a gerar ainda mais dados para o treinamento e o teste.

Como mencionado acima, alguns parâmetros foram escolhidos de forma aleatória na geração das RIRs. Entretanto, para garantir o balanceamento por classe na quantidade de diferentes valores de RT_{60}, optou-se por utilizar a mesma quantidade para todas as classes. Logo, um vetor balanceado com toda a lista de valores de RT_{60} foi criado e usado na seleção do tempo de reverberação para geração das respostas ao impulso. Dessa forma, foi possível garantir que não haveria predomiâancia em quantidade de amostras de uma classe em relação às demais.

Por fim, quando todos os áudios foram processados, os dados de entradas e saída foram salvos juntamente com as seguintes informação para cada sinal:

1. SNRs usadas na inclusão de ruído;

2. dimensões de sala;

3. distância entre fonte e receptor;

4. valor de RT_{60}.

Após esse procedimento, são armazenados em arquivo 136.590 vetores de entrada de 1.173 valores. Isso gera uma matriz de entrada de 1.173×136.590. Já a matriz de saída é de dimensão 19×136.590, visto que temos 19 possíveis classes. Para a validação, geramos uma matriz de 1.173×7.185 amostras de entrada e, consequentemente, uma matriz de saída de 19×7.185 amostras.

Após a geração dos dados para treinamento e validação dos algoritmos, foi aplicada uma metodologia para validação dos resultados, a qual será apresentada a seguir.

4.3 Calibração dos Modelos e Comparações

A metodologia usada nesse trabalho é dividida em duas partes: primeiramente é realizado uma série de procedimentos para calibração dos modelos de redes neurais propostos e seleção do melhor modelo. Em seguida, esse modelo de rede é comparado ao método SDD, a uma rede DNN de referência e ao algoritmo de PSD, sendo essa última análise objetivando verificar a diferença entre métodos cego e não-cego.

4.3.1 Treinamento e comparação entre os modelos

Em [2] foi utilizada para estimação do RT_{60} uma rede *feedfoward* com 3 camadas escondidas de h neurônios, uma camada de entrada com n neurônios, e uma de saída com 19 neurônios. A função custo usada é a MSE, e as funções sigmóide e softmax são usadas para ativação dos neurônios das camadas escondidas e da camada de saída, respectivamente. Por falta de informação sobre a arquitetura da rede acima, foram levantados os seguintes questionamentos:

1. quantos neurônios n usar na cama de entrada;
2. quantos neurônios h usar em cada camada escondida;
3. qual algoritmo usar para atualização dos pesos e *bias*;
4. por que usar função custo MSE, e não *CrossEntropy*, visto o melhor desempenho esperado pela última para uma rede de classificação, conforme visto na Seção 3.1.5;
5. usar algum tipo de regularização [11];
6. aplicar algum tipo de pré-processamento (normalização) aos dados de entrada.

A partir desses questionamentos e testes iniciais realizados, decidiu-se utilizar a seguinte arquitetura:

1. número de neurônios da camada de entrada igual ao tamanho do vetor de entrada, ou seja, 1.173 neurônios;
2. número de neurônios na faixa $[25, 400]$ em cada uma das 3 camadas escondidas, sendo o valor ótimo obtido via calibração da rede;
3. atualização dos pesos e bias pelo algoritmo *backpropagation with momentum and adaptive learning rate* [31];
4. funções de erro *MSE* e *CrossEntropy*, sendo a melhor definida no processo de calibração da rede;

5. taxa de aprendizado definida na calibração da rede;

6. norma L^2 usada na regularização [32], a qual está disponível na implementação do matlab da rede neural;

7. não foi aplicada normalização aos dados, nem no treinamento nem na validação, devido ao escalamento do sinal reverberante em etapa anterior e por não ter sido observado ganho no seu uso em simulações iniciais.

Antes de iniciar a calibração, foi realizada uma avaliação para verificar se a quantidade de dados seria suficiente para o treinamento das redes. Sabe-se que, a princípio, não existe como afirmar que um certo número de amostras corresponde à quantidade mínima necessária para o treinamento de uma rede. Isso se deve ao fato de que esse valor depende de diversas variáveis, tais como: arquitetura da rede, número de neurônios, dimensão das amostras, número de saídas, etc. Por isso, resolveu-se avaliar a relação entre a quantidade de amostras de treinamento (136.590) e os valores de: número de pesos ou sinápses e o número de classes utilizadas (19) [33].

Assim, para uma rede de 25 neurônios nas camadas escondidas, temos um total de 31.144 sinápses. Já para uma rede de 50 e 100 neurônios, temos um total de 64.769 e 139.519 sinápses, respectivamente. Considerando essa relação (número de sinápses pelo número de amostras) é razoável não ultrapassar o número de 100 neurônios, já que teriamos mais sinápses que amostras nesse caso.

Por outro lado, considerando a relação entre o número de amostras e o número de saídas (ou classes), temos aproximadamente 7.189 amostras por classe, o que provavelmente é suficiente para a que rede possa aprender a distinguir entre as classes, conforme [33].

Para a calibração dos parâmetros dessa arquitetura, tais como número de neurônios das camadas escondidas, tipos de função custo, valor da taxa de aprendizado e valor da regularização, um conjunto de simulações foram planejadas e realizadas para a obtenção dos valores ótimos de cada parâmetro.

Objetivando ainda uma comparação entre as redes do tipo *MSE* com as de *CrossEntropy*, os demais parâmetros foram testados, de igual forma para ambos os tipos de rede, até que a melhor configuração fosse obtida para cada uma delas. Por fim, os resultados para essas configurações foram novamente comparados para a escolha final. O melhor resultado considerado no processo de calibração das redes corresponde ao que gerou o menor MSE. Já para comparação das redes propostas com os métodos de referência, será usada como métrica o erro médio absoluto (*Mean Absolute Error* - MAE), de acordo com [2] e [7]. O desvio médio absoluto (*Mean Absolute Deviation* - MAD) também será calculado como forma de apresentação dos resultados comparativos com os métodos considerados "estado da arte".

O procedimento de calibração inicia com a busca pela melhor taxa de aprendizado lr, para um total de 5 diferentes configurações de rede. Cada configuração difere pela quantidade de neurônios h nas camadas escondidas. Dessa forma, por exemplo, para cada valor de lr são realizadas 5 simulações, uma para cada valor de h. São os seguintes os valores para h: $[25, 50, 100, 200, 400]$. Esses foram escolhidos considerando que o número de neurônios nas camadas escondidas não deve ser menor que o número de neurônios na camada de saída, e que para valores maiores que 400 neurônios, o tempo de simulação passa a ser impraticável para a realização de tantas simulações em um único computador. Além disso, para valores maiores que 400 neurônios, não se identificou nenhuma redução na taxa de erro. Após a calibração de lr, é feita a inclusão da regularização λ, utilizando o melhor valor de lr encontrado e os 3 melhores valores de h. Por fim, após a realização desses procedimentos para os dois tipos de função custo, *MSE* e *CrossEntropy*, a melhor seleção de parâmetros é obtida para cada um deles.

Em relação ao lr, o algoritmo de *backpropagation with momentum and adaptive learning rate* permite a atualização automática desse parâmetro, através do aumento ou diminuição do mesmo baseado em limiares. Isso, por outro lado, não significa que o valor inicial de lr não venha a influenciar o resultado do treinamento da rede. Dessa forma, essa técnica permite selecionar outros parâmetros, além de lr, para o treinamento da rede. Essas variáveis e a dinâmica de atualização de lr são apresentadas na equação [34]:

$$lr(n) = \begin{cases} lr(n-1) + lr(n-1) \cdot lr\_inc, & \text{se } err(n-1) < goal \\ lr(n-1) - lr(n-1) \cdot lr\_dec, & \text{se } err(n-1) \geq err(n-2) * err\_inc \end{cases} \tag{4.2}$$

onde $lr(n)$ é a taxa de aprendizado na n-ésima amostra, lr_inc e lr_dec são os valores de incremento e decremento, respectivamente, aplicados à taxa de aprendizado. O $err(n)$ é o erro ou função custo. Dessa forma, o incremento só ocorre caso o erro de treinamento anterior, $err(n-1)$, seja menor que o erro mínimo especificado, $goal$. Já o decremento, por sua vez, só ocorre caso o erro anterior tenha aumentado de um percentual err_inc.

A atualização dos pesos e *bias* ocorre da seguinte forma:

$$dX(n) = mc \cdot dX(n-1) + lr(n) \cdot mc \cdot \frac{derr(n-1)}{dX} \tag{4.3}$$

onde mc é o momento e $\frac{derr(n-1)}{dX}$ é a derivada do erro em relação a X. Aqui, X representa tanto um dos peso como um dos *bias*.

Na busca pelo valor ótimo de lr, utilizou-se inicialmente $mc = 0.9$. Já para

lr_inc e *lr_dec*, usou-se $lr\_inc = lr\_dec = 0$, inicialmente. Após isso, os valores foram alterados para $lr\_inc = 1.005$ e $lr\_dec = 0.5$, de forma a se otimizar o desempenho da rede treinada (*tunning*). Assim, esse procedimento foi usado somente nas simulações relativas à variação de lr, e foi mantido durante a calibração da regularização. Quanto aos valores de lr, percebeu-se que quanto maior esse parâmetro, mais rápida é a convergência do erro no treinamento, isto é, menor é a quantidade de épocas para se atingir um patamar de mínimo. Por outro lado, quanto menor o valor de lr, mais lenta é a convergência. Além disso, para $lr > 1$ e $lr < 10^{-4}$, a convergência não ocorre e se torna muito lenta, respectivamente. Por isso, decidiu-se testar lr para os seguintes valores: $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$. Como mencionado acima, buscou-se o valor ideal de lr antes de aplicar o *tunning*, que atualiza automaticamente lr, a cada época, otimizando o resultado. Para essa etapa de simulações, tivemos um total de, pelo menos, $\#lr \times \#h = 20$ simulações de treinamentos/testes de redes. Na verdade, mais simulações necessitaram ser feitas para confirmar alguns resultados, sendo o mesmo feito nas demais etapas que se seguem.

Como mencionado anteriormente, após a escolha de lr, são selecionadas 3 configurações de rede ($\#h$), para serem usadas na escolha da regularização. Isso ajuda a reduzir o número de simulações a serem executadas. Em seguida, um procedimento semelhante foi realizado para obtenção do valor ótimo do parâmetro regularização λ, que por padrão é usado $\lambda = 0$. Para valores de regularização maiores que $\lambda > 1$, a convergência não corre ou se torna muito lenta, sem aparente melhora. Isso pode ser entendido se considerarmos a expressão da regularização usada:

$$E_k = \frac{1}{2N}\sum_x (t_k - o_k)^2 + \frac{\lambda}{2N}\sum_w w_i^2 \tag{4.4}$$

onde E_k é o erro do k-ésimo neurônio, t_k e o_k são respectivamente os valores esperado e de saída desse neurônio, w_i é o i-ésimo peso associado ao neurônio e N é o número de amostras.

Por essa equação, podemos perceber que quanto maior for o valor de λ, menor será o valor do somatório $\sum_w w_i^2$ para que o erro diminua. Essa redução nos pesos, em excesso, pode prejudicar a convergência da rede. Por esse motivo, decidiu-se por utilizar os mesmos valores que na seleção de lr para λ: $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$. Aqui, a princípio, tivemos uma quantidade de $\#\lambda \times 3 = 15$ simulações para cada uma das duas funções custo. Selecionamos, então, o par λ e h que forneceu o melhor resultado.

Segue, em resumo, os passos da metodologia usada tanto para redes com função *MSE* como para *CrossEntropy*:

1. realizar o treinamento da rede proposta, variando o número de neurônios das camadas escondidas (h) dentre os valores $[25, 50, 100, 200, 400]$, e variando a

taxa de aprendizado (lr) dentre os valores $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$;

2. para a melhor rede de cada configuração de h, isto é, a rede de menor MSE, aplicar $lr\_inc = 1.005$ e $lr\_dec = 0.5$ no procedimento de *tunning* da Equação (4.2);

3. a partir dessas simulações, selecionar lr que fornece menor MSE, e os três valores de h que também fornecem os menores valores de MSE;

4. calibrar a regularização λ, variando esse parâmetro dentre os valores $[10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$, usando as 3 configurações de h e a lr selecionadas anteriormente; essas simulações são realizadas mantendo o *tunning*;

5. a partir dessas simulações, selecionar o λ que gera o menor MSE e os três valores de h que também fornecem os menores valores de MSE;

6. verificar se o uso da regularização gerou melhores resultados;

7. selecionar a melhor configuração para cada método (*MSE* e *CrossEntropy*).

Por fim, os dois modelos de rede são comparados, através do erro médio absoluto, para diferentes configurações de sala, distância entre fonte e receptor e nível de rúido, de forma a se selecionar o melhor modelo e poder comparar o mesmo aos métodos SDD e rede DNN de referência. Essa comparação é realizada da mesma forma, gerando resultados de erro médio absoluto (MAE) para as diferentes configurações de sala, distância fonte-receptor e nível de ruído, analisando-os tanto no formato de tabelas como na forma de histogramas.

4.3.2 Comparação com o método PSD

Para a comparação com a PSD, algumas adaptações foram necessárias. Dessa forma, seguem as configurações usadas, destacando-se as mudanças em relação ao método apresentado em [7]:

1. Taxa de amostragem $f_s = 16$ kHz para a rede neural, sendo usado $f_s = 8$ kHz no método PSD;

2. Número de amostras por segmento $N = 1024$;

3. Número de segmentos $J = 1000$, com deslocamento de 3 ms entre segmentos, correspondendo a um total de aproximadamente 3 s de sinal; foi usado $J = 500$ na PSD;

4. Número de intervalos contidos nos J segmentos: $I = 5$; assim, teremos intervalos de $J/I = 200$ segmentos para o cálculo do tempo de reverberação; foi usado $I = 8$ na PSD; optou-se por usar um valor menor para que a quantidade de segmentos por intervalo fosse maior, e dessa forma mais dados fossem usados para o cálculo do RT_{60};

5. Tamanho do frame $M = 128$, com *overlap* de 127 amostras e janela *hamming*;

6. Parâmetro $\mu = 1 \times 10^{-8}$ na PSD, ao invés de $\mu = 3 \times 10^{-7}$ usado em [7]; esse valor foi obtido após diversas simulações, fixando-se os demais parâmetros.

As simulações foram realizadas com RIRs geradas conforme [7], denominadas de RIRs sintéticas. Dessa forma, não foi preciso estimar o valor da variância ν^2 de $b(n)$, conforme descrito na Seção 1.1. Foi usado o valor $\nu^2 = 0.1$.

Com a adequação descrita acima, os sinais reverberantes foram gerados usando-se dois sinais de voz, um feminino e outro masculino, obtidos da base *Stereo Audio Source Separation Evaluation Campaign* [35]. Os sinais foram obtidos de gravações de 10 s de duração, amostrados com $f_s = 16$ kHz. Foram usados 6 valores de RT_{60} para as RIRs sintéticas: 0.2 s, 0.33 s, 0.42 s, 0.6 s, 0.82 s e 1.0 s. Já quanto ao nível de ruído, foram usados 2 valores de SNR: 0 dB e 20 dB. Como mencionado acima, o cálculo do RT_{60} para o algoritmo PSD é realizado por intervalos. Assim, 5 resultados são obtidos, expressos na forma de erro médio absoluto (MAE). De forma a ter um resultado comparável ao obtido pela rede neural, calculou-se a média e o desvio padrão desses 5 erros.

No próximo capítulo serão apresentados os resultados de treinamento, gerados a partir da metodologia descrita acima, e os resultados comparativos dos demais métodos.

Capítulo 5

Resultados

Nessa seção serão apresentados os resultados de treinamento da rede proposta, conforme metodologia detalhada na seção 4.3, e resultados comparativos dos métodos considerados estado da arte, de acordo com [2] e [7].

5.1 Resultados da seleção da rede

Os resultados apresentados aqui, dizem respeito ao treinamento, validação e teste das redes simuladas. Pode-se visualizar os gráficos das curvas de convergência das redes, tabelas com o erro médio quadrático (MSE) de cada simulação para calibração dos parâmetros discutidos na Seção 4.3 e histogramas comparativos entre os melhores resultados obtidos com as diferentes configurações. Todos esses dados foram gerados para diferentes configurações de sala e níveis de ruído. Decidiu-se por trabalhar com 5.000 e 10.000 épocas de treinamento, visto que para valores maiores, em quase todos os casos, o resultado não apresentou melhora significativa, além deste limite servir para prevenir *overfitting*.

Nas Figuras 5.1 e 5.2, observam-se os comportamentos de convergência do algoritmo de treinamento, variando-se o valor de lr, para redes do tipo MSE e *Cross-Entropy*, respectivamente, para 10.000 épocas. Já nas Figuras 5.3 e 5.4 são apresentadas as curvas de convergência para diferentes valores de λ.

Essas são curvas de validação obtidas durante o processo de treinamento de cada rede. Nesses gráficos, podemos verificar o padrão esperado de convergência, onde quanto maior o valor de lr, mais rápido o decaimento das curvas de erro. Para $lr = 10^{-4}$ o decaimento é bem mais lento que para os demais valores. Por outro lado, para $lr = 0.1$, e $lr = 0.1$ variável (*tunning*), temos as curvas com decaimento mais abrupto, e que resultam no menor erro. Quanto ao valor de regularização λ, observou-se que o efeito no erro não é significativo nas redes do tipo *CrossEntropy*, sendo que o mesmo não é observado nas redes do tipo *MSE*. Nessas, a curva com menor erro foi obtida com $\lambda = 10^{-1}$.

Além disso, podemos concluir que, metade do número de épocas para treinamento já seria suficiente para todos os valores de lr, a menos para $lr = 10^{-4}$. Resolveu-se, então, treinar até 5.000 épocas. Caso a convergência ainda não tenha sido alcançada, continua-se até 10.000 épocas. Isso gerou uma economia de tempo de processamento.

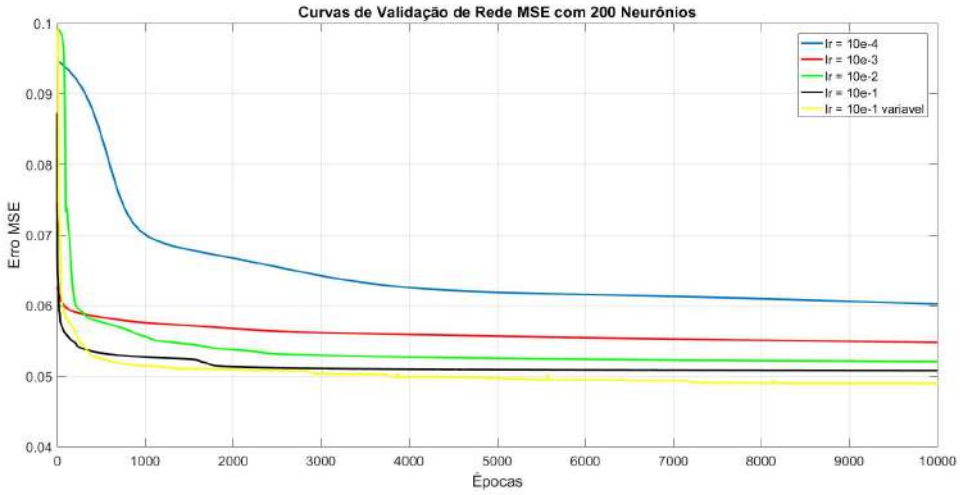

Figura 5.1: Convergência na calibração do lr das redes *MSE* de 200 neurônios

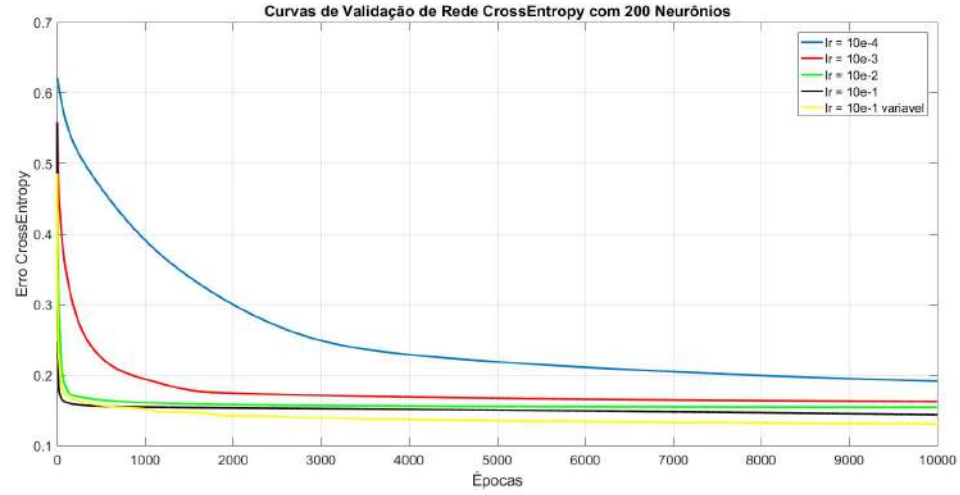

Figura 5.2: Convergência na calibração do lr das redes *CrossEntropy* de 200 neurônios

Em seguida, os resultados de *MSE* na calibração dos parâmetros propostos são apresentados na forma de tabelas. Na Tabela 5.1 temos os resultados de MSE para as redes do tipo *MSE*, variando os valores de lr e h, e com $\lambda = 0$. Os três menores valores estão destacados em vermelho. Estes foram obtidos para $lr = 0.1$ variável e $h = [50, 100, 200]$ neurônios. Como esperado, considerando os resultados anteriores das curvas de convergência, para valores menores de lr, o erro tendeu a

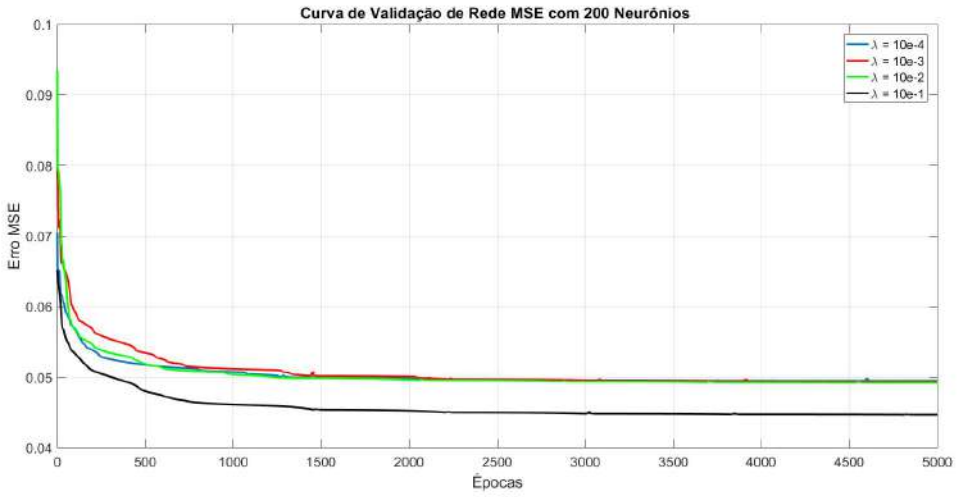

Figura 5.3: Convergência na calibração do λ das redes *MSE* de 200 neurônios

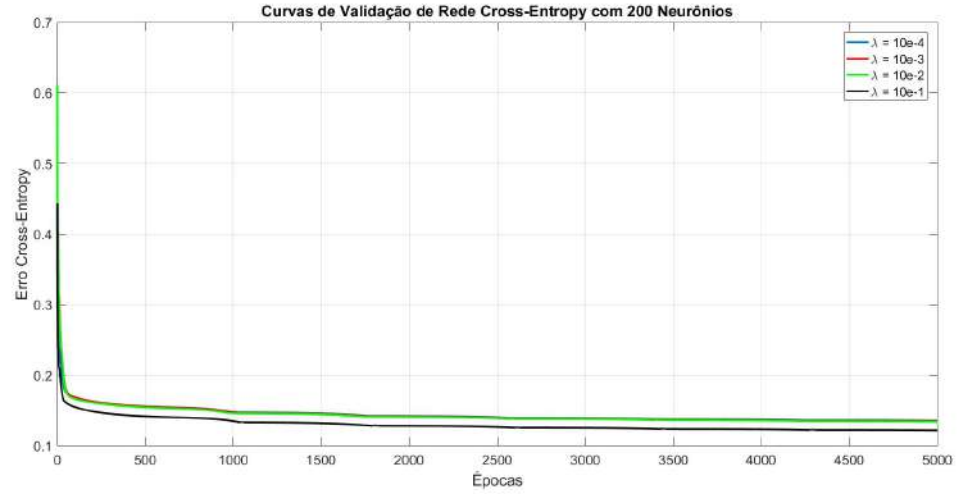

Figura 5.4: Convergência na calibração do λ das redes *CrossEntropy* de 200 neurônios

ser maior. Além disso, para as redes com maior número de neurônios, tivemos os piores resultados, o que pode ser devido ao efeito de *overfitting*.

Já na Tabela 5.2, os valores de λ são alterados para as 3 configurações selecionadas a partir da tabela anterior. Pode-se verificar que mudanças em λ não geram grandes mudanças no erro. Ainda assim, foram destacados em vermelho os melhores resultados. Essas configurações de h e λ são, respectivamente: $[50; 10^{-2}]$, $[100; 10^{-1}]$ e $[200; 10^{-1}]$.

Tabela 5.1: MSE após convergência da rede tipo *MSE* para diferentes valores de lr

# de neurônios	variável	10^{-1}	10^{-2}	10^{-3}	10^{-4}
25	0.0558	0.0821	0.1157	0.0884	0.0888
50	0.0517	0.0751	0.0998	0.0973	0.1359
100	0.0554	0.0754	0.0853	0.0816	0.0736
200	0.0530	0.0726	0.0809	0.0859	0.0887
400	0.0673	0.0802	0.0762	0.0893	0.0642

Tabela 5.2: MSE após convergência da rede tipo *MSE* para diferentes valores de λ

# de neurônios	10^{-1}	10^{-2}	10^{-3}	10^{-4}
50	0.0522	0.0488	0.0509	0.0494
100	0.0530	0.0549	0.0563	0.0613
200	0.0581	0.0605	0.0604	0.0534

De igual forma, nas Tabelas 5.3 e 5.4 são apresentados os resultados de MSE para as redes do tipo *CrossEntropy*. Mais uma vez, os resultados para valores diferentes de lr indicam um erro menor para $lr = 0.1$ variável. Esses valores foram obtidos para as redes com $h = [25, 50, 100]$ neurônios. Aqui também foi observado o mesmo comportamento, onde em geral os maiores erros foram observadas com os menores valores de lr.

Para as 3 melhores configurações de h e lr, também alterou-se o valor de λ. Os menores valores foram obtidos com os pares h e λ: $[25, 10^{-2}]$, $[50, 10^{-1}]$ e $[100, 10^{-2}]$. Como antes, não foram observadas grandes mudanças no valor do erro com alterações em λ. Em vermelho foram destacados os três menores erros.

Tabela 5.3: MSE após convergência da rede do tipo *CrossEntropy* para diferentes valores de lr

# de neurônios	variável	10^{-1}	10^{-2}	10^{-3}	10^{-4}
25	0.0427	0.0691	0.0802	0.0820	0.2139
50	0.0413	0.0729	0.0787	0.0896	0.0957
100	0.0425	0.0733	0.0757	0.0801	0.0796
200	0.0511	0.0638	0.0749	0.0861	0.0805
400	0.0544	0.0643	0.0769	0.0799	0.0822

Tabela 5.4: MSE após convergência da rede do tipo *CrossEntropy* para diferentes valores de λ

# de neurônios	10^{-1}	10^{-2}	10^{-3}	10^{-4}
25	0.0431	0.0428	0.0394	0.0427
50	0.0408	0.0428	0.0441	0.0476
100	0.0492	0.0433	0.0435	0.0438

Por fim, a melhor configuração obtida para cada tipo de rede:

1. Rede *MSE*:
   (a) $h = 50$;
   (b) $lr = 0.1$ variável;
   (c) $\lambda = 0.01$;
2. Rede *Cross-Entropy*:
   (a) $h = 25$;
   (b) $lr = 0.1$ variável;
   (c) $\lambda = 0.001$

Com isso, são geradas as Tabelas 5.5 e 5.6 com o erro médio absoluto (MAE) das duas configurações de rede anteriores. Esses valores foram obtidos para todas as diferentes configurações de sala, distância fonte-receptor e nível de ruído.

Tabela 5.5: MAE após convergência para rede proposta do tipo *MSE*

SALA	DIST	SNR					Erro Médio
		-10dB	0dB	10dB	20dB	30dB	
Pequena	Próximo	0.303 ± 0.177	0.151 ± 0.101	0.140 ± 0.093	0.122 ± 0.072	0.135 ± 0.086	0.175 ± 0.119
	Distânte	0.294 ± 0.173	0.176 ± 0.107	0.129 ± 0.083	0.125 ± 0.078	0.105 ± 0.066	0.166 ± 0.114
Média	Próximo	0.294 ± 0.175	0.179 ± 0.113	0.135 ± 0.085	0.136 ± 0.088	0.112 ± 0.078	0.164 ± 0.116
	Distante	0.293 ± 0.189	0.202 ± 0.122	0.126 ± 0.083	0.126 ± 0.083	0.124 ± 0.0811	0.174 ± 0.122
Grande	Próximo	0.305 ± 0.176	0.156 ± 0.099	0.146 ± 0.098	0.133 ± 0.086	0.133 ± 0.090	0.159 ± 0.110
	Distante	0.282 ± 0.177	0.160 ± 0.101	0.129 ± 0.082	0.111 ± 0.084	0.114 ± 0.081	0.151 ± 0.110
Média Ruido		0.295 ± 0.177	0.168 ± 0.106	0.132 ± 0.087	0.125 ± 0.082	0.117 ± 0.079	0.164 ± 0.115

Tabela 5.6: MAE após convergência para rede proposta do tipo *CrossEntropy*

SALA	DIST	SNR					Erro Médio
		-10dB	0dB	10dB	20dB	30dB	
Pequena	Próximo	0.238 ± 0.134	0.156 ± 0.092	0.132 ± 0.088	0.126 ± 0.075	0.131 ± 0.082	0.162 ± 0.106
	Distânte	0.228 ± 0.132	0.168 ± 0.099	0.123 ± 0.087	0.124 ± 0.080	0.103 ± 0.068	0.152 ± 0.103
Média	Próximo	0.243 ± 0.124	0.170 ± 0.101	0.132 ± 0.083	0.138 ± 0.089	0.107 ± 0.072	0.153 ± 0.101
	Distante	0.234 ± 0.134	0.185 ± 0.103	0.125 ± 0.082	0.117 ± 0.088	0.136 ± 0.086	0.162 ± 0.110
Grande	Próximo	0.244 ± 0.146	0.144 ± 0.092	0.143 ± 0.093	0.129 ± 0.082	0.133 ± 0.086	0.150 ± 0.100
	Distante	0.221 ± 0.125	0.140 ± 0.087	0.121 ± 0.088	0.119 ± 0.079	0.112 ± 0.071	0.140 ± 0.094
Média Ruido		0.235 ± 0.134	0.159 ± 0.097	0.128 ± 0.084	0.122 ± 0.079	0.116 ± 0.074	0.153 ± 0.108

Pelas tabelas, não se observa mudanças significativas para as diferentes configurações de tamanho de sala e distância fonte receptor. Isso pode ser justificado pelo balanceamento entre classes, considerado no treinamento das redes. Ainda assim, para a configuração de sala [Grande, Distante] os erros foram os menores obtidos, na média, para ambas as redes propostas. A rede do tipo *CrossEntropy*, como esperado, foi melhor que a rede do tipo *MSE* em todos os cenários, apresentando valor de MAE menor.

Já com relação ao ruído, como esperado, temos os piores resultados para a menor *SNR*. Além disso, na média, o erro tende a ser menor quanto maior a *SNR*, para ambas as redes. Pode-se perceber, por outro lado, que essas variações não são tão abruptas, o que se deve à robustez ao ruído dos métodos que usam informação espectral.

Com relação a amplitude dos erros apresentados, os mesmos são em média de 3 a 4 vezes o tamanho do bin (≈ 0.042) da escala de classificação. Esses erros altos podem estar associados a incertezas em alguns parâmetros usados no treinamento das redes, mas que não foram informados pelo trabalho de referência, conforme será apresentado na próxima seção.

Também foram gerados os histogramas de MAE para as duas redes, para todas as configurações possíveis, como pode ser observado nas Figuras 5.5 e 5.6, e para configurações específicas nas Figuras 5.7, 5.8 e 5.9.

Pelos histogramas, pode-se confirmar o maior número de erros da rede do tipo *MSE* em relação a rede do tipo *CrossEntropy*. Além disso, considerando os histogramas dos casos específicos, percebe-se que a maioria dos erros da rede de entropia cruzada está concentrado em valores menores, em comparação com a outra rede.

Por fim, na Figura 5.10, temos uma análise para diferentes faixas de RT_{60} dos desempenhos das redes propostas. O objetivo é verificar o comportamento das redes para valores pequenos, médios e altos de reverberação. Convencionou-se analisar os seguintes intervalos:

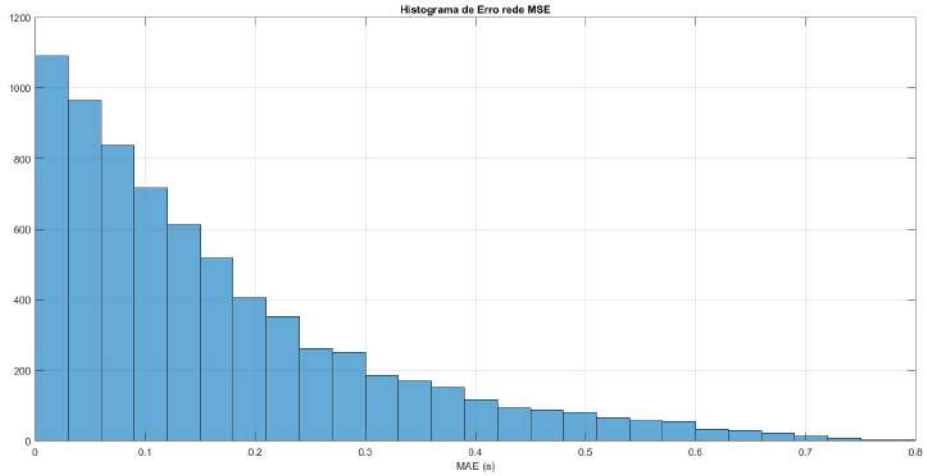

Figura 5.5: Histograma da distribuição do MAE estimado para a rede *MSE*

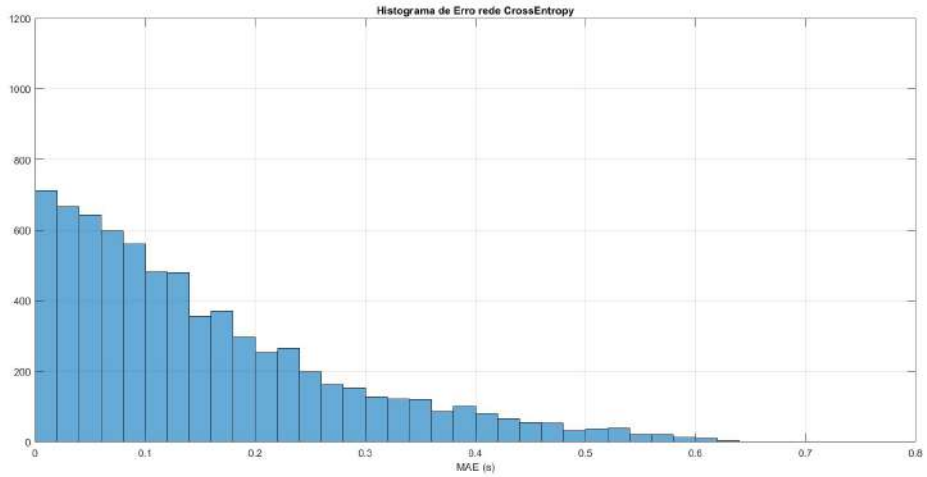

Figura 5.6: Histograma da distribuição do MAE estimado para a rede *Cross-Entropy*

1. $RT60 \leq 0.4$ s: [0.2000, 0.2444, 0.2889, 0.3333, 0.3778] s;
2. 0.4 s $< RT60 \leq 0.7$ s: [0.4222, 0.4667, 0.5111, 0.5556, 0.6000, 0.6444, 0.6889] s;
3. $RT60 > 0.7$ s: [0.7333, 0.7778, 0.8222, 0.8667, 0.9111, 0.9556, 1.0000] s;

onde para cada intervalo, temos as classes correspondentes aos respectivos RT_{60}. Pelo gráfico, verificamos que ambas as redes erram mais para valores menores de reverberação, isto é, $RT_{60} \leq 0.4$ s. Além disso, no intervalo $RT_{60} > 0.7$ s, a rede *MSE* conseguiu um resultado muito próximo do obtido pela rede *CrossEntropy*. Em vista dos resultados anteriores, optou-se pelo modelo de rede de entropia cruzada, *CrossEntropy*, como modelo proposto a ser comparado com os demais métodos de referência.

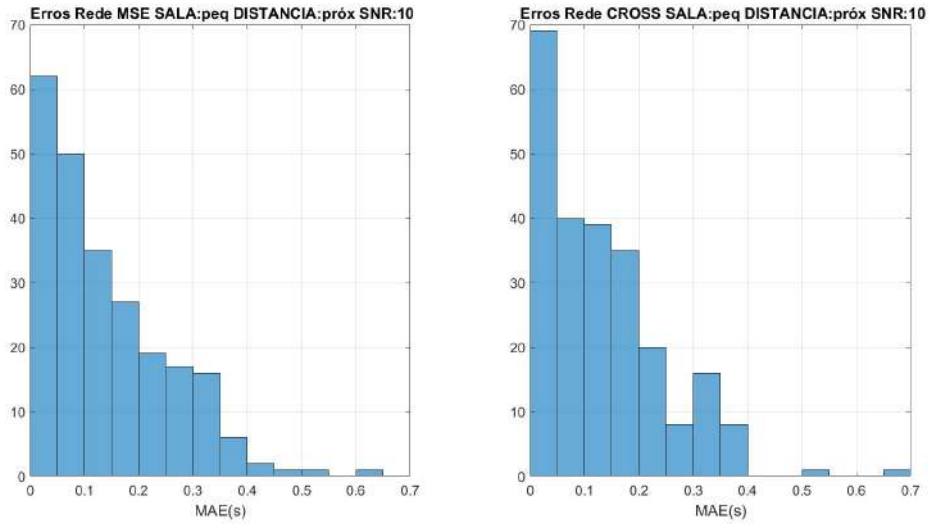

Figura 5.7: Histogramas dos MAEs de ambas as redes para a configuração pequena, próxima e 10 dB de SNR

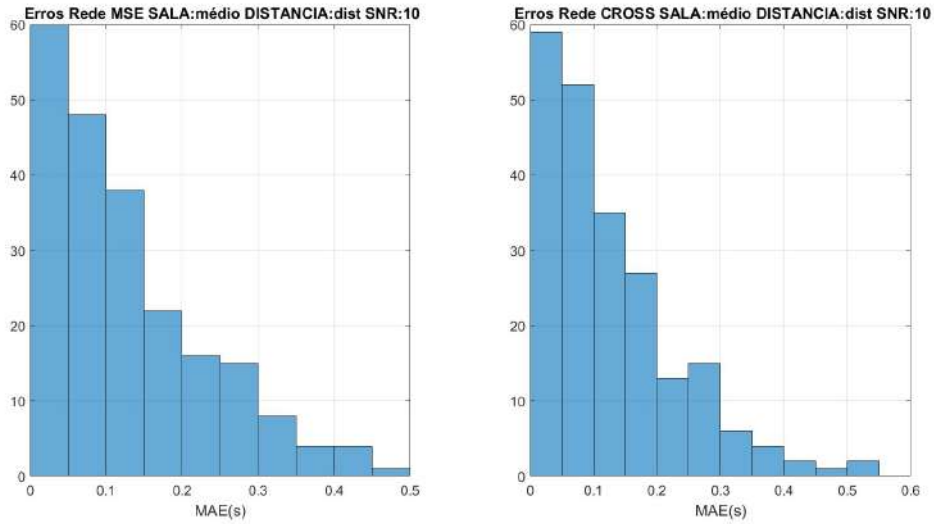

Figura 5.8: Histogramas dos MAEs de ambas as redes para a configuração: média, distante e 10 dB de SNR

Assim, na próxima seção, serão apresentados os resultados comparativos, através da métrica do erro médio absoluto (MAE), dos outros métodos. Nesses resultados, serão destacados os comportamentos para as diferentes configurações de sala e níveis de ruído.

5.2 Resultados comparativos

Nessa seção, são apresentados os resultados comparativos entre a rede proposta e o método SDD e a rede DNN descritos em [2]. Também são feitas comparações entre a rede proposta e o método PSD apresentado em [7].

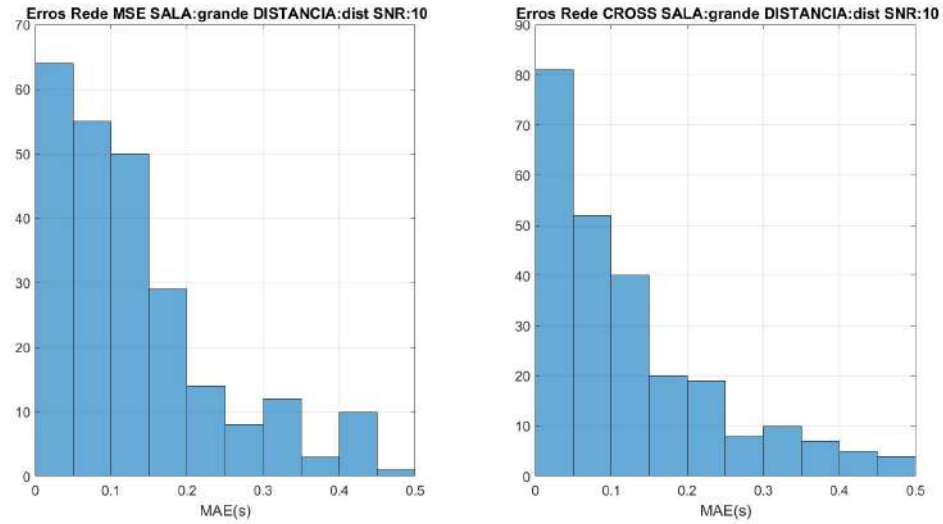

Figura 5.9: Histograma dos MAEs de ambas as redes para a configuração: grande, distante e de 10 dB de SNR

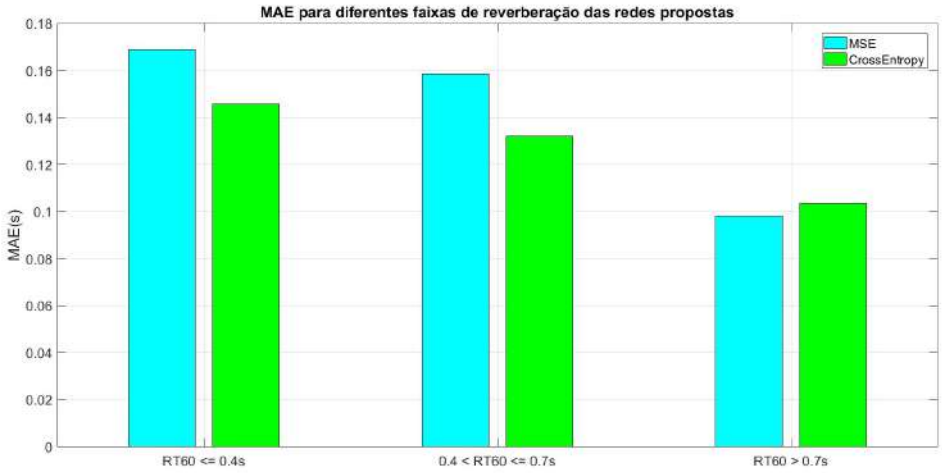

Figura 5.10: MAE para diferentes faixas de reverberação das redes propostas

O SDD (*Spectral Decay Distribution*) é uma classe de métodos que fazem um mapeamento entre a taxa de decaimento e um gradiente obtido a partir da magnitude do log da STFT de frames consecutivos do sinal reverberante.

Já a rede DNN, como mencionado anteriormente, é uma rede neural profunda de função custo *MSE*, cujas especificações foram usadas nas redes propostas. Vale lembrar que, como mencionado na Seção 4.3.1, várias informações não foram fornecidas em [2] sobre a especificação da rede. Dessa forma, os resultados obtidos pela rede proposta podem ser bem diferentes dos apresentados pela rede de referência DNN. Os valores de MAE foram comparados para as diferentes configurações de sala descritas na seção anterior. Ao final, são apresentados resultados comparativos com o método PSD, seguindo a metodologia especificada na Seção 4.3.2.

Nas Tabelas 5.7 e 5.8 encontram-se os resultados dos métodos SDD e DNN, respectivamente. Esses valores de MAE foram extraídos de [2]. Pode-se perceber que o método SDD e a rede *CrossEntropy* proposta tiveram resultados próximos, mas ligeiramente inferiores aos obtidos pela rede de referência. Isso, como observado antes, pode ser fruto da falta de informação referente a muitos dos parâmetros usados no treinamento da rede, tais como: número de neurônios nas camadas de entrada e intermediária, uso de regularização, algoritmo de atualização de pesos e *bias*, parâmetros para geração das RIRs pelo método das imagens, etc. Além disso, ao contrário do que ocorre para a rede proposta, na rede de referência pode-se observar diferenças significativas de erro para as diversas configurações de sala. Isso pode ser justificado pelo desbalanceamento da quantidade de amostras entre as configurações, permitido na metodologia usada para geração dos dados de treinamento da rede de referência. Por outro lado, considerando os resultados da rede proposta e do método SDD, verifica-se que para SNR de -10 dB, a rede obteve o menor erro médio. Considerando-se o menor erro médio de todas as configurações, a rede gerou um erro médio de 0.153 s contra 0.163 s do método SDD. Isso demostra a eficiência que pode ser alcançada com o uso de redes neurais frente a métodos estado da arte.

Tabela 5.7: MAE para diferentes configurações com o método SDD

SALA	DIST	SNR					Erro Médio
		-10dB	0dB	10dB	20dB	30dB	
Pequena	Próximo	0.358	0.164	0.104	0.095	0.088	0.162
	Distânte	0.341	0.149	0.100	0.095	0.091	0.155
Média	Próximo	0.342	0.171	0.088	0.084	0.110	0.159
	Distante	0.341	0.198	0.121	0.101	0.109	0.174
Grande	Próximo	0.259	0.123	0.102	0.119	0.148	0.150
	Distante	0.341	0.177	0.124	0.122	0.124	0.178
Média Ruido		0.330	0.164	0.107	0.103	0.112	0.163

Tabela 5.8: MAE para diferentes configurações com o método DNN

SALA	DIST	SNR					Erro Médio
		-10dB	0dB	10dB	20dB	30dB	
Pequena	Próximo	0.140	0.038	0.031	0.037	0.044	0.058
	Distânte	0.135	0.040	0.028	0.034	0.040	0.056
Média	Próximo	0.070	0.033	0.025	0.025	0.027	0.036
	Distante	0.093	0.063	0.050	0.039	0.040	0.057
Grande	Próximo	0.183	0.058	0.040	0.041	0.043	0.073
	Distante	0.224	0.124	0.096	0.091	0.086	0.124
Média Ruido		0.141	0.059	0.045	0.045	0.047	0.067

Os gráficos a seguir permitem visualizar os resultados das configurações que apresentaram maior diferença, entre os métodos SDD, DNN e a rede proposta. Na Figura 5.11, são mostrados os resultados para os diferentes tipos de sala e distâncias fonte-receptor. Podemos ver que a rede proposta *CrossEntropy* apresenta erro equivalente a da rede SDD, com erro menor na maioria dos cenários, principalmente na configuração de sala grande e distância fonte-receptor próximo. Já para os diferentes níveis de ruído, pela Figura 5.12, verifica-se que a rede proposta apresenta erros menores para SNRs mais baixas, de -10 dB e 0 dB, e erro médio equivalente ao do método SDD.

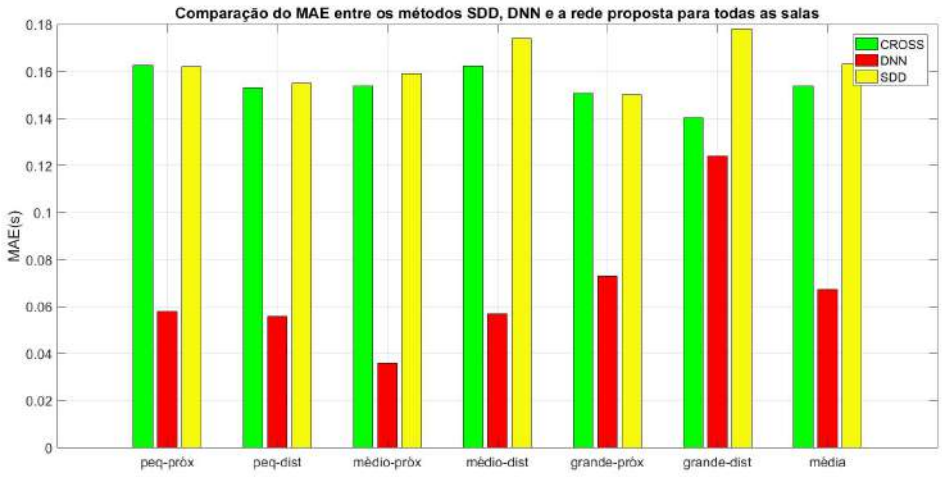

Figura 5.11: Comparação do MAE dos métodos SDD, DNN e a rede proposta para os 3 tipos de sala e distâncias fonte-receptor

Agora, considerando o algoritmo PSD para o cálculo do RT_{60}, seguindo a metodologia apresentada no capítulo anterior, foram gerados os resultados para uma voz

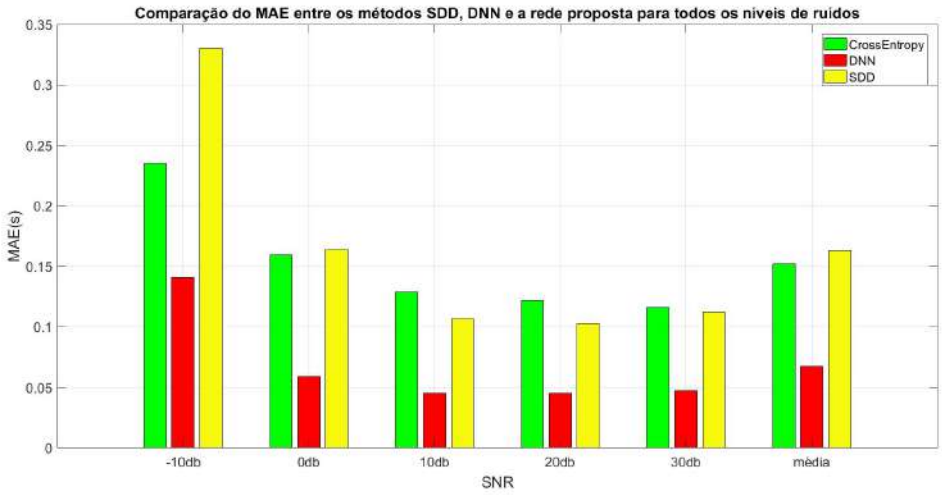

Figura 5.12: Comparação do MAE dos métodos SDD, DNN e a rede proposta para todos os valores de SNR

feminina, apresentados na Tabela 5.9, e para uma voz masculina, apresentados na Tabela 5.10.

Tabela 5.9: MAE da rede proposta *Cross-Entropy* e algoritmo PSD para voz feminina

Algoritmos	RT_{60}(s)						SNR
	0.20	0.33	0.42	0.60	0.82	1	
PSD	0.002 ± 0.002	0.005 ± 0.005	0.009 ± 0.008	0.018 ± 0.017	0.031 ± 0.029	0.045 ± 0.041	20dB
CROSS-NET	0.151 ± 0.076	0.106 ± 0.077	0.044 ± 0.050	0.184 ± 0.143	0.092 ± 0.057	0.049 ± 0.037	
PSD	0.004 ± 0.002	0.013 ± 0.005	0.020 ± 0.007	0.039 ± 0.088	0.069 ± 0.023	0.100 ± 0.035	0dB
CROSS-NET	0.378 ± 0.162	0.287 ± 0.162	0.088 ± 0.059	0.267 ± 0.088	0.190 ± 0.050	0.146 ± 0.059	

Tabela 5.10: MAE da rede proposta *Cross-Entropy* e algoritmo PSD para voz masculina

Algoritmos	RT_{60}(s)						SNR
	0.20	0.33	0.42	0.60	0.82	1	
PSD	0.004 ± 0.003	0.012 ± 0.008	0.019 ± 0.013	0.036 ± 0.025	0.063 ± 0.042	0.090 ± 0.057	20dB
CROSS-NET	0.237 ± 0.108	0.244 ± 0.162	0.324 ± 0.169	0.255 ± 0.140	0.137 ± 0.026	0.075 ± 0.068	
PSD	0.009 ± 0.003	0.025 ± 0.007	0.040 ± 0.011	0.075 ± 0.020	0.130 ± 0.031	0.182 ± 0.043	0dB
CROSS-NET	0.488 ± 0.089	0.381 ± 0.076	0.360 ± 0.033	0.288 ± 0.044	0.232 ± 0.073	0.209 ± 0.127	

Pelas Tabelas 5.9 e 5.10, é possível verificar que, como esperado, os erros de estimação dos métodos não-cegos são em geral bem menores do que os de métodos cegos. Também pode-se observar que o desempenho da PSD não é muito afetado

com a redução drástica da SNR, ao contrário do que foi verificado anteriormente para os métodos baseados em redes neurais.

Com relação às faixas de tempo de reverberação, percebe-se que o erro associado ao algoritmo PSD cresce com o aumento desse valor, não tendo sido esse comportamento observado nos demais modelos. Isso pode estar associado ao fato da PSD calcular o RT_{60} através de intervalos de tempo cujo tamanho pode não ser suficiente para o cálculo mais preciso da taxa de decaimento.

Por fim, constata-se que o erro médio associado à voz masculina é maior que o da voz feminina para ambos os métodos, e para todos os tempos de reverberação e níveis de ruído. As Figuras 5.13 e 5.14 mostram os coeficientes mel, de cada frame, para as vozes masculina e feminina, respectivamente. Observa-se uma diferença significativa entre os gráficos, o que pode justificar a diferença nos erros. Um estudo mais cuidadoso se faz necessário para avaliar o impacto dessas diferenças dos coeficientes mel dos dois tipos de voz no erro de estimação do tempo de reverberação.

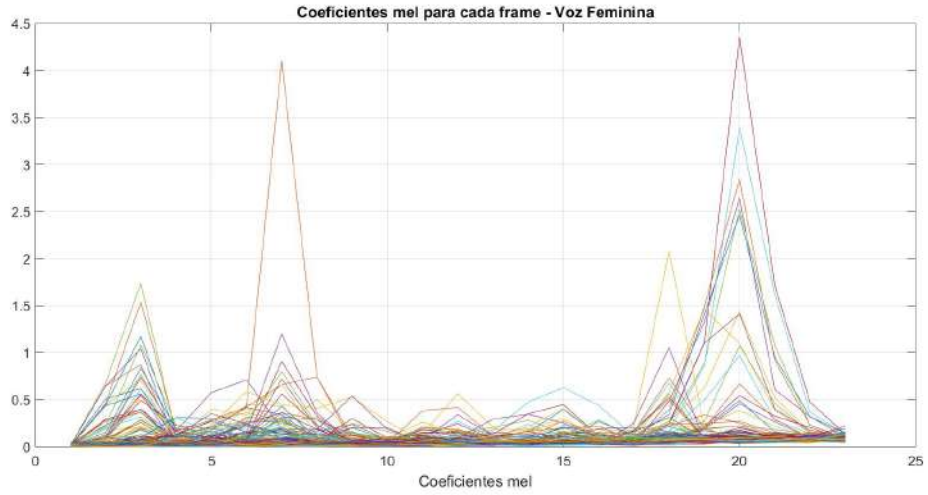

Figura 5.13: Coeficientes mel para voz feminina

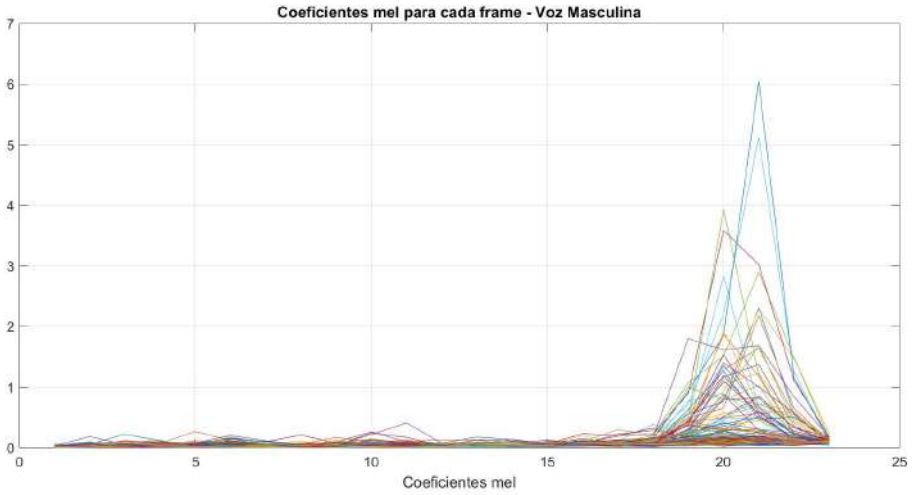

Figura 5.14: Coeficientes mel para voz masculina

Capítulo 6

Conclusão

O presente trabalho teve como objetivo investigar a possibilidade do uso de redes neurais na estimação do tempo de reverberação, através da comparação dessas redes com métodos estado da arte como *spectral decay distribution*(*SDD*), *Power Spectral Density*(*PSD*) e outra rede neural profunda (*DNN*), cujas informações foram usadas para iniciar a calibração do modelo de rede.

Foi criada uma metodologia para a calibração da rede proposta, visto que parte das informações para o treinamento da mesma não foram informadas pelo trabalho de referência usado. Além disso, pretendeu-se explorar o uso de redes de entropia cruzada, que como verificado, tendem a apresentar melhores resultados em redes de classificação. Dessa forma, diversos valores de parâmetros necessitaram ser obtidos experimentalmente, tais como: número de neurônios nas camadas de entrada e camadas escondidas, algoritmo para atualização dos pesos e *bias*, uso de regularização e pré-processamento nos dados de entrada da rede. Para a seleção desses parâmetros, levou-se em consideração o menor erro médio quadrático produzido por cada configuração.

Aplicada a metodologia, considerou-se escolher o modelo de rede neural com o menor erro médio absoluto, de forma a ser compatível com os dados de referência, em diversas configurações de tamanho de sala, distância fonte-receptor, nível de rúido e faixas de tempo de reverberação. Observou-se que, em quase todas as situações, a rede do tipo entropia cruzada (*CrossEntropy*) foi a que gerou os menores valores de erro. Dessa forma, esse modelo de rede foi selecionado para ser comparado com os métodos de referência *SDD*, *DNN* e *PSD*, sendo esse útimo um método não-cego.

Nessa comparação, ficou evidente que o erro da rede de referência é menor que os dos demais métodos cegos, com amplitude em torno de 1 a 2 vezes o valor do bin da escala de classificação. Já os demais métodos tiveram erros de 3 a 4 vezes o valor do bin. Por outro lado, também pode-se verificar que a rede proposta apresentou resultados semelhantes aos do método *SDD*, sendo superior na média para as diferentes configurações de sala e nível de ruído.

Com relação ao método *PSD*, seu resultado foi comparado ao da rede proposta de entropia cruzada, de forma a se verificar a diferença dos erros de estimação do RT_{60} entre um método cego e um não-cego. Para isso, também foi criada uma metodologia de calibração da *PSD*, além do cuidado para que os dados fossem pré-processados de igual forma para ambos os métodos. Pelos resultados, pode-se confirmar a superioridade do método *PSD*, com valores de erro inferiores a todos os resultados dos demais métodos, além do seu comportamento onde quanto maior o valor do RT_{60}, maior o erro de estimação. Esse comportamento não foi observado na rede proposta, o que pode estar relacionado ao tamanho dos intervalos dentro dos quais o sinal é dividido e onde o tempo de reverberação é calculado. Também observou-se que, ao se comparar os resultados obtidos para vozes femininas e masculinas, o erro médio foi maior para as masculinas em ambos os métodos. Isso pode estar relacionado às diferenças entre os coeficientes mel encontrados para os dois tipos de vozes, sendo que um estudo mais cuidadoso para avaliação e melhoria do método de estimação do RT_{60} será necessário.

Pode-se concluir finalmente que os experimentos realizados comprovaram a capacidade de estimação das redes neurais para o cálculo do RT_{60}, considerando uma faixa de 0.2s até 1s de reverberação, uso dos coeficientes mel e para diferentes condições de sala e razão sinal-ruído.

Como propostas futuras, quanto á rede neural treinada, pretende-se ampliar a metodologia usada de forma a explorar novas topologias de rede, como as redes recorrentes e convolucionais. Além disso, outros dados da RIR, além do RT_{60}, podem ser estimados de forma a se caracterizar a qualidade sonora de um espaço fechado. Dentre os parâmetros que se pretende estimar, temos o *Early Decay Time*(EDT), o Speech Clarity Index(SCI) e o *Direct-To-Reverberant Energy Ratio*(DRR) [36]. O EDT é o intervalo de tempo que leva para o sinal decair 10 dB, o SCI é a razão entre a energia dos 50 ms iniciais de reflexão e as demais reflexões. Por fim, o DRR é a razão entre a energia do som direto pela energia da reverberação.

Referências Bibliográficas

[1] BERANEK, L. "W. C. Sabine's personal papers". v. 5, pp. 015001–015001, 01 2009. doi: 10.1121/1.3072523.

[2] XIAO, X., ZHAO, S., ZHONG, X., et al. "Learning to estimate reverberation time in noisy and reverberant rooms". In: *INTERSPEECH*, 2015.

[3] R. SCHROEDER, M. "New Method of Measuring Reverberation Time", v. 37, pp. 409–412, 03 1965.

[4] R. RATNAM, D. L. JONES, B. C. W. W. D. O. J. C. R. L., FENG, A. S. "Blind estimation of reverberation Time", v. 114, pp. 2877–2892, 08 2003.

[5] HEINRICH W. LÖLLMANN, YILMAZ E., M. J. P. V. "Algorithm for Blind Reverberation Time Estimation". In: *Proceedings of International Workshop on Acoustic Echo and Noise Control (IWAENC)*, pp. 1–4, 2010.

[6] WEN, J. Y. C., HABETS, E. A. P., NAYLOR, P. A. "Blind estimation of reverberation time based on the distribution of signal decay rates". In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 329–332, March 2008. doi: 10.1109/ICASSP.2008.4517613.

[7] FARAJI, N., AHADI, S. M., SHEIKHZADEH, H. "Reverberation time estimation based on a model for the power spectral density of reverberant speech". In: *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 1453–1457, Aug 2016. doi: 10.1109/EUSIPCO.2016.7760489.

[8] LEE, M., CHANG, J. "Blind estimation of reverberation time using deep neural network". In: *2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pp. 308–311, Sept 2016. doi: 10.1109/ICNIDC.2016.7974586.

[9] "Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between". `https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html`. Accessed: 2018-09-23.

[10] ALLEN, J. B., BERKLEY, D. A. "Image method for efficiently simulating small-room acoustics", *The Journal of the Acoustical Society of America*, v. 65, n. 4, pp. 943–950, 1979. doi: 10.1121/1.382599. Disponível em: <https://doi.org/10.1121/1.382599>.

[11] HAYKIN, S. "Neural Networks and Learning Machines". In: *Neural Networks and Learning Machines*, Pearson Prentice Hall, cap. 7, 2009.

[12] . *Found. Trends Mach. Learn.*, v. 2, n. 1, 2009. ISSN: 1935-8237.

[13] HAYKIN, S. "Neural Networks and Learning Machines". In: *Neural Networks and Learning Machines*, Pearson Prentice Hall, cap. 1, 2009.

[14] HOCHREITER, S., SCHMIDHUBER, J. "Long Short-term Memory". 1997.

[15] "Convolutional Neural Networks (LeNet)". http://deeplearning.net/tutorial/lenet.html. Accessed: 2018-11-25.

[16] DE PÁDUA BRAGA, A. *Redes neurais artificiais: teoria e aplicações*. LTC Editora, 2007. ISBN: 9788521615644. Disponível em: <https://books.google.com.br/books?id=R-p1GwAACAAJ>.

[17] "Cross Entropy Cost Function". http://deeplearningbook.com.br/cross-entropy-cost-function/, . Accessed: 2018-12-09.

[18] "Why You Should Use Cross-Entropy Error Instead Of Classification Error Or Mean Squared Error For Neural Network Classifier Training". https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or- . Accessed: 2018-12-11.

[19] "PERFORMANCE EVALUATION OF SPEAKER IDENTIFICATION SYSTEM USING MEL SCALE AND BARK SCALE". http://shodhganga.inflibnet.ac.in/bitstream/10603/153406/12/12_chapter3.pdf.pdf. Accessed: 2019-31-03.

[20] "The Bark Frequency Scale". https://ccrma.stanford.edu/~jos/bbt/Bark_Frequency_Scale.html. Accessed: 2019-31-03.

[21] "Auditory Scale Analysis and Evaluation of Phonemes in MISING Language". https://pdfs.semanticscholar.org/04ab/b6af6e7926b94fbf28b0e29f7567dfaddea6.pdf. Accessed: 2019-31-03.

[22] "RIR-Generator". https://github.com/ehabets/RIR-Generator. Accessed: 2018-10-03.

[23] "Supported and Compatible Compilers – Release 2016b". https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/support/sysreq/files/SystemRequirements-Release2016b_SupportedCompilers.pdf. Accessed: 2018-10-03.

[24] "Choose a C++ Compiler". https://www.mathworks.com/help/matlab/matlab_external/choose-c-or-c-compilers.html. Accessed: 2018-10-03.

[25] "WSJCAM0 Cambridge Read News". https://catalog.ldc.upenn.edu/LDC95S24. Accessed: 2018-10-03.

[26] "CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit". https://datashare.is.ed.ac.uk/handle/10283/2651. Accessed: 2018-10-03.

[27] "TIMIT Acoustic-Phonetic Continuous Speech Corpus". https://github.com/philipperemy/timit. Accessed: 2018-10-04.

[28] "MOCHA-TIMIT". http://www.cstr.ed.ac.uk/research/projects/artic/mocha.html. Accessed: 2018-10-04.

[29] HABETS, E. *Room Impulse Response Generator*. Relatório técnico, 01 2006.

[30] "Divide Data for Optimal Neural Network Training". https://www.mathworks.com/help/deeplearning/ug/divide-data-for-optimal-neural-network-training.html, . Accessed: 2019-04-09.

[31] YU, C.-C., LIU, B.-D. "A backpropagation algorithm with adaptive learning rate and momentum coefficient", , n. 2, pp. 1218–1223, 2002. doi: 10.1109/IJCNN.2002.1007668.

[32] "Implementing Neural Network L2 Regularization". https://jamesmccaffrey.wordpress.com/2017/06/29/implementing-neural-network-l2-regularization/. Accessed: 2018-10-18.

[33] "How Much Training Data is Required for Machine Learning?" https://machinelearningmastery.com/much-training-data-required-machine-learning/, . Accessed: 2019-04-04.

[34] "Gradient descent with momentum and adaptive learning rate back-propagation". `https://www.mathworks.com/help/deeplearning/ref/traingdx.html`. Accessed: 2018-10-19.

[35] "Stereo Audio Source Separation Evaluation Campaign". `http://www.irisa.fr/metiss/SASSEC07`. Accessed: 2018-10-03.

[36] FALK, T. H., CHAN, W.-Y. "Temporal Dynamics for Blind Measurement of Room Acoustical Parameters", *IEEE Transactions on Instrumentation and Measurement*, v. 59, pp. 978–989, 2010.