UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

EDRESSON CASANOVA

SÍNTESE DE VOZ APLICADA AO PORTUGUÊS BRASILEIRO USANDO APRENDIZADO PROFUNDO

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2019

EDRESSON CASANOVA

SÍNTESE DE VOZ APLICADA AO PORTUGUÊS BRASILEIRO USANDO APRENDIZADO PROFUNDO

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Computação".

Orientador: Prof. Dr. Arnaldo Candido Junior

MEDIANEIRA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso de Ciência da Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

SÍNTESE DE VOZ APLICADA AO PORTUGUÊS BRASILEIRO USANDO APRENDIZADO PROFUNDO

Por

EDRESSON CASANOVA

Este Trabalho de Conclusão de Curso foi apresentado às 09:00h do dia 01 de julho de 2019 como requisito parcial para a obtenção do título de Bacharel no Curso de Ciência da Computação, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Arnaldo Candido Junior
UTFPR - Câmpus Medianeira

Prof. Dr. Pedro Luiz de Paula Filho
UTFPR - Câmpus Medianeira

Prof. Me. Jorge Aikes Junior
UTFPR - Câmpus Medianeira

A folha de aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

CASANOVA, E.. SÍNTESE DE VOZ APLICADA AO PORTUGUÊS BRASILEIRO USANDO APRENDIZADO PROFUNDO. 83 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

Redes Neurais Artificiais Profundas tem sido utilizadas para solucionar uma ampla gama de problemas. Em particular, tal metodologia permitiu aumentar substancialmente o estado da arte na área de síntese de voz. Neste trabalho explorou-se o estado da arte da síntese de voz para o Português Brasileiro, para tal foi necessário a criação de uma base de áudio contendo aproximadamente 10 horas de um único locutor no idioma. Redes neurais profundas são formadas por um número de nós, ou unidades, conectados por ligações, estes nós representam neurônios artificiais e são organizados em camadas conectadas por conjuntos de pesos. Uma série de modelos para síntese de voz foram investigados, a exemplo o DCTTS, o Tacotron e o TTS da Mozilla. Alguns experimentos foram propostos visando explorar os principais modelos de síntese de voz e *vocoders* da literatura. Os resultados demonstraram que o modelo TTS da Mozilla soa mais natural e possui um melhor desempenho que os demais modelos explorados, entretanto, a qualidade dos áudios sintetizados pelo modelo DCTTS fica muito próxima. Adicionalmente, explorou-se o uso de transferência de aprendizado do idioma Inglês para o Português, o que demonstrou vantagens na aplicação de tal técnica.

Palavras-chave: Inteligência Artificial, Rede Neural Artificial, Fala de Computador

ABSTRACT

CASANOVA, E.. SPEECH SYNTHESIS APPLIED TO BRAZILIAN PORTUGUESE USING DEEP LEARNING. 83 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

Deep Artificial Neural Networks have been used to solve a wide range of problems. In particular, such methodology allowed to substantially increase the state of the art in the area of speech synthesis. In this work we explored the state of the art of speech synthesis for Brazilian Portuguese, for this it was necessary to create an audio base containing approximately 10 hours of a single speaker in the language. Deep neural networks are formed by a number of nodes, or units, connected by links, these nodes represent artificial neurons and are arranged in layers connected by sets of weights. A number of models for voice synthesis were investigated, such as DCTTS, Tacotron and Mozilla TTS. Some experiments were proposed to explore the main models of speech synthesis and vocoders in the literature. The results showed that the Mozilla TTS model sounds more natural and performs better than the other explored models, however, the audio quality synthesized by the DCTTS model is very close. In addition, the use of transfer learning from the English to Portuguese was explored, which demonstrate advantages in the application of such technique.

Keywords: Artificial Intelligence, Artificial Neural Network, Computer Talk, Voice Synthesis, Text-to-Speech

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CNNs	*Convolutional Neural Networks*
CReLU	Concatenated Rectified Linear Units
DCTTS	*Deep Convolutional Text To Speech*
DTW	*Dynamic Time Warping*
GRU	Gated Recurrent Unit
IPA	International Phonetic Alphabet
MFCCs	Mel-Frequency Cepstral Coefficients
MLP	Multilayer Perceptron
MOS	Mean Opinion Score
ReLU	Rectifier Linear Unit
RMSE	*Root Mean Square Error*
RNA	Rede Neural Artificial
RNRs	Redes Neurais Recorrentes
SSIM	Structural Similarity Index Method
TTS	Text-to-Speech

SUMÁRIO

1 INTRODUÇÃO

Uma das aplicações dos sistemas de Síntese de Voz (TTS – *Text-to-Speech*), em conjunto com o Reconhecimento Automático de Voz (ASR – *Automatic Speaker Recognition*) é transformar a forma como o ser humano interage com a máquina, fazendo essa interação ser o mais próximo de uma conversa. Atualmente, modelos de síntese de voz estão muito presentes no dia a dia, como as aplicações Siri (GRUBER, 2009), Cortana (CHRIS, 2014) e Alexa (PURINGTON et al., 2017) cujas as interações são uma forma de facilitar as tarefas diárias. Entretanto, segundo Tachibana et al. (2017), os sistemas tradicionais de síntese de voz não são simples de serem desenvolvidos, já que esses sistemas são tipicamente compostos de muitos módulos específicos. Um sistema de síntese de voz tradicional é uma integração elaborada de muitos módulos, como um analisador de texto, um gerador de espectro, um estimador de pausa e um codificador de voz (*vocoder*) que sintetizam uma forma de onda a partir dos dados de entrada.

O Aprendizado Profundo ou Aprendizagem Profunda (*Deep Learning*) (GOODFELLOW et al., 2016b) tem o potencial de unir todos esses processos em um único modelo e conectá-lo diretamente à entrada e à saída. Esse tipo de técnica pode ser chamada de ponta-a-ponta (*end-to-end learning*). Embora tal modelo seja por vezes criticado como de difícil interpretação, os sistemas de síntese de voz treinados de ponta a ponta como Tacotron (WANG et al., 2017b), DCTTS (TACHIBANA et al., 2017) e DeepVoice3 (PING et al., 2017), que estimam diretamente um espectrograma a partir de uma entrada de texto, alcançaram desempenhos promissores. Os modelos DeepVoice3 e DCTTS são baseados em Redes Neurais Convolucionais (CNNs – Convolutional Neural Networks), visando diminuir o custo computacional dos modelos de síntese de voz. As Redes Neurais Convolucionais provaram ser muito eficazes como extratores de característica na área de visão computacional (RAZAVIAN et al., 2014). CNNs também foram utilizadas para outras tarefas como reconhecimento de objetos (HE et al., 2016) e identificação de faces (FARFADE et al., 2015).

É difícil encontrar trabalhos na literatura que utilizam o estado da arte para síntese de voz aplicado ao Português, já que essa língua possui poucos recursos computacionais

disponíveis. A maior parte dos modelos são treinados utilizando base de áudio em Inglês. Existem poucas bases de dados de áudio abertas e gratuitas em Português e as existentes são de pequeno porte (MACLEAN, 2018; NETO et al., 2008), de modo que não foram encontradas neste trabalho bases para síntese de voz contendo uma quantidade de horas suficiente para conseguir um bom desempenho no treinamento. Geralmente bases para síntese de voz possuem mais de 7 horas de áudio. Portanto, este trabalho propõe a implementação de um modelo baseado em Rede Neural Convolucional para síntese de voz baseado no modelo de Tachibana et al. (2017), bem como a criação de uma base de dados de fala com apenas um locutor com aproximadamente 10 horas de discurso para o treinamento do modelo em Português, além de uma análise de desempenho sobre o aprendizado de máquina.

1.1 OBJETIVO GERAL E ESPECÍFICOS

O objetivo geral deste trabalho é comparar o desempenho de modelos de destaque na literatura para síntese de voz em Português Brasileiro. Esse objetivo geral pode ser dividido nos seguintes objetivos específicos:

- Criação de base de texto contendo frases, utilizando textos de domínio público, inicialmente da Wikipédia [1];
- Gravação da base de dados com aproximadamente 10 horas de áudio de um único locutor;
- Implementação de uma Rede Neural Convolucional para síntese de voz baseada no modelo DCTTS;
- Treinamento dos modelos de destaque da literatura para síntese de voz;
- Comparação entre os modelos treinados utilizando a base proposta no trabalho.

1.2 JUSTIFICATIVA

A fala é um meio essencial de comunicação entre pessoas. Por isso, reproduzir esse processo em sistemas computacionais que permitem a comunicação da máquina com o humano

[1] Wikipédia: https://pt.wikipedia.org

é de grande importância para novas tecnologias. Estudos na área de inteligência artificial vem se tornando fundamentais para construção de algoritmos que tentam agir o mais próximo do ser humano, sendo o Deep Learning (ou Aprendizagem Profunda) atualmente uma das abordagens mais proeminentes com esse fim.

A maioria das abordagens para processamento de voz focam-se na língua Inglesa como ponto de partida. Adicionalmente, nos países falantes de português, o estudo do Deep Learning com foco a síntese de voz é pouco explorado. Como consequência, há falta de bases de dados para treinar modelos de síntese de voz em português.

Portanto, este trabalho explora a síntese de voz para o português utilizando os conceitos de Deep Learning, trazendo uma base de dados e um trabalho livremente acessível para futuras utilizações e melhorias.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Esse documento será organizado da seguinte forma. O Capítulo 2 apresenta o tema fonética e fonologia. Seguindo o capítulo, uma descrição sobre redes neurais artificiais, bem como uma introdução ao aprendizado profundo. O Capítulo ainda apresenta uma breve introdução ao pré-processamento de fala utilizando espectrogramas e MFCCs. Ao final do capítulo é apresentada uma descrição sobre síntese de voz, bem como trabalhos relacionados. Os materiais e métodos que foram utilizados na produção deste trabalho são apresentados no Capítulo 3. No Capítulo 4 são apresentados os resultados obtidos a partir dos materiais e métodos propostos e também a discussão deles. Por fim, é realizada uma conclusão de todo o trabalho aqui descrito no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados a estrutura básica e conceitos sobre Redes Neurais Artificiais e sintetizadores de voz. A seguir, foca-se no estudo do Aprendizado Profundo e como ele é fundamental na aprendizagem de máquina. Este capítulo também apresenta estudos de síntese de voz e sua fundamentação.

2.1 FONÉTICA E FONOLOGIA

A fonologia é a área linguística que descreve a maneira com a qual os sons são realizados de forma distinta em contextos diferentes (JURAFSKY; MARTIN, 2014). Por sua vez, o objetivo de estudo da fonética é a apresentação dos métodos para descrever, classificar e transcrever os sons da fala, explicando como ocorre sua produção (SILVA, 1999).

A prosódia é o estudo da pronúncia das palavras e faz parte do campo da fonética. A pronúncia de uma palavra pode ser representada como uma cadeia de símbolos que são chamados de fonemas. Um fonema é um som da fala, que podem ser representados como símbolos fonéticos que se assemelham a uma letra em um alfabeto (JURAFSKY; MARTIN, 2014). Existem convenções bem definidas para representação de fonemas, sendo o IPA (do inglês, *International Phonetic Alphabet*) (DECKER et al., 1999) a mais difundida.

Os sons produzidos pelo aparelho fonador do falante são gerados por um conjunto de órgãos que podem serem subdivididos em três principais grupos: os órgãos respiratórios, que são responsáveis por produzir as correntes de ar (pulmões, brônquios e traqueia); após, laringe e cordas vocais são os órgãos responsáveis pela produção das vibrações utilizadas na fala; e por fim, faringe, boca e fossas nasais são responsáveis pelos diversos sons da fala. A produção do som forma diferentes fluxos de ar, possibilitando a classificação do som em vogais e consoantes. Nas consoantes acontece uma obstrução no fluxo de ar, e o som passa livremente pelo aparelho fonador do falante na produção das vogais (SERRANI, 2015).

Uma frase foneticamente balanceada é uma sentença que contém unidades fonéticas de acordo com sua frequência de ocorrência em um determinado idioma (GIBBON et al., 2012). Uma unidade fonética pode ser um fone, um bi-fone, um tri-fone ou uma silaba (KOMINEK; BLACK, 2004). Os corpus que contem frases foneticamente balanceadas são chamados de "corpus foneticamente balanceados" (MURTOZA et al., 2011).

2.2 REDES NEURAIS ARTIFICIAIS

De acordo com Russell e Norvig (2016), a primeira Rede Neural Artificial (RNA), foi criada por Rosenblatt (1958) e foi baseada no algoritmo Perceptron. Surgiram expectativas na comunidade científica do modelo ser o pilar para estruturar um sistema de inteligência artificial forte. A área de RNAs, contudo, entrou em crise em 1960 devido ao artigo de Minsky e Selfridge (1960) com previsões pessimistas, especificamente, na capacidade do Perceptron em resolver o problema do ou-exclusivo da lógica proposicional. O interesse na área ressurgiu em 1989 com um algoritmo de aprendizado de máquina adequado, conhecido como Backpropagation e descrito na Seção 2.4. O uso de RNAs vem crescendo muito com o aprendizado profundo desde 2006 (GOODFELLOW et al., 2016b). O interesse na pesquisa em Rede Neural Artificiais oscilou muito, mas é inegável que ganhou espaço recentemente como um algoritmo de última geração (*state-of-the-art*) para muitas aplicações (QUINTANILHA, 2017).

As redes neurais artificiais foram desenvolvidas inspiradas no funcionamento do cérebro humano. Portanto, o Neurônio Artificial é inspirado no funcionamento do Neurônio Biológico. Da mesma forma que o cérebro aprende a partir de experiências, as RNAs também tendem a melhorar conforme são treinadas para uma dada tarefa (HAYKIN et al., 2009).

2.2.1 Neurônio Biológico

O cérebro humano é formado por neurônios, e é com o trabalho conjunto entre esses neurônios que a informação é processada. O neurônio é uma célula com a parte central (núcleo)

e o corpo (soma) onde reações químicas e elétricas simbolizam o processamento de informação. A Figura 1 representa um modelo simplificado de um único neurônio real.

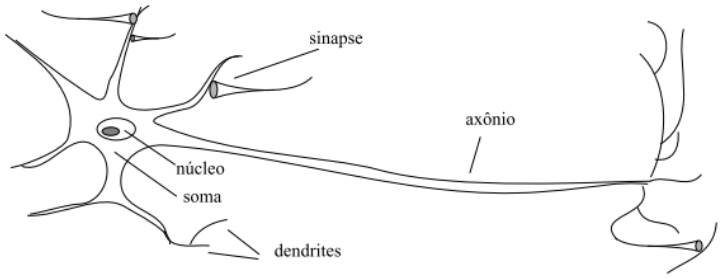

Figura 1 – Neurônio biológico
Fonte: (RAUBER, 2005)

Dendritos são prolongamentos que se estendem através do corpo celular e recebem sinais de outros neurônios ou de células sensoras. Quando um neurônio é suficientemente excitado, ele dispara. A saída da informação do soma é realizada por impulsos elétricos que se propagam através do axônio. No fim do axônio existem diversas ramificações que distribuem a informação para outros neurônios vizinhos. A conexão com outros neurônios é realizada através de sinapses que são conexões ou vãos entre o axônio de um neurônio e dendrito do outro neurônio. A sinapse é o ponto de contato entre dois neurônios, e é composta por um lado pré-sináptico que é onde o sinal está sendo enviado, e um lado pós-sináptico que é onde o sinal está sendo recebido. A sinapse dispara uma substância química quando é excitada através do impulso do axônio. De acordo com as excitações que as células vizinhas transmitem para a célula em questão, ela processa a informação e a retransmite através de seu axônio (RAUBER, 2005).

2.2.2 Neurônio Artificial

Uma RNA é constituída por um número de nós, ou unidades, conectados por ligações, estes nós representam neurônios artificiais. Cada uma destas ligações possuem um peso numérico associado a ela. O aprendizado geralmente ocorre atualizando esses pesos. Os nós são organizados em camadas: a camada de entrada das informações; camadas intermediárias ou ocultas; e camada de saída. As conexões entre neurônios de diferentes camadas simulam uma sinapse de um neurônio biológico.

Durante o treinamento, os pesos são modificados de forma a associar entradas às suas respectivas saídas (RUSSELL; NORVIG, 2016). Cada neurônio possui três elementos, sendo eles um conjunto de sinapses, um agente somador e uma função de ativação, conforme observado na Figura 2.

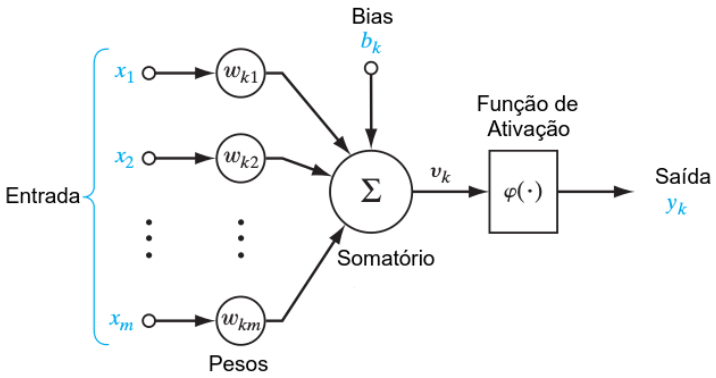

Figura 2 – Modelo de um neurônio artificial

Fonte: Adaptado de Haykin et al. (2009)

O conjunto de sinapses é representado pelas n entradas de x_i, onde cada entrada é ponderada por um dado peso sináptico w_{ki}. O agente somador é responsável por acumular todas as sinapses atualizadas $(x_i * w_{ki})$ em u_k. Por fim, a função de ativação (φ) é responsável pela propagação da amplitude do sinal. O limiar de ativação (*bias*), tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, e é denotado por b_k (HAYKIN et al., 2009).

O neurônio k mostrado na Figura 2, pode ser representado matematicamente utilizando as equações 1 e 2. A Equação 1 apresenta o agente somador que acumula no vetor u as multiplicações entre as sinapses e seus respectivos pesos.

$$u_k = \sum_{i=1}^{n} w_i x_i \tag{1}$$

Na Equação 2 é apresentado o sinal y_k que é o valor de saída da Rede Neural.

$$y_k = \varphi(u_k + bk) \tag{2}$$

2.2.3 Rede Multilayer Perceptron Feedforward

Redes Multilayer Perceptron (MLP) são uma extensão do modelo Perceptron no qual vários neurônios são organizados em camadas. O tipo mais simples de rede MLP são as redes *feedforward*.

Em redes *feedforward*, a informação flui apenas da camada de entrada para a camada de saída. Dessa forma, não há nenhuma informação da camada de saída sendo retro-alimentada para a entrada. Atuam como aproximadores universais de função (HORNIK, 1991). Neurônios de uma dada camada se comunicam com os neurônios da camada anterior através de pesos sinápticos (HAYKIN et al., 2009).

Em particular, no caso de redes completamente conectadas, todos os neurônios entre duas camadas vizinhas estão conectadas por sinapses, mas os neurônios da mesma camada não são interligados entre si, conforme ilustrado na Figura 3. A Rede Neural *feedforward* tem uma entrada, conectada a todos os neurônios na primeira camada oculta, e tem uma saída, conectada a cada neurônio da penúltima camada, as camadas internas são totalmente conectadas entre suas adjacentes. Geralmente, a entrada é chamada de camada de entrada, a saída é chamada de camada de saída e as camadas entre elas são conhecidas como camadas ocultas (RUSSELL; NORVIG, 2016).

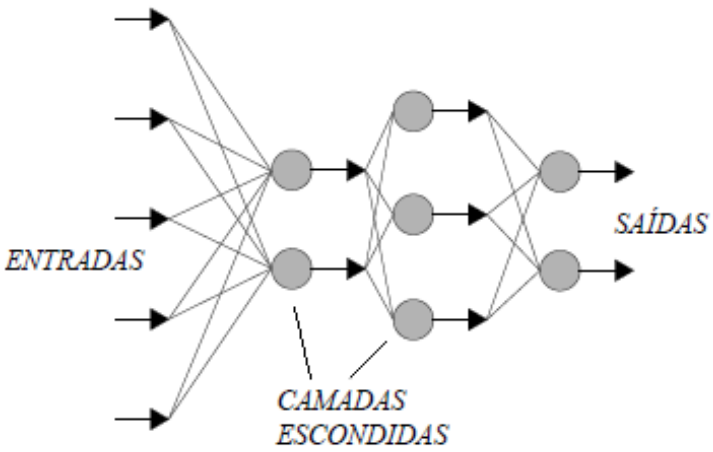

Figura 3 – Rede Multilayer Perceptron Feed-Forward completamente conectada
Fonte: Adaptado de Rauber (2005)

Se duas camadas ocultas vizinhas na rede *feedforward* tiverem ativações lineares, a composição entre as funções de ativação pode ser reduzida a uma única função linear. Assim, uma Rede Neural *feedforward* feita de apenas neurônios lineares em camadas arbitrárias

pode sempre ser descrita por uma Rede Neural Artificial com uma única camada linear (QUINTANILHA, 2017). Isso não é desejável, pois desta forma, a rede só poderá distinguir classes que são linearmente separáveis, o que limita o poder de aprender separações mais complexas. Por esta razão, é desejável que a ativação seja alterada para uma função não linear (HAYKIN et al., 2009).

2.3 FUNÇÕES DE ATIVAÇÃO

Como discutido anteriormente na Seção 2.2, a função de ativação é muito importante para a Rede Neural, pois é ela que definirá a propagação da saída do neurônio. Toda ativação recebe um único valor e executa uma operação matemática fixa nele. Existem várias funções geralmente utilizadas.

2.3.1 Sigmoide Logística

A função Sigmoide Logística é contínua, e portanto diferenciável em todos os seus pontos (HAYKIN et al., 2009). Matematicamente essa função de ativação pode ser definida pela Equação 3 e graficamente pode ser representada como na Figura 4. A entrada x é um número real e a função o converte para um intervalo entre 0 e 1 (GOODFELLOW et al., 2016b). A função já foi amplamente utilizada em modelos neurais, entretanto ela possui duas desvantagens. A primeira é a saturação que é causada quando a função se aproxima de 0 ou de 1, próximo a esses valores sua derivada é quase zero, isso afeta os cálculos do gradiente requeridos para o treinamento da rede (QUINTANILHA, 2017). A segunda é que a Sigmoide Logística nunca tem 0 como saída. Esse tipo de saída, quando ocorre permite simplificar os cálculos na camada seguinte e acelerar o treinamento da rede (LECUN et al., 1991).

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3}$$

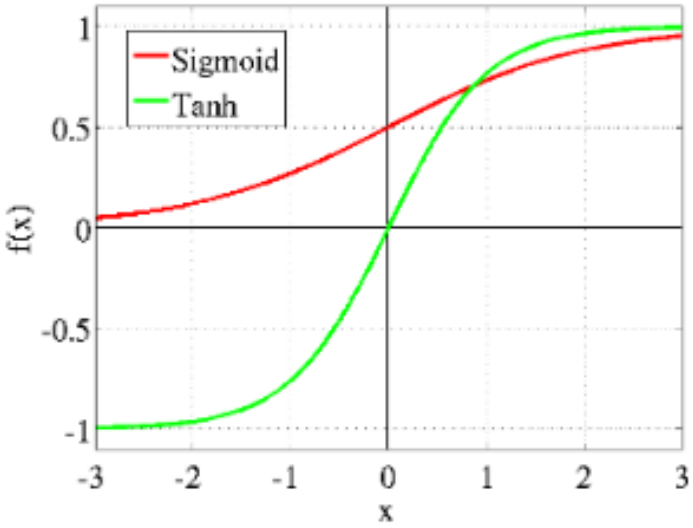

Figura 4 – Funções de ativação Sigmoid e Tanh

Fonte: (GLOROT et al., 2011)

2.3.2 Tangente Hiperbólica

A função Tangente Hiperbólica, ou tanh, é semelhante a Sigmoide Logística, mas a escala de saída vai de -1 a 1, enquanto na Sigmoide Logística a escala é de 0 a 1. Assim como a Sigmoide Logística, também tem como desvantagem o problema da saturação. Matematicamente a função pode ser definida pela Equação 4 e graficamente representada como disposto na Figura 4.

$$tanh(x) = 2\sigma(x) - 1 \tag{4}$$

2.3.3 ReLU

A Função de Unidade Linear Retificadora (ReLU – *Rectifier Linear Unit*) pode ser definida matematicamente pela Equação 5 e sua representação gráfica pode ser vista na Figura 5 .

Uma desvantagem dessa função recebe o nome de “Dying ReLU”. Se a função

tiver uma entrada com um valor negativo, a saída será zero. Além disso, o gradiente também será zero, o que significa que a unidade ReLU permanecerá em zero indefinidamente (QUINTANILHA, 2017). Para solucionar este problema surgiu a função LeakyReLU (MAAS et al., 2013) que é uma derivação da ReLU (ANTHIMOPOULOS et al., 2016), e não será detalhada neste trabalho.

$$\phi(x) = max(0,x) \tag{5}$$

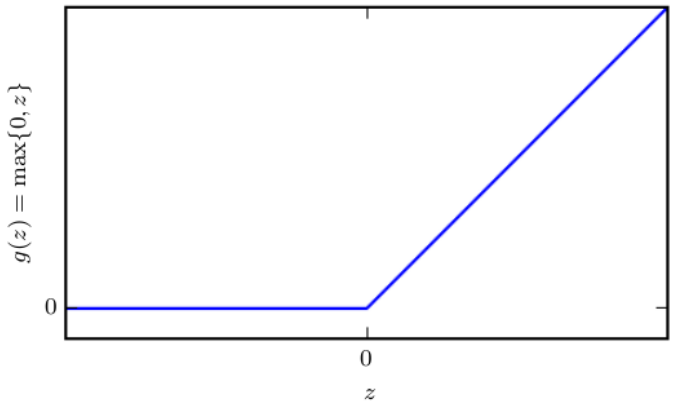

Figura 5 – ReLu
Fonte: (GOODFELLOW et al., 2016b)

2.3.4 CReLU

A função ReLU Concatenada (CReLU – *Concatenated Rectified Linear Unit*) é basicamente a concatenação da Relu aplicada à entrada positiva e negativa. Matematicamente é definida pela Equação $f'(x) = (\phi(x), \phi(-x)))$ na qual o ϕ é a função ReLU. Está função foi projetada para permitir explicitamente acertos positivos e negativos, visando aliviar a redundância entre filtros de convolução nas Redes Convolucionais causados pela não linearidade da ReLU (SHANG et al., 2016). Redes Convolucionais serão apresentadas mais a frente.

Existem outras funções que podem ser utilizadas como função de ativação, como a Softmax (GOODFELLOW et al., 2016b), SELU (KLAMBAUER et al., 2017), PReLU (HE et al., 2015) e ELU (CLEVERT et al., 2015) entre outras, mas não serão abordadas neste trabalho.

2.4 BACKPROPAGATION

Backpropagation é um algoritmo baseado em programação dinâmica e retropropagação de gradiente. Trata-se de um algoritmo muito eficiente que é utilizado no treinamento de uma Rede Neural MLP. Esse algoritmo é combinado com um algoritmo de aprendizado como o Gradiente Descendente no processo de treinamento (QUINTANILHA, 2017). O processo aplica descida de gradiente para tentar minimizar uma função de custo, geralmente associada aos erros cometidos pela rede, comparando-se os valores de saída obtidos pela rede e os valores desejados de saída.

O treinamento da Rede Neural com o algoritmo *Backpropagation* ocorre em dois passos. O Primeiro conhecido como *forward pass*, uma entrada é passada para a rede, a atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. No segundo passo, conhecido como *backward pass*, a saída obtida é comparada à saída desejada, se esta não estiver correta, o erro é calculado. O ajuste de erro é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo ajustados conforme o erro é retropropagado (CARVALHO, 2017).

Para realizar o ajuste dos pesos e dos *biases* é preciso se ter o peso a ser ajustado (w_{ij}), a taxa de aprendizagem (η), o valor obtido do nó ou entrada anterior (x_i) e o valor δ. A Equação 6 demostra o cálculo para o ajuste dos pesos e dos bias, considerando o *bias* sendo um peso (w_i) onde o valor da entrada ou nó anterior (x_i) é sempre 1. O cálculo é adaptado de Haykin et al. (2009).

$$w_{ij} = w_{ij} + \eta \delta_j x_i \tag{6}$$

O cálculo do ajuste de erro na última camada do segundo passo depende do valor δ definido pela Equação 7. Sendo f' a derivada da função de ativação da camada em função do potencial de ativação, j' a derivada da função de custo em função das saídas obtida (y_j) e desejada (d_j).

$$\delta_j = f'(v_j) j'(d_j, y_j) \tag{7}$$

Nas camadas ocultas, o processo é um pouco diferente. Para ajustar os pesos e os *biases* dessas camadas é necessário saber os valores dos nós que o peso está se conectando e seus respectivos pesos futuros, produzindo assim um somatório de pesos a se calcular. Com o

resultado da Equação 8, basta aplicar na Equação 6 e o peso e o *bias* será atualizado.

$$\delta_j = f'(v_j) \sum (w_{jk} \delta_k) \tag{8}$$

2.5 APRENDIZADO PROFUNDO

Métodos de Aprendizado Profundo (*Deep Learning*) são atualmente o estado da arte em muitos problemas possíveis de se resolver utilizando o aprendizado de máquina, em particular problemas de classificação. Por exemplo, abrange diversos problemas de visão computacional (KRIZHEVSKY et al., 2012). Isso se deve a dois motivos: primeiro, existe a disponibilidade de bases de dados com milhões de imagens (DENG et al., 2009; RUSSAKOVSKY et al., 2015); segundo, há computadores capazes de reduzir o tempo necessário para realizar o treinamento utilizando-se dessas bases de dados.

A área de aprendizado profundo atraiu grande atenção dos pesquisadores em particular devido ao bom desempenho de modelos neurais na base Imagenet[1] (DENG et al., 2009). Um dos primeiros modelos a obter um bom desempenho foi proposto por Krizhevsky et al. (2012), popularmente conhecido como AlexNet.

Redes Neurais Profundas, do inglês *Deep Neural Network* (DNN), basicamente são redes Multilayer Perceptron com a adição de mais camadas escondidas, dependendo das necessidade do problema a ser solucionado (YU; DENG, 2016). Essa metodologia exige uma uma grande quantidade de dados para treinamento (GOODFELLOW et al., 2016b). Quanto maior a base, maior será a possibilidade da rede generalizar novas instâncias nunca vistas.

A função de custo Entropia Cruzada (*Cross Entropy*) é comumente utilizada em tarefas de classificação por redes profundas. A Entropia Cruzada utiliza dos valores desejados e obtidos para realizar o ajuste de custo dos pesos presentes na rede (PONTI; COSTA, 2017). Considerando-se a classe real (desejada) sendo d e a predita (obtida) sendo $y = f(x)$ pode-se obter a soma das entropias cruzadas (entre a predição e a classe real) de cada classe utilizando a Equação 9 (GOODFELLOW et al., 2016b).

$$j = \sum_j d_j . log(y_j) \tag{9}$$

[1] http://www.image-net.org/

2.6 REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais (CNNs) (LECUN et al., 1989), também frequentemente chamadas de Redes Convolucionais, provaram ser muito eficazes como extratores de características na área de visão computacional (RAZAVIAN et al., 2014). Por exemplo, obtendo sucesso no reconhecimento de objetos (HE et al., 2016) e identificando faces (FARFADE et al., 2015), também obtiveram êxito em tarefas como o reconhecimento automático de fala (AMODEI et al., 2016) e síntese de voz (TACHIBANA et al., 2017; PING et al., 2017).

Uma Rede Convolucional é uma rede perceptron multicamadas localmente conectada. Para reconhecer formas bidimensionais, aplicam-se convoluções 2D na matriz de pixeis de uma dada imagem (HAYKIN et al., 2009). Ao contrário das Redes Multilayer Perceptons, as Redes Convolucionais preservam as relações locais em uma dada vizinhança (por exemplo, pixeis vizinhos em uma imagem), e aprendem uma representação interna da entrada usando filtros. Essa difícil tarefa é aprendida utilizando aprendizado supervisionada (LECUN et al., 1995). Segundo Quintanilha (2017), uma Rede Convolucional típica possui três estágios: camada convolucional (*convolutional layer*) descrita na Seção 2.6, função não linear (*nonlinear function*) como a ReLU descrita previamente na Seção 2.3 e camada de *pooling* (*pooling layer*), conforme apresentado na Seção 2.6.

2.6.1 Camada Convolucional

A camada convolucional é o principal elemento da Rede Convolucional. Ela leva em conta a estrutura da entrada, focando-se numa vizinhança bem definida. Nas camadas completamente conectadas isso não ocorre, pois cada neurônio recebe informações da entrada inteira, também implementa o compartilhamento de peso que possibilita trabalhar com grandes entradas.

A camada de convolução consiste em neurônios responsáveis por extrair diferentes recursos da sub-região das imagens de entrada (HIJAZI et al., 2015). Em um CNN para análise de imagens, neurônios são usados para extrair uma determinada característica e são agrupados em um filtro de estrutura bidimensional. Cada camada convolucional, neste caso, consiste em

vários filtros sobrepostos, gerando uma estrutura tridimensional.

Na camada convolucional, é especificado a quantidade de filtros, seus tamanhos, além da passada (também conhecida como *stride*), que define o tamanho da vizinhança que o neurônio de cada camada processará (VARGAS et al., 2016). A camada convolucional é assim chamada devido à operação de convolução que ela realiza. No caso de processamento de imagens, é comum que a primeira camada receba um tensor 3D que tem canais cores C, a altura H e a largura W. Tem, portanto, dimensões $(C \times H \times W)$. O processo de convolução é caracterizado por deslizar (convolver) o filtro através da entrada, realizando em cada posição um produto interno através do qual é obtido um escalar chamado de potencial de ativação.

As dimensões do mapa de ativação dependem do tamanho da entrada, do tamanho do filtro e de três hiperparâmetros: passada, tamanho do *kernel* e preenchimento de bordas com zeros (*zero-padding*). Se a passada (*stride*) tem valor 1, os filtros são deslocados de coluna em coluna ou de linha em linha cada vez, ou seja caminham um passo de cada vez. Ao aplicar uma camada de convolução após a outra repetidamente, o tamanho da representação diminui sucessivamente em cada camada até desaparecer. O preenchimento (*zero-padding*) pode ser usado para adicionar uma borda de zeros à entrada visando manter a saída com o mesmo tamanho de entrada após a convolução (QUINTANILHA, 2017). A Figura 6 apresenta um exemplo de camada convolucional onde o tamanho da entrada é 28×28, o tamanho do kernel é 5×5, a passada (*stride*) é 1, resultando em uma camada oculta de 24×24.

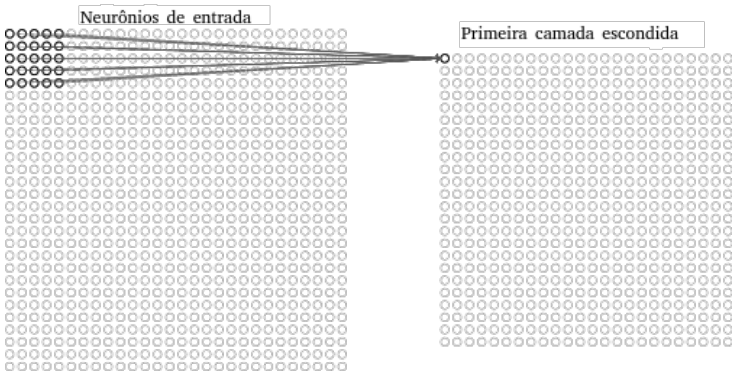

Figura 6 – Exemplo de camada convolucional

Fonte: Adaptado de Nielsen (2015)

2.6.2 Camada de *Pooling*

As camadas convolucionais diminuem a dimensionalidade de suas entradas, de acordo com os hiperparâmetros utilizados. Isso é importante para tratar o problema da maldição da dimensionalidade, comumente presente em dados como imagens e sinais de áudio. Contudo, novas reduções ainda podem ser necessárias (MAZZA, 2017). A camada de *pooling* normalmente é adicionada logo após uma camada convolucional, reduzindo o número de conexões para as camadas seguintes, um procedimento comum para o *pooling* é conhecido como *max-pooling* (NIELSEN, 2015).

Uma camada *max-pooling* retorna os valores máximos obtidos em seus filtros. Esta camada não realiza nenhum aprendizado, mas reduz o número de parâmetros a serem aprendidos nas camadas seguintes (PENHA; CASTRO, 2017). Por exemplo, cada unidade na camada de *pooling* pode resumir uma região $n \times m$ neurônios na camada anterior, considerando o n e o m sendo valores inteiros. Por exemplo, utilizando o *max-pooling* uma unidade de *pooling* simplesmente exibe a ativação máxima na região de entrada nesse caso de 2×2, conforme ilustrado no Figura 7.

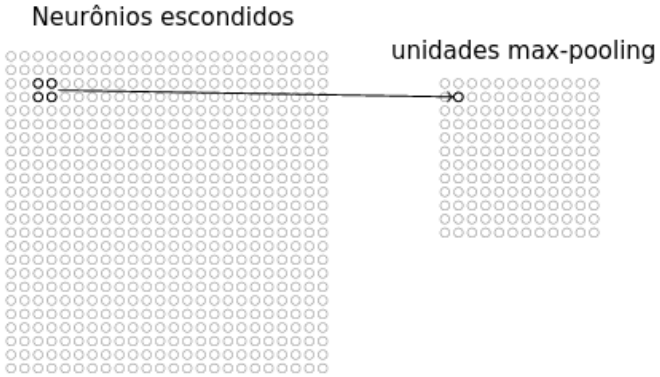

Figura 7 – Exemplo Max-Pooling
Fonte: Adaptado de Nielsen (2015)

2.6.3 Dilatação Convolucional

A dilatação convolucional (*dilated convolution*) foi referida no passado como convolução com filtro dilatado[2]. Ela desempenha um papel fundamental no algoritmo "a trous" (YU; KOLTUN, 2015), um algoritmo para a decomposição de wavelets (HOLSCHNEIDER et al., 1990). A ideia principal da dilatação convolucional é inserir "buracos" (zeros) entre os pixeis em núcleos (*kernels*) convolucionais para aumentar a resolução da imagem, permitindo assim a extração de características densas em Redes Neurais Convolucionais profundas (WANG et al., 2017a). Ela pode aumentar muito o tamanho do campo receptivo, sem aumentar o custo computacional. A convolução pode ser um caso especial com $d = 1$, considerando d o fator de dilatação. Considerando o tamanho da dilatação na i-ésima camada d_i e o mesmo tamanho de filtro k ser utilizado em todas as camadas, o tamanho efetivo do filtro em relação a camada de entrada é $(k-1)\sum_i d_i + 1$. As dilatações são tipicamente configuradas para dobrar todas as camadas $d_{i+1} = 2d_i$, então o tamanho efetivo do campo receptivo pode crescer exponencialmente. Assim, a capacidade contextual de uma Rede Neural convolucional pode ser controlada, manipulando o tamanho do filtro, o tamanho da dilatação e a profundidade da rede (YANG et al., 2017).

A dilatação convolucional é usada para ampliar o campo receptivo dos núcleos (*kernels*) convolucionais. Yu e Koltun (2015) usam camadas serializadas (*serialized layers*) com taxas crescentes de dilatação para permitir a agregação de contexto. A dilatação convolucional tem sido aplicada a uma ampla gama de tarefas, como detecção de objetos (DAI et al., 2016), geração de áudio (OORD et al., 2016a), fluxo óptico (SEVILLA-LARA et al., 2016), tradução automática (GEHRING et al., 2017) e síntese de voz (TACHIBANA et al., 2017). A Figura 8 demonstra que com o uso de dilatação convolucional, o tamanho do campo receptivo pode crescer exponencialmente. Na Figura 8(a) é apresentada um filtro convolucional 3×3. Na Figura 8(b) é apresentada a convolução com fator de dilatação igual a 2 ($d = 2$), aplicada no campo receptivo anterior (3×3) , resultando em um campo receptivo de tamanho 7×7. Na Figura 8(c) é apresentada a convolução com fator de dilatação igual a 4 ($d = 4$), aplicada no

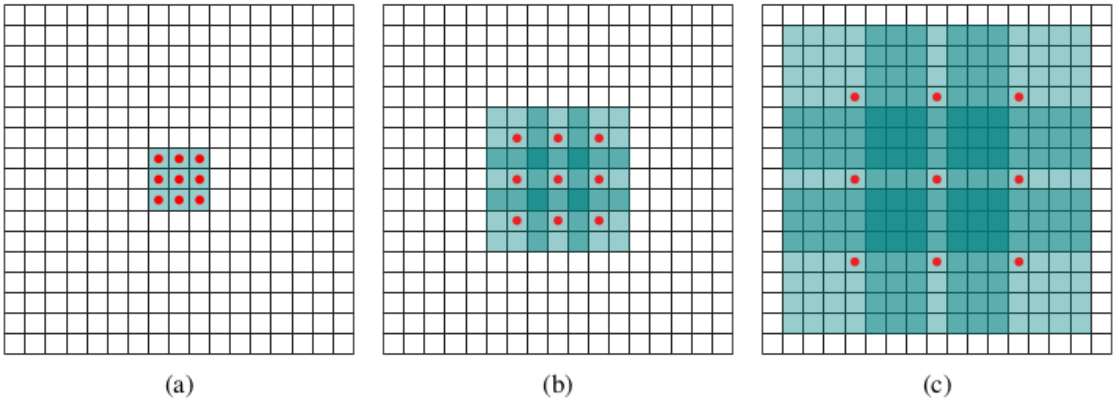

Figura 8 – Exemplo de dilatação convolucional

Fonte: Adaptado de Yu e Koltun (2015)

campo receptivo anterior (7×7), resultando em um campo receptivo 15×15.

2.7 REDES NEURAIS RECORRENTES

Redes Neurais Recorrentes (RNRs, do inglês *Recurrent Neural Networks*) foram criadas na década de 1980, mas ganharam popularidade recentemente com os avanços da aprendizagem profunda e com o crescente poder computacional das unidades de processamento gráfico (QUINTANILHA, 2017). As RNRs têm sido usadas como estado da arte para diversas tarefas como tradução automática (FIRAT et al., 2016), modelagem de linguagem (MIKOLOV et al., 2010; MIKOLOV et al., 2013), entre outras.

A diferença entre uma Rede Neural Feedforward e uma RNR pode parecer simples, porém as implicações para o aprendizado sequencial são de longo alcance. Uma Rede Feedforward só pode mapear dados da entrada para a saída, enquanto uma RNR pode mapear todas as entradas anteriores (o histórico de entradas) para a saída (GRAVES, 2012). Dessa forma, em uma RNR a saída da rede depende tanto da entrada atual quanto do estado atual. Essa característica permite a possibilidade de realizar computação dependente do contexto e aprender dependências a longo prazo. Uma entrada que é fornecida a uma rede recorrente em um instante de tempo t pode alterar o comportamento dessa rede em um momento $t + k$, considerando $k > 0$ (BEZERRA, 2016). Uma Representação simples de uma RNR, contendo

[2] O termo "dilatação convolucional" foi adotado em vez de "convolução com filtro dilatado" para esclarecer que nenhum "filtro dilatado" é construído ou representado (YU; KOLTUN, 2015)

apenas uma camada escondida pode ser vista na Figura 9.

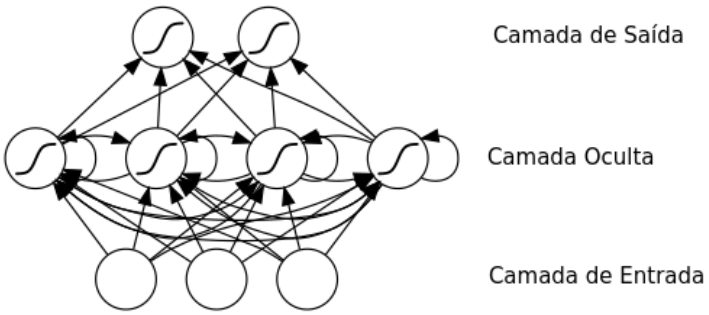

Figura 9 – Rede Neural Recorrente
Fonte: Adaptado de Graves (2012)

Em geral, uma Rede Neural Recorrente pode ser definida em duas partes principais. A primeira parte é uma função que mapeia a entrada e o estado anterior para o estado atual. A segunda parte é uma outra função que mapeia o estado atual para a saída. Um exemplo de RNR simples é dado no qual a primeira função pode ser representada pela Equação 10, e a segunda pela Equação 11. Para simplificar, a rede opera sobre uma sequência de entrada que contém vetores $\mathbf{x}^{(t)} \in \mathbb{R}^D$ com o índice de intervalo de tempo t variando de 1 a T. Tais vetores tem dimensionalidade D. Na prática, as RNRs operam em mini-lotes das sequências, com um comprimento de sequência T diferente para cada membro do mini-lote (QUINTANILHA, 2017).

$$h^{(t)} = tanh(\mathbf{W}_{xh}^T \mathbf{x}^{(t)} + \mathbf{W}_{hh}^T \mathbf{h}^{(t-1)} + b_h) \tag{10}$$

$$y^{(t)} = \mathbf{W}_{hy}^T \mathbf{h}^{(t)} + b_y \tag{11}$$

Nas equações dadas, $\mathbf{h}^{(t)}$ representa a memória da rede e o $\mathbf{y}^{(t)}$ a saída da rede no intervalo de tempo t. A matriz W_{xh}^T são os pesos que conectam o vetor de entrada $x^{(t)}$ ao vetor oculto $h^{(t)}$. A matriz W_{hh}^T conecta $h^{(t-1)}$ (isto é, o vetor oculto produzido pela entrada anterior $x^{(t-1)}$) ao vetor em produção $h^{(t)}$. O vetor b_h representa os *biases* da camada oculta. A função tangente hiperbólica é representada por $tanh$. Por fim, a matriz W_{hy}^T por sua vez conectada o vetor oculto $h^{(t)}$ ao vetor de saída $y^{(t)}$ juntamente com os *biases* b_y. A Figura 10 ilustra o processo da topologia de uma RNR com uma camada recorrente.

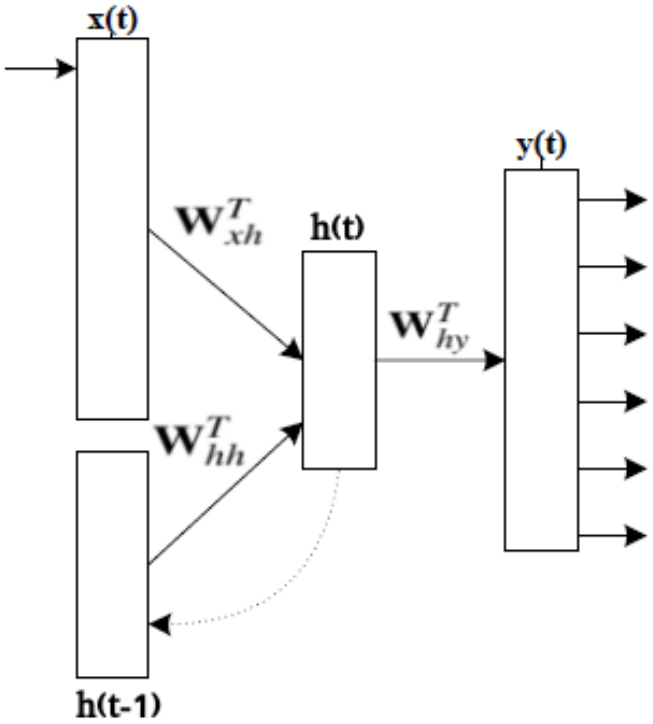

Figura 10 – Topologia de uma RNR com uma camada recorrente
Fonte: Adaptado de Guo (2013)

2.7.1 Long Short Term Memory

A grande vantagem no uso da Rede Neural Recorrente é que com ela é possível mapear informações em sequências de entrada e saída. Infelizmente, devido ao fato da dissipação do gradiente (GOODFELLOW et al., 2016b) e explosão de gradiente causados por multiplicações de matrizes (PASCANU et al., 2013), a informação não é propagada por um longo período de tempo. Por exemplo, dado um modelo de linguagem hipotético que tem o objetivo de adivinhar a próxima palavra em uma frase, e utilizando a seguinte frase: “As nuvens estão no [Próxima palavra]”, não é necessário nenhum contexto adicional, é bem provável que a próxima palavra seja “céu”. Agora o modelo de linguagem deve prever a última palavra da seguinte frase: “Eu cresci em Matelândia, Brasil. Estou estudando Ciência da Computação e falo fluentemente [Próxima palavra]”. As duas últimas palavras da frase (“falo fluentemente”) indicam que a próxima palavra tem uma grande probabilidade de ser o nome de uma língua, mas para descobrir qual a língua, é necessário saber o contexto do falante, sugerido no texto prévio pelo item “Matelândia, Brasil”. Devido à dissipação do gradiente, as RNRs são incapazes de aprender a conectar informação de longo prazo (QUINTANILHA, 2017).

Para solucionar o problema surgiu a rede *Long Short Term Memory* (LSTM). A arquitetura LSTM é inspirada na área de circuitos digitais, sendo constituída por um conjunto de subredes conectadas recorrentemente, que são conhecidos como blocos de memória. Cada um desses blocos possui uma célula de memória c_t, sendo uma ou mais células auto conectadas. Também possui três unidades, conhecidas como porta de entrada i_t, porta de saída o_t e a porta de esquecimento f_t (GRAVES, 2012). As portas da célula de memória LSTM permitem o acesso a informações e o armazenamento por um longo período de tempo, minimizando o problema da dissipação do gradiente. Se o potencial de ativação não for alto o suficiente para ativar a porta de entrada, a informação na célula não será sobrescrita por novas entradas e poderá ser utilizada por um longo prazo para futuras saídas (GRAVES, 2012).

A porta de entrada é responsável por atribuir informação na célula de memória. A ativação da porta de entrada pode ser expressa pela Equação 12, considerando x_t sendo o valor da entrada no tempo t, h_{t-1} sendo a saída anterior do neurônio, c_{t-1} sendo o valor anterior da ativação da célula, σ sendo a função de ativação e b sendo o *bias*. Nessa porta normalmente utiliza-se a função de ativação Sigmoide Logística (YU; DENG, 2014).

$$i_t = \sigma(w^{xi}x_t + w^{hi}h_{t-1} + w^{ci}c_{t-1} + b^i) \tag{12}$$

A porta de esquecimento mantém ou esquece as informações na célula LSTM. Utilizando o valor recebido é efetuado um cálculo, o mesmo pode ser representado pela Equação 13. O cálculo resulta no quanto da informação de entrada do neurônio será mantida, ou seja, será armazenada. Geralmente nessa porta aplica-se a função de ativação Sigmoide Logística (YU; DENG, 2014).

$$f_t = \sigma(w^{xf}x_t + w^{hf}h_{t-1} + w^{cf}c_{t-1} + b^{pf}) \tag{13}$$

A célula de memória também necessita de um cálculo para sua ativação, e este pode ser visualizado na Equação 14, onde "$\circ$" denota a multiplicação elementar. A função de ativação fica a critério do projetista (GRAVES, 2012). A célula de memoria LSTM pode ser vista na Figura 11.

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(w^{xc}x_t + w^{hc}h_{t-1} + b^c) \tag{14}$$

A função da porta de saída é de ler as informações da célula de memória e encaminha-la para a rede recorrente. Conforme o valor recebido é armazenado o quanto de saída será desejado. A Equação 15 apresenta os cálculos necessários para computar a porta de saída da rede. A Equação 16 computa a saída da celula LSTM. A função de ativação fica a critério do projetista (GRAVES, 2012). A Figura 11 mostra um bloco de memória LSTM com suas

respectivas portas.

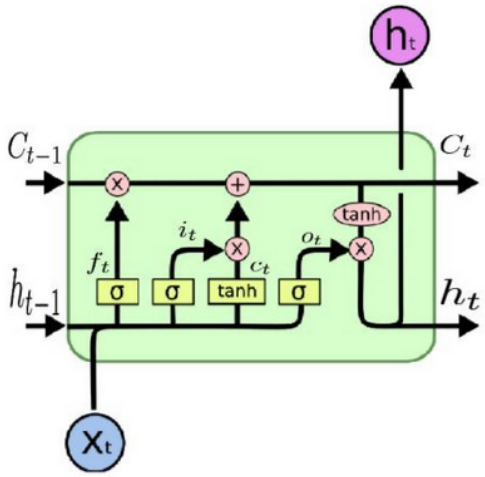

Figura 11 – Bloco de Memória LSTM com uma célula

Fonte: Adaptado de Olah (2015)

$$o_t = \sigma(w^{xo}x_t + w^{ho}h_{t-1} + w^{co}c_t + b^o) \tag{15}$$

$$h_t = o_t \circ tanh(c_t) \tag{16}$$

2.8 REDES NEURAIS QUASE RECORRENTES

As Redes Neurais Recorrentes, incluindo as LSTMs, possuem limitações quanto a sua capacidade de lidar com tarefas que envolvam sequências de entrada muito longas. Isso ocorre, pois o cálculo de estados para partes diferentes da sequência não pode ocorrer em paralelo. Na tentativa de solucionar esse problema, surgem as Redes Neurais Quase Recorrentes (QRNNs, do inglês *Quasi-Recurrent Neural Networks*) (BRADBURY et al., 2016). As QRNNs, como as Redes Neurais Convolucionais, permitem a computação massivamente paralela, permitindo alta taxa de transferência e bom desempenho para sequências longas. Bradbury et al. (2016) descrevem aplicações das QRNNs para várias tarefas de processamento de linguagem natural. Os modelos apresentados superaram os principais modelos baseados em LSTM. As QRNNs também reduziram drasticamente o tempo de treino dos modelos em comparação com os

modelos baseados em LSTM.

Uma camada quase recorrente é formada por dois tipos de subcomponentes, o subcomponente de convolução e o de *pooling* que são semelhante as camadas convolucionais e camada de *pooling* em CNNs. O componente convolucional permite a computação massivamente paralela em lotes de instâncias. Por sua vez, o componente de *pooling*, como as camadas de *pooling* em CNNs, também permite a computação massivamente paralela da entrada. Dada uma sequência de entrada $X \in \mathbb{R}^{t \times n}$ de t vetores n-dimensionais $x_1, x_2, ..., x_{n-1}, x_n$, o subcomponente convolucional de um QRNN executa convoluções na dimensão do intervalo de tempo com um banco de m filtros, produzindo uma sequência $Z \in \mathbb{R}^{t \times m}$ de t vetores m-dimensionais z_t. Para terem um bom desempenho em tarefas que incluem a previsão do próximo símbolo, os filtros não devem permitir que o cálculo de qualquer intervalo de tempo fornecido acesse informações de intervalos de tempo futuros. Ou seja, com filtros de largura k, cada saída z_t depende apenas de x_{t-k+1} até entrada x_t. Esse conceito, é conhecido como uma convolução mascarada (OORD et al., 2016b).

As QRNNs utilizam aplicação de convoluções adicionais com bancos de filtros separados para obter sequências de vetores para as portas elementares que são necessárias para a função de *pooling*. Durante o *pooling*, os vetores candidatos Z são passados pela função Tangente Hiperbólica (descrita na Seção 2.3). As portas usam a função de ativação Sigmoide Logística (Seção 2.3). A função de *pooling* necessita de uma porta de esquecimento F e uma porta de saída O em cada intervalo de tempo. O conjunto completo de cálculos no componente convolucional está disposto nas Equações 17, 18, 19.

$$Z = tanh(W_z * X) \tag{17}$$

$$F = \sigma(W_f * X) \tag{18}$$

$$O = \sigma(W_o * X) \tag{19}$$

onde Z representa os vetores candidatos obtidos na camada convolucional, F é a porta de esquecimento, O é a porta de saída, W_z, W_f e W_o, cada um em $\mathbb{R}^{k \times n \times m}$, são os bancos de filtros convolucionais e "*" representam convoluções mascaradas (OORD et al., 2016b) ao longo da dimensão do intervalo de tempo.

Funções adequadas para o subcomponente de *pooling* podem ser construídas a partir das portas da célula LSTM tradicional. A função necessária deve ser capaz de misturar estados através de intervalos de tempo, mas atuar independentemente em cada canal do vetor de estado. Balduzzi e Ghifary (2016) apresentam uma solução simples denominada de *pooling* dinâmico médio (*dynamic average pooling*). Bradbury et al. (2016) apresentam algumas alternativas para adaptar *pooling* dinâmico médio no contexto de QRNNs.

A primeira alternativa consiste em utilizar a porta de esquecimento (F) e o vetor candidato (Z) da QRNN para produzir uma saída da camada (H), denominando-a como *f-pooling* e representada matematicamente pela Equação 20.

$$h_t = f_t h_{t-1} + (1 - f_t) z_t \tag{20}$$

Uma segunda alternativa para implementar a função do subcomponente de *pooling* inclui uma porta de saída (O), sendo denominada *fo-pooling* e representada matematicamente pelas Equações 21, 22.

$$c_t = f_t c_{t-1} + (1 - f_t) z_t \tag{21}$$

$$h_t = o_t c_t \tag{22}$$

Uma terceira alternativa para o componente de *pooling* inclui uma entrada independente i, sendo denominada *ifo-pooling* e representada matematicamente pelas Equações 23 e 24.

$$c_t = f_t c_{t-1} + i_t z_t \tag{23}$$

$$h_t = o_t c_t \tag{24}$$

Uma única camada QRNN executa, assim, uma combinação linear controlada de recursos convolucionais dependente da entrada, seguida por um *pooling*. Como nas Redes Neurais Convolucionais, duas ou mais camadas QRNN devem ser empilhadas para criar um modelo com a capacidade de aproximar funções mais complexas.

2.9 MECANISMO DE ATENÇÃO

Recentemente, as redes neurais recorrentes (RNNs) tem sido uma tecnologia muito utilizada para mapear uma sequência para outra sequência, especialmente no campo de processamento de linguagem natural e tradução automática (CHO et al., 2014). O Mapeamento de sequência baseado em Redes Neurais Recorrentes possuem alguns problemas (TACHIBANA et al., 2017). Um problema com a abordagem codificador-decodificador (AutoEncoder) é que uma Rede Neural precisa ser capaz de comprimir todas as informações necessárias de uma sentença de entrada em um vetor de tamanho fixo. Isso pode tornar difícil trabalhar com

sentenças longas, especialmente aquelas que são mais longas que as utilizadas no treinamento. Para solucionar este problema foi proposto uma técnica chamada de mecanismo de atenção, que se tornou muito utilizado em mapeamento de sequência (seq2seq). Se trata de uma extensão ao modelo AutoEncoder que aprende a alinhar e predizer em conjunto de forma que ao predizer um determinado fragmento da saída procura na entrada as posições chave que possuem informação mais relevante e concentra-se nessas posições. No caso da tradução automática, o modelo prevê uma palavra baseado nas palavras de origem que possuem as informações mais relevantes (BAHDANAU et al., 2014).

Bahdanau et al. (2014) mostraram como é possível utilizar o mecanismo de atenção para redes neurais recorrentes. Enquanto Gehring et al. (2017) propuseram uma ideia de como usar o mecanismo de atenção com Redes Neurais Convolucionais para mapeamento de sequencias, e mostraram que o método é bastante eficaz para a tradução automática.

2.10 HIGHWAY NETWORKS

Muitos avanços no aprendizado de máquina foram alcançados com a utilização de Redes Neurais Artificiais grandes e profundas. A profundidade da rede (o número de camadas sucessivas), desempenha um papel importante nesses avanços. Por exemplo, Yu et al. (2013) demonstram a superioridade de redes mais profundas para a tarefa de conversão de fala em texto. O treinamento de uma Rede Neural Artificial muito profunda enfrenta alguns problemas. O empilhamento de várias transformações não-lineares em redes *feed-forward* convencionais normalmente resultam em uma baixa propagação de ativações e gradientes durante o treinamento pelo algoritmo Backpropagation, dificultando a investigação dos benefícios do uso de Redes Neurais profundas.

As *highway networks* (SRIVASTAVA et al., 2015a) foram projetadas para suprir essa demanda. Foram inspiradas nas redes recorrentes LSTM. Os autores modificaram a arquitetura de redes *feedforward* muito profundas, de modo que o fluxo de informações entre as camadas seja mais fácil e rápido. Utiliza um mecanismo de filtragem adaptável inspirado no LSTM, que permite alguns caminhos nos quais a informação pode fluir por muitas camadas sem atenuação. Os autores chamaram esses caminhos de "*information highways*" (vias de informação). Ainda demonstraram que as *highway networks* profundas podem ser treinadas diretamente usando o algoritmo gradiente descendente estocástico (SGD) (BOTTOU, 2010).

Uma camada totalmente conectada pode ser definida pela Equação 25, considerando H uma transformação não linear, geralmente seguida por uma função de ativação também não linear, X é o vetor de entrada da camada e W_H os pesos, Y o vetor de saída da camada, os *biases* e os índices foram ocultados por simplicidade.

$$Y = H(X, W_H) \tag{25}$$

Para uma camada *highway*, os autores definiram adicionalmente duas transformações não lineares $T(x, W_T)$ e $C(x, W_C)$ adaptarando a Equação 25 para acomodar esses termos, resultando na Equação 26, T sendo a operação da porta de transformação (*transform gate*) , C a operação da porta de transporte (*carry gate*), $\circ$ denotando a multiplicação elementar, considerando C como $1 - T$ pode-se obter a Equação 27 eliminando a necessidade da porta de transporte.

$$y = H(x, W_H) \circ T(x, W_T) + x \circ C(x, W_C) \tag{26}$$

$$y = H(x, W_H) \circ T(x, W_T) + x \circ (1 - T(x, W_T)) \tag{27}$$

Assim, dependendo da saída da porta de transformação, uma camada *highway* pode atuar tanto como uma camada totalmente conectada convencional quanto como uma camada bastante simples que repassa suas entradas para a próxima camada sem modificá-las, ou ainda como um meio termo entre os dois tipos de camada. Assim como uma camada convencional consiste em várias unidades de computação de tal forma que a i-ésima unidade calcula $yi = H_i(x)$, uma *highway network* consiste em vários blocos, de tal forma que o i-ésimo bloco calcula um estado do bloco $H_i(x)$ e a saída da porta de transformação $T_i(x)$, resultando na saída de bloco $y_i = H_i(x) \circ T_i(x) + x_i \circ (1 - T_i(x))$, que é conectado à próxima camada. A Figura 12 apresentação dois blocos *highway*.

2.11 ESPECTROGRAMAS E MFCCS

O pré-processamento do áudio desempenha um papel importante em sistemas de fala, seja no reconhecimento automático de fala (do inglês, *Automatic Speech Recognition*) (AMODEI et al., 2016), no reconhecimento de locutor (BREDIN, 2017), na síntese de

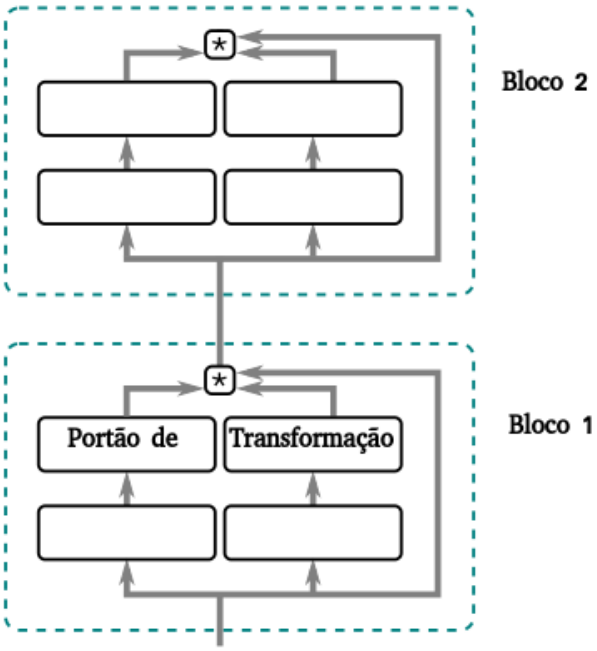

Figura 12 – Blocos de uma *highway networks*
Fonte: Adaptado de Greff et al. (2016)

voz (WANG et al., 2017b; SHEN et al., 2017; PING et al., 2017; ARIK et al., 2017; TACHIBANA et al., 2017; ARIK et al., 2017) e nas diferentes aplicações que envolvam fala. A técnica de extração de atributos Mel-Frequency Cepstral Coefficients (MFCCs)(DAVIS; MERMELSTEIN, 1990) foi muito popular por um longo tempo, porém recentemente, os bancos de filtros estão se tornando cada vez mais populares (FAYEK, 2018).

Bancos de filtros e MFCCs são obtidos com procedimentos parecidos, quando os bancos de filtros são computados com mais alguns passos extras é possível obter os MFCCs. Resumidamente, um sinal de áudio passa por um filtro de pré-ênfase, sendo logo em seguida cortado em quadros sobrepostos e uma função de janela é aplicada a cada quadro, depois é realizado uma Transformada de Fourier de Curto Prazo em cada quadro e calculado o espectro de energia, e após é calculado os bancos de filtros. Para a obtenção dos MFCCs, uma Transformada Discreta de Cosseno (DCT) é aplicada aos bancos de filtros mantendo um número de coeficientes resultantes, enquanto o restante é descartado (MOHAMED, 2014).

A primeira etapa é aplicar um filtro de pré-ênfase no sinal para amplificar as altas frequências. O filtro de pré-ênfase é usado por diferentes motivos. Primeiramente, equilibra o espectro de frequências, pois as frequências altas geralmente possuem magnitudes menores em comparação com frequências mais baixas. Em segundo lugar, também pode melhorar o relação sinal-ruído (SNR) (RIBANI et al., 2004). Em terceiro lugar, previne problemas numéricos durante a operação de transformada de Fourier, o que não deve ser um problema para as

implementações atuais (FAYEK, 2018).

Após a pré-ênfase, se faz necessário dividir o sinal em curtos quadros de tempo. Isso é necessário pois a fala é um sinal não estacionário, o que significa que suas propriedades estatísticas não são constantes ao longo do tempo. Em vez de analisar todo o sinal, é necessário extrair dados de uma pequena janela de fala que poderia caracterizar um fone (Seção 2.1) ou caractere específico e para os quais pode-se fazer a suposição de ser estacionário (QUINTANILHA, 2017). Por esse motivo, na maioria dos casos, não faz sentido fazer a Transformada de Fourier em todo o sinal, perdendo os contornos de frequência do sinal ao longo do tempo. Portanto, uma Transformada de Fourier sobre esses curtos quadros de tempo se faz necessária. Assim possibilitando obter uma boa aproximação dos contornos de frequência do sinal pela concatenação de quadros adjacentes (GORDILLO, 2013).

O passo final para se obter o banco de filtros é a aplicação de filtros triangulares, em uma escala de Mel para o espectro de energia assim extraindo as bandas de frequência. A escala de Mel tem como objetivo imitar a percepção não-linear do ouvido humano, sendo mais discriminativa nas frequências mais baixas e menos discriminativa nas frequências mais altas (FAYEK, 2018). Após obter as energias filtradas na escala de Mel, pega-se o seu logaritmo, desta forma obtendo o espectrograma de mel (QUINTANILHA, 2017). Para obter um espectrograma linear, não se deve filtrar o espectro de energia para a escala de mel, assim mantendo-o linear.

Os coeficientes do banco de filtros são muito correlacionados, o que pode causar alguns problemas em alguns algoritmos de aprendizado de máquina. Portanto, pode-se aplicar a Transformada de Cosseno Discreta (DCT) para descolar os coeficientes do banco de filtros e gerar uma representação comprimida dos bancos de filtros (FAYEK, 2018), obtendo-se assim os *Mel-Frequency Cepstral Coefficients* (MFCCs). Vários trabalhos (PING et al., 2017; ARIK et al., 2017; SHEN et al., 2017; WANG et al., 2017b; TACHIBANA et al., 2017) mostraram que é possível obter bons resultados para síntese de voz utilizando espectogramas, como alternativa aos MFCCs.

2.12 MODELOS PONTA A PONTA PARA SÍNTESE DE VOZ

Com o advento do aprendizado profundo, os sistemas de síntese de voz evoluíram muito, e ainda estão em processo de evolução. Modelos baseados em Redes

Tabela 1 – Pontuação MOS.

Avaliação	Qualidade	Distorção
5	Excelente	Imperceptível
4	Boa	Apenas perceptível, mas não irritante
3	Razoável	Perceptível e ligeiramente irritante
2	Pobre	Irritante, mas não inutilizável
1	Ruim	Muito irritante e inutilizável

Fonte: Adaptado de Ribeiro et al. (2011)

Neurais Recorrentes ganharam destaque, porém esses modelos possuem um elevado custo computacional. Isso motivou diversos trabalhos que visam diminuir o custo computacional, principalmente com a utilização de camadas convolucionais, mais mantendo a qualidade dos áudios gerados.

Os modelos de síntese de voz utilizam a Pontuação de Opinião Média (MOS – Mean Opinion Score) para comparar a qualidade dos modelos. MOS também é usada para avaliar muitos métodos de processamento de sinal. Como os estudos de qualidade laboratorial são demorados e caros, os pesquisadores geralmente realizam pequenos estudos com menor significância estatística ou usam medidas objetivas que se aproximam apenas da percepção humana. Por esse motivo Ribeiro et al. (2011) propuseram uma medida econômica e conveniente chamada crowdMOS, obtida por meio da participação de usuários da Internet em um estudo de escuta semelhante ao MOS. Os usuários ouvem e avaliam sentenças, usando seu próprio hardware, em um ambiente de sua escolha. Como esses indivíduos não podem ser supervisionados, também propuseram métodos para detectar e descartar pontuações imprecisas.

Para automatizar o teste do crowdMOS, os autores ofereceram um conjunto de ferramentas de código aberto livremente distribuíveis para o Amazon Mechanical Turk, uma plataforma projetada para facilitar o *crowdsourcing*. Essas ferramentas implementam a metodologia de testes de MOS descrita no artigo de Ribeiro et al. (2011). Os usuários avaliam o áudio de forma similar a mostrada na Tabela 1. Nesta seção será abordado o estado da arte na síntese de voz.

2.12.1 Deep Voice 1

O modelo proposto por Arik et al. (2017) foi o primeiro modelo baseado unicamente em Redes Neurais Artificiais para tarefa de síntese de voz. Tais modelos são conhecidos como

modelos ponta a ponta (*end-to-end*) e se diferem de trabalhos anteriores que utilizavam Redes Neurais Artificiais apenas em alguns subcomponentes de um modelo maior para síntese de voz. Por exemplo, Rao et al. (2015) propôs o uso de Redes Neurais Recorrentes LSTM para um modelo de conversão grafema para fonema. Zen e Sak (2015) utilizaram também redes LSTM para um modelo de previsão de duração de fonemas.

Deep Voice 1 é baseado nos modelos tradicionais, de conversão de texto em voz pipelines (TAYLOR, 2009), e adota a mesma estrutura, porém substitui todos os componentes por redes neurais e usa recursos mais simples. Pelo modelo ser baseado totalmente em Redes Neurais Artificiais o sistema é mais aplicável a novos conjuntos de dados, vozes e domínios sem necessitar de quaisquer dados manuais, notação ou engenharia de recursos adicionais. Essa tarefa requer profissionais trabalhando por dias em modelos tradicionais. A capacidade de lidar com novos conjuntos de dados sem demandar uma reengenharia do sistema de síntese de voz foi demostrada no artigo de Arik et al. (2017) por meio do retreinamento de todas as redes em um conjunto de dados novo, e o tempo necessário foi de apenas algumas horas de esforço manual e o tempo de *hardware* necessário para treinar as redes. As Redes Neurais de Deep Voice 1 utilizam camadas convolucionais e camadas quase recorrentes. Deep Voice 1 no melhor experimento obteve uma pontuação de opinião média de 3,94 no inglês americano.

2.12.2 Tacotron

Tacotron (WANG et al., 2017b), diferentemente do modelo Deep Voice 1 que possuía varias Redes Neurais Artificiais e que cada componente é treinado de forma independente, se utiliza apenas uma Rede Neural que é treinada de ponta a ponta. O modelo inclui um codificador (*encoder*), um decodificador (*decoder*) com um mecanismo de atenção (BAHDANAU et al., 2014) e uma rede de pós-processamento. Esses módulos são composto por um banco de filtros convolucionais, por *skip connections* aplicadas em Highway Networks (SRIVASTAVA et al., 2015a), por camadas com Unidades Recorrentes Bloqueadas (GRUs) (CHUNG et al., 2014) Bidirecionais e por camadas com recorrência sem portas (recorrência simples). Tacotron utiliza o algoritmo Griffin-Lim (GRIFFIN; LIM, 1984), para sintetizar formas de onda no domínio do tempo a partir dos espectrogramas. Griffin-Lim é um algoritmo para estimar um sinal a partir de sua Transformada de Fourier de Curto Tempo Modificada. Tacotron no melhor experimento obteve uma pontuação de opinião média de 3,82 no inglês americano. Uma importante implementação do Tacotron é o modelo TTS da Mozilla (MOZILLA, 2019), nesse modelo

os autores propuseram diversas melhorias visando o melhor desempenho do modelo. Dentre as melhorias propostas estão a utilização de atenção sensível à localização (CHOROWSKI et al., 2015), decaimento de pesos (KROGH; HERTZ, 1992) e transcrição fonética.

2.12.3 Deep Voice 2

O modelo de Arık et al. (2017) introduziu uma técnica para aumentar o desempenho dos modelos de síntese de voz, a conversão de texto em fala utilizando encadeamentos de locutores, ou seja, treinamento com muitos locutores, visando gerar vozes diferentes a partir de um único modelo. Também demostrou melhorias para os dois modelos de síntese de voz, Deep Voice 1 e Tacotron. Ainda apresentou o modelo Deep Voice 2, que é baseado no Deep Voice 1. Uma diferença importante entre o Deep Voice 2 e o Deep Voice 1 é a separação dos modelos de duração e frequência do fonema. O Deep Voice 1 tem um modelo único para prever conjuntamente a duração do fonema e o perfil de frequência. No Deep Voice 2, as durações dos fonemas são previstas primeiro e depois são usadas como entradas para o modelo de frequência. Os autores demonstraram uma melhoria significativa na qualidade de áudio em relação ao Deep Voice 1, compararam Deep Voice 1, Deep Voice 2 e Tacotron utilizando os mesmos avaliadores no crowdMOS, enquanto Deep Voice 1 obteve um MOS de 2,05, Deep Voice 2 obteve um MOS de 2,96 e Tacotron obteve um MOS de 2,57.

Arık et al. (2017) propuseram também uma melhoria no Tacotron substituindo o Griffin-Lim por um *vocoder* baseado na WaveNet (OORD et al., 2016a), demonstrando uma melhoria significativa na qualidade do áudio, tal modelo obteve MOS de 4,17. Em seguida, demonstraram a técnica para treinamento utilizando vários locutores para Deep Voice 2 e Tacotron utilizando dois conjuntos de dados diferentes. Os autores mostraram ainda que um único sistema de síntese neural pode aprender centenas de vozes únicas a partir de menos de meia hora de dados por locutor, ao mesmo tempo em que obtém uma síntese de alta qualidade de áudio (Deep Voice 2 treinado usando vários locutores obteve MOS de 3,53) e preservam muito bem as identidades dos falantes (ARIK et al., 2017).

2.12.4 Deep Voice 3

Deep Voice 3 (PING et al., 2017), é um modelo neural para síntese de voz baseado em atenção totalmente convolucional. Os autores demostraram que Deep Voice 3 é treinado dez vezes mais rápido que o Tacotron. O modelo foi treinado em mais de oitocentas horas de áudio, contendo mais de dois mil locutores. A critério de comparação os autores utilizaram três diferentes métodos para sintetizar formas de onda no domínio do tempo a partir dos espectrogramas, sendo eles o algoritmo Griffin-Lim (GRIFFIN; LIM, 1984), WaveNet (OORD et al., 2016a), e WORLD *vocoder* (MORISE et al., 2016). Os resultados indicam que a síntese de forma de onda mais natural pode ser feita com um *vocoder* neural Wavenet. Mesmo com a utilização do Wavenet tendo uma maior naturalidade na fala, o tempo de execução pode tornar o WORLD *vocoder* preferível, pois é aproximadamente 13 vezes mais rápido que a implementação WaveNet (ARIK et al., 2017; PING et al., 2017). Os autores compararam os modelos Deep Voice 2, Tacotron e Deep Voice 3, para treinamento com um único locutor o Deep Voice3 com o *vocoder* neural Wavenet obteve um MOS igual ao do Tacotron de 3,78, enquanto Deep Voice 2 obteve 2,74.

2.12.5 Tacotron2

Tacotron 2 (SHEN et al., 2017) combina o modelo Tacotron com o *vocoder* Wavenet modificado (TAMAMORI et al., 2017a). O Tacotron 2 é composto por uma rede de previsão de recursos de sequência a sequência recorrente que mapeia incorporação de caracteres em espectrogramas em escala de Mel, seguido por um modelo WaveNet modificado atuando como um *vocoder* para sintetizar formas de onda no domínio do tempo a partir desses espectrogramas. O modelo atinge uma pontuação MOS de 4,53, comparável a um MOS de 4,58 para o discurso gravado profissionalmente. Ainda demonstraram que o uso de espectrogramas de Mel como a entrada de condicionamento para WaveNet, ao invés de características linguísticas, permite uma redução significativa no tamanho da arquitetura WaveNet (SHEN et al., 2017).

2.13 SÍNTESE DE VOZ BASEADA NO MODELO DCTTS

Tachibana et al. (2017) propuseram o modelo DCTTS, um modelo neural para síntese de voz totalmente convolucional. A sua arquitetura é bastante semelhante ao Tacotron, mas é baseada em um modelo de aprendizagem que mapeia uma a sequencia para outra sequencia (*sequence-to-sequence*) (GEHRING et al., 2017) totalmente convolucional. O Tacotron, Deep Voice, Tacotron2 e Deep Voice 2, possuem uma desvantagem, utilizam muitas unidades recorrentes, que são muito caras computacionalmente. Dessa forma, o treinamento demora um longo tempo, tornando-se quase inviável para equipamentos de poder computacional relativamente modesto, assim tornando difícil estudá-los e aprimorá-los. Já os sistemas DCTTS (TACHIBANA et al., 2017) e DeepVoice3 (PING et al., 2017) utilizam unidades convolucionais, que possuem um menor custo computacional, consequentemente o tempo de treinamento desses modelos diminui. O DeepVoice3, por exemplo, tem o tempo de treinamento 10 vezes menor que o tempo de treino do Tacotron. Isso ocorre, pois modelos convolucionais exploram altamente o paralelismo de uma GPU durante o treinamento (PING et al., 2017). Diferentemente dos modelos até aqui apresentados o DCTTS utiliza o *vocoder* RTISI-LA (ZHU et al., 2007). O DCTTS utiliza dilatação convolucional ao invés de unidades recorrentes para levar informações contextuais em conta.

2.13.1 Arquitetura

DCTTS consiste em duas redes, a primeira denominada de *Text2Mel* (texto para espectrograma de mel), que tem como objetivo sintetizar um espectrograma de mel a partir de um texto de entrada e a segunda *Spectrogram Super-resolution Network* (SSRN), que converte um espectrograma de mel para o espectrograma STFT (espectrograma de transformada de Fourier de tempo curto) completo (BENESTY et al., 2011). A Figura 13 mostra a arquitetura geral do modelo.

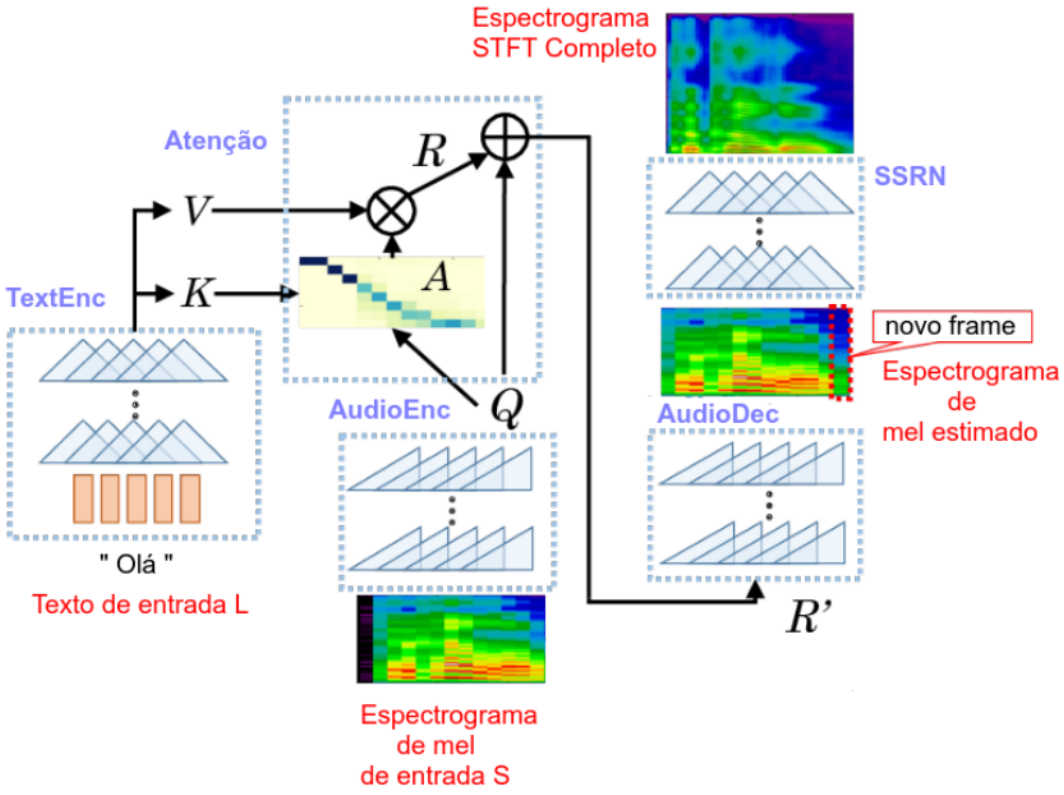

Figura 13 – Arquitetura geral do modelo DCTTS

Fonte: Adaptado de Tachibana et al. (2017)

2.13.2 Text2Mel

Essa parte do modelo é dividido em quatro submódulos, sendo eles o codificador de texto (*TextEnc*), codificador de áudio (*AudioEnc*), atenção (*Att*) e decodificador de áudio (*AudioDec*). A rede *TextEnc* primeiro codifica a sentença de entrada, denotado como $L = [l_1, .., l_N]$ que é um vetor do tipo *Char*, contendo de 1 a N caracteres, nas duas matrizes K, V. Já a rede *AudioEnc* codifica o espectrograma de Mel denotado por S ($S_{1:F,1:T}$) pertencente ao $\mathbb{R}^{F \times T}$, da fala das palavras contidas no vetor L, em um matriz Q pertencente a $\mathbb{R}^{d \times T}$, sendo T o comprimento da matriz.

$$(K, V) = TextEnc(L). \tag{28}$$

$$Q = AudioEnc(S_{1:F,1:T}). \tag{29}$$

Uma matriz de atenção A pertencente a $\mathbb{R}^{N \times T}$, avalia com que intensidade o enésimo caractere l e t-ésimo quadros de tempo $S_{1:F,t}$ estão relacionados, sendo representada matematicamente pela Equação 30.

$$A = softmax_{eixo-n}(\frac{K^T Q}{\sqrt{d}}). \tag{30}$$

O componente A_{nt} indica que o módulo está analisando o enésimo caractere l_n no tempo t, e ele analisa a região de l_n, l_{n+1} e seu entorno no tempo após o $t+1$, sendo codificados na enésima coluna de V. Assim, a matrix $R \in \mathbb{R}^{d \times T}$, decodificada para os quadros subsequentes $S_{1:F,2:T+1}$, pode ser obtida utilizando a Equação 31, tal que $Att(Q, K, V)$ é o produto da matriz V com A.

$$R = Att(Q, K, V) \tag{31}$$

O R obtido é concatenado com o áudio codificado Q, resultando em R'. Segundo os autores, seus experimentos indicaram que a concatenação é benéfica para melhorar o desempenho do modelo. Então a matriz concatenada $R' \in \mathbb{R}^{2d \times T}$ é decodificada pelo módulo *AudioDec* para sintetizar um espectrograma de mel, como demonstrado na Equação 32.

$$Y_{1:F,2:T+1} = AudioDec(R'). \tag{32}$$

A matriz resultante da última equação, é comparada com o espectrograma denotado como S, por uma função de custo L, e o erro é propagado de volta para os parâmetros da rede,

utilizando o algoritmo *BackPropagation*. A função de custo L é resultante da soma do custo de $L1$ e da divergência binária (D_{bin}), considerando M como sendo a média as Equações 34 e 33 representam matematicamente $L1$ e D_{bin} respectivamente.

$$D_{bin}(Y|S) = M(-SY + log(1 + exp(Y))) \tag{33}$$

$$L1(Y,S) = M(|Y - S|) \tag{34}$$

A função de perda final $L(Y,S)$ é representada matematicamente pela Equação 35 e seu resultado é sempre positivo e é zero quando $Y = S$, ou seja, quando o valor de saída da rede é igual ao valor real do espectrograma.

$$L(Y,S) = D_{bin}(Y|S) + L1(Y,S) \tag{35}$$

2.13.3 Spectrogram Super-resolution Network (SSRN)

Essa parte do modelo tem como objetivo sintetizar um espectrograma STFT completo, denotado por $|Z| \in \mathbb{R}^{F \times 4T}$, a partir do espectrograma de mel obtido da rede Text2Mel, $Y \in \mathbb{R}^{F \times T}$. A função de custo manteve-se a mesma que foi utilizada na rede Text2Mel, que consiste na soma da divergência binária e da distância $L1$ entre o espectrograma sintetizado (S) e o espectrograma final $|Z|$.

2.13.4 Topologia

Para a representação da topologia da rede foram utilizadas algumas convenções. A camada convolucional 1D (LECUN; BENGIO, 1998) foi denotada por $C_{k*\delta}$ considerando k o tamanho do kernel e δ sendo o fator de dilatação. O passo (*stride*) da convolução é sempre 1. Para a sequência de entrada ser sempre constante é utilizado o *zero-padding* (preenchimento com zeros), que se faz necessário pois frases de entrada podem ser de tamanhos distintos. A camada de de-convolução (*Up Sampling*) é denotada como $D_{k*\delta}$ utilizando a mesma notação de

k e δ da camada convolucional. A de-convolução utiliza sempre passo 2. A função de ativação ReLU é denotada por *ReLU* e a função de ativação Sigmoid sendo denotada por *Sig*. Utiliza-se "$\dashrightarrow$" para denotar a divisão das camadas da rede. *CharEmbed* é uma camada de incorporação de caracteres. A Tabela 2 apresenta a topologia de cada um dos quatro submódulos do modelo.

Tabela 2 – Topologia dos submódulos do modelo

$TextEnc = CharEmbed \dashrightarrow C_{1*1} \dashrightarrow ReLU \dashrightarrow C_{1*1} \dashrightarrow (HC_{3*1} \dashrightarrow HC_{3*3} \dashrightarrow HC_{3*9} \dashrightarrow HC_{3*27})^2 \dashrightarrow (HC_{3*1})^2 \dashrightarrow (HC_{1*1})^2$
$AudioEnc = C_{1*1} \dashrightarrow ReLU \dashrightarrow C_{1*1} \dashrightarrow ReLU \dashrightarrow C_{1*1} \dashrightarrow (HC_{3*1} \dashrightarrow HC_{3*3} \dashrightarrow HC_{3*9} \dashrightarrow HC_{3*27})^2 \dashrightarrow (HC_{3*3})^2$
$AudioDec = C_{1*1} \dashrightarrow HC_{3*1} \dashrightarrow HC_{3*3} \dashrightarrow HC_{3*9} \dashrightarrow HC_{3*27} \dashrightarrow (HC_{3*1})^2 \dashrightarrow (ReLU \dashrightarrow C_{1*1})^3 \dashrightarrow C_{1*1} \dashrightarrow Sig$
$SSRN = C_{1*1} \dashrightarrow HC_{3*1} \dashrightarrow HC_{3*3} \dashrightarrow (D_{2*1} \dashrightarrow HC_{3*1} \dashrightarrow HC_{3*3})^2 \dashrightarrow C_{1*1} \dashrightarrow (HC_{3*1})^2 \dashrightarrow C_{1*1} \dashrightarrow (C_{1*1} \dashrightarrow ReLU)^2 \dashrightarrow C_{1*1} \dashrightarrow Sig$

Fonte: Adaptado de Tachibana et al. (2017)

No próximo Capítulo serão descritos os materiais e métodos utilizados no desenvolvimento do projeto.

3 MATERIAIS E MÉTODOS

Neste capítulo serão descritas as etapas do projeto e as principais tecnologias empregadas. Diversas tecnologias foram empregadas para construir o sintetizador de voz, descritas na Seção 3.1. Os procedimentos para a criação do sintetizador são descritos na Seção 3.2.

3.1 MATERIAIS

A seguir, são apresentadas as tecnologias propostas para uso no trabalho.

Hardware utilizado:

Na Tabela 3 pode ser visualizada as especificações de hardware dos computadores utilizados para o treinamento dos modelos.

Tabela 3 – Especificações de hardware dos computadores utilizados no treinamento.

Especificações	**Computador 1**	**Computador 2**
Processador	i7-8700	i7-7700
Memória Ram	16 GB	32 GB
Placa de video	Nvidia GeForce Gtx Titan V	Nvidia GeForce Gtx 1080 TI
Sistema Operacional	Ubuntu 18.04	Windows 10

Fonte: Autoria Própria

Ambiente de desenvolvimento:

- Python[1]: é uma linguagem de programação de alto nível, interpretada, com suporte a diversos paradigmas como o orientado a objeto, o funcional e o imperativo. É gerenciada pela Python Software Foundation e desenvolvida pela comunidade;
- NumPy[2]: é uma biblioteca Python de código aberto, que adiciona ao Python o suporte

[1] Site oficial: https://www.python.org/
[2] Site oficial: http://www.numpy.org/

a tensores e tensores multidimensionais. Possui uma grande coleção de funções matemáticas de alto nível para manipular esses tensores. O NumPy é mantido pela comunidade. É necessária, pois outras bibliotecas citadas neste trabalho usam tensores no formato Numpy. Também foi utilizada para serialização de objetos em disco;

Bibliotecas de pré-processamento e captura de áudio:

- Pyaudio[3]: O PyAudio permitir a integração em Python para o PortAudio. PortAudio[4] é uma biblioteca de entrada/saída de áudio multiplataforma. Com o PyAudio, pode-se facilmente reproduzir e gravar áudio em diversas plataformas usando o Python. Seu propósito principal no trabalho é acesso ao microfone para coleta de áudio;
- Wave[5]: O módulo Python Wave fornece uma interface amigável para o formato de som WAV, trabalhando com áudio mono e estéreo. O módulo será usado para salvar os áudios coletados pelo Pyaudio;
- SoX[6]: é um utilitário de linha de comando multi-plataforma que pode fazer a conversão de arquivos de áudios para diversos formatos. Ele também pode aplicar vários efeitos a esses arquivos de áudio. Seu propósito principal no trabalho é converter áudio coletado com o módulo Wave para outros formatos;
- PyQt: é um empacotador desenvolvido para dar suporte a biblioteca gráfica Qt (HÜBSCHLE et al., 2011) em Python (SUMMERFIELD, 2007). Foi empregado para construir uma interface amigável para permitir que o usuário locutor forneça sua voz com mais facilidade;
- Librosa[7]: é uma biblioteca de código aberto criada para a análise de música e áudio. Seu propósito original é fornecer os blocos necessários para criar sistemas de recuperação de informações musicais, mas pode ser empregada em diversas outras tarefas de análise de áudio. Uma de suas funcionalidades é a extração dos MFCCs e dos espectrogramas (MCFEE et al., 2015);
- RNNoise[8]: é uma biblioteca de supressão de ruído que utiliza uma Rede Neural Recorrente. Combina o processamento clássico de sinais com o aprendizado profundo, criando um algoritmo de supressão de ruído em tempo real pequeno e rápido. Também possui melhor desempenho do que os sistemas tradicionais de supressão de ruído (VALIN, 2017);
- SciPy[9]: é uma biblioteca Python de código aberto criada para ser utilizada na computação

[3]Site oficial: https://people.csail.mit.edu/hubert/pyaudio/
[4]Site oficial: http://www.portaudio.com/
[5]Site oficial: https://docs.python.org/2.4/lib/module-wave.html
[6]Site oficial: http://sox.sourceforge.net/
[7]Site oficial: https://librosa.github.io/librosa/
[8]Site oficial: https://people.xiph.org/ jm/demo/rnnoise/
[9]Site oficial: https://scipy.org/scipylib/index.html

científica e computação técnica. Foi utilizada entre outras coisas, para auxiliar na extração de características dos áudios;

Biblioteca de treinamento:

- Tensorflow[10]: é uma biblioteca pública de código aberto criada e mantida por pesquisadores da Google. Tem como seu principal objetivo facilitar a implementação de modelos matemáticos para o aprendizado de máquina, com foco no Aprendizado Profundo.

3.2 MÉTODOS

Nesta seção serão abordados os métodos utilizados no desenvolvimento do projeto. O trabalho é composto pelos métodos apresentados na Figura 14.

3.2.1 Criação da Base de Áudio

Para criação da base de áudio foram utilizados textos de domínio público. Inicialmente foram utilizados artigos em destaques do Wikipédia e também conversas da córpus Chatterbot-corpus[11], criado para a construção de robôs de bate-papo (*chatbot*). Também foram utilizadas as frases propostas por Seara (1994), disponibilizados em conjuntos de frases foneticamente balanceadas, apresentando 20 conjuntos de frases, cada um contendo 10 frases.

Para a captura da fala foi desenvolvido um programa utilizando a linguagem de programação Python e a biblioteca Pyaudio foi utilizada para capturar o áudio. Posteriormente a biblioteca Wave é utilizada para armazenar em disco o arquivo de áudio em formato .wav. Para o desenvolvimento de uma interface amigável que forneça edição do texto a ser falado utilizou-se a biblioteca gráfica Qt (HÜBSCHLE et al., 2011). Em Python a utilização do Qt é possível por meio do empacotador PyQt (SUMMERFIELD, 2007). O objetivo é formar uma base com aproximadamente 10 horas de fala de um único locutor. O autor do trabalho foi o

[10]Site oficial: https://www.tensorflow.org/

[11]Github: https://github.com/gunthercox/chatterbot-corpus/tree/master/chatterbot_corpus/data/portuguese

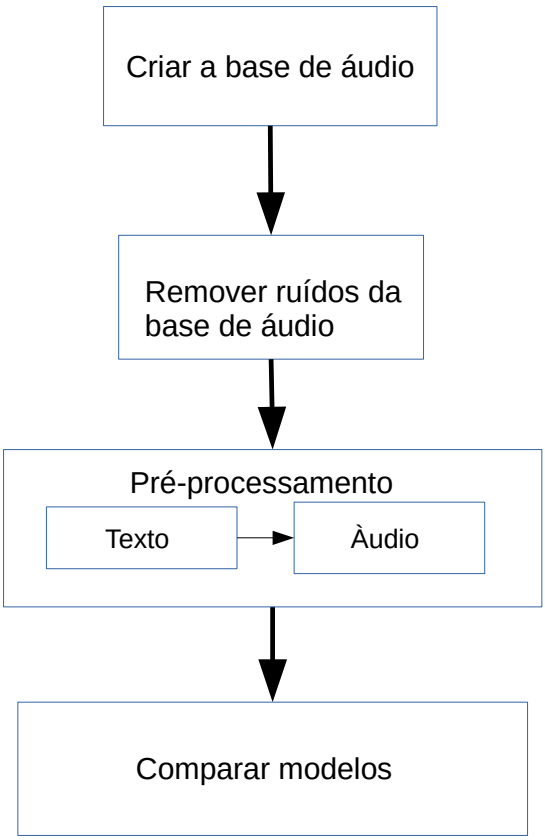

Figura 14 – Fluxograma das atividades propostas.

Fonte: Autoria Própria

locutor e a base foi denominada como TTS-Portuguese Corpus.

3.2.2 Remoção dos Ruídos da Base de Áudio

Para a síntese de voz, é necessário um pré-processamento dos áudios utilizados para treinamento, pois a presença de ruídos no som deve ser minimizada para a Rede Neural não aprender a sintetizar ruídos juntamente com a fala do locutor. Desta forma, deve existir um aumento no desempenho dos modelos. Após a base ser gravada é necessário retirar possíveis ruídos de ambiente. Como os áudios não foram gravados em estúdio com isolamento acústico,

a possibilidade de ruido é alta. Devido ao complexo trabalho que seria necessário para modelar uma Rede Neural Artificial para a remoção de ruídos, optou-se pela utilização da biblioteca de supressão de ruído RNNoise (VALIN, 2017). Ela é baseada em Redes Neurais Recorrentes, mais especificamente, Redes Neurais com Unidades Recorrentes Fechadas (GRU) (CHO et al., 2014) e demonstrou um bom desempenho para remoção de ruídos (VALIN, 2017).

3.2.3 Pré-processamento do Texto Escrito

Inicialmente, para ser possível o treinamento, é necessário pré-processar o texto escrito da base de áudio, transformando o texto em um vetor de números inteiros. Para fazer a transformação, o texto é normalizado, removendo caracteres indesejáveis e também deixando todas as palavras em letras minúsculas. Após, o texto é convertido para um vetor de números inteiros, utilizando uma cadeia de caracteres (*string*) contendo todos os caracteres válidos. Os caracteres válidos são as letras de "a" até "z", ".", "?", letras com acentuações e espaço em branco. Esses caracteres são adicionados a uma *string*, utilizando-a para converter os textos em sequências de números inteiros. Um exemplo é a conversão da frase "você é feliz?", tomando como base a *string* de caracteres válidos "_abcdefghijklmnopqrstuvwxyzçãàáêéíóôõúûâ.?". A respectiva conversão para uma lista de números inteiros seria $(22, 15, 3, 33, 0, 34, 0, 6, 5, 12, 9, 26, 40)$. O "v" é o elemento 22 da *string* de caracteres válidos, o "o" é o elemento 15, e assim por diante para converter toda a frase.

3.2.4 Extração Características

Como já foi argumentado anteriormente (Seção 2.11), a técnica de extração de atributos Coeficientes Mel Cepstrais (Mel-Frequency Cepstral Coefficients – MFCCs) (DAVIS; MERMELSTEIN, 1990) foi muito popular por um longo tempo, entretanto, com o advento do aprendizado profundo os bancos de filtros estão se tornando cada vez mais populares. Por este motivo, e considerando que na literatura boa parte dos trabalhos relacionados o utilizam, o método de extração de características escolhido foram os bancos de filtros também chamados de espectrogramas. Para realizar a extração das características é utilizada a biblioteca Librosa. Os parâmetros a serem utilizados na extração dos espectrogramas podem ser visualizado na Tabela

Tabela 4 – Parâmetros para extração das características.

Taxa de amostragem do sinal de áudio	22050
Função de janelamento STFT	Hanning (GUDMUNDSON; ANDERSON, 1996)
Tamanho do espectrograma STFT completo	$513 \times 4T$ (T depende do tamanho do áudio)
Tamanho do espectrograma de Mel	$80 \times T$ (T depende do tamanho do áudio)

Fonte: Adaptado de (TACHIBANA et al., 2017)

4.

3.2.5 Comparação entre modelos de destaque na literatura utilizando a base de áudio proposta no trabalho

Afim de comparação entre alguns modelos de destaque na literatura treinados na base de áudio proposta no trabalho, além da implementação do modelo DCTTS se fez necessário treinar outros modelos, para essa tarefa foi escolhida implementações de código aberto e foi proposto modificações para atender as necessidades de cada experimento proposto. Foram utilizadas as seguintes implementações: DCTTS (TACHIBANA et al., 2017) implementado e disponível em (CASANOVA, 2019); WaveRNN (KALCHBRENNER et al., 2018) fornecido por Wang (2018); Tacotron 1 (WANG et al., 2017b) fornecido por Park (2018b); TTS fornecido por Mozilla (2019).

3.3 EXPERIMENTOS PROPOSTOS

Nesta seção foram propostos experimentos para comparar modelos de destaques na literatura utilizando a base proposta nesse trabalho. Para manter os resultados reproduzíveis, foram utilizadas implementações de modelos de código aberto e buscou-se replicar os trabalhos relacionados da forma mais fiel possível. No entanto, alguns trabalhos não especificam totalmente os hiperparâmetros usados, então se fez necessário estimar bons parâmetros para alguns modelos para obter resultados razoáveis no conjunto de dados proposto. Como exemplo, os autores do DCTTS não especificam como a normalização foi aplicada em seu modelo (TACHIBANA et al., 2017; PARK, 2018a). Além disso, foi utilizado um conjunto de dados de voz especialmente desenvolvido para a língua portuguesa, que é pequeno quando comparado

com o conjunto de dados do inglês, o que pode ter implicação no desempenho do modelo.

- **Experimento 1.1**: este experimento replica a implementação do modelo DCTTS, treinando o modelo para o português com o TTS-Portuguese Corpus em sua forma original (sem transcrições fonéticas). O *vocoder* padrão, RTISI-LA, foi usado. Para obter uma boa conversão, testou-se diferentes opções de normalização e o melhor resultado foi obtido utilizando em todas as camadas, um *dropout* de 5% e a técnica *layer normalization* (BA et al., 2016). Adicionalmente não foi utilizada uma taxa de aprendizado fixa, conforme descrito no artigo original. Em vez disso, foi utilizado uma taxa de aprendizado inicial de 0,001 decaindo a mesma usando o esquema de decaimento da taxa de aprendizado de Noam (VASWANI et al., 2017).
- **Experimento 1.2**: é semelhante ao experimento anterior, entretanto após o áudio ser sintetizado ele é pós-processado, utilizando a biblioteca de supressão de ruído RNNoise (VALIN, 2017).
- **Experimento 2.1**: é semelhante ao experimento 1.1, mas as transcrições fonéticas são usadas para entrada da rede em vez dos textos originais. Utilizou-se a ferramenta PETRUS (SERRANI, 2015) para a conversão do texto no formato do International Phonetic Alphabet (IPA).
- **Experimento 2.2**: é semelhante ao experimento anterior, entretanto após a sintetização do áudio ele é pós-processado, utilizando a biblioteca de supressão de ruído RNNoise.
- **Experiência 3.1**: este experimento explora a modificação do modelo DCTTS para a utilização do *vocoder* WaveRNN (KALCHBRENNER et al., 2018) em vez do padrão RTISI-LA, alterando o módulo SSRN do modelo DCTTS para gerar espectrograma de Mel em vez de espectrograma STFT completo, após utiliza-se o *vocoder* WaveRNN para sintetizar o espectrograma de Mel.
- **Experimento 3.2**: é semelhante ao experimento anterior, entretanto o áudio é pós-processado após ser sintetizado pelo modelo, para tal utilizou-se da biblioteca RNNoise.
- **Experimento 4.1**: este experimento consiste em uma versão modificada do modelo DCTTS que gera diretamente as características do WORLD *vocoder* a partir do texto. Este modelo é chamado DCC2WORLD e busca explorar o uso do WORLD *vocoder* na síntese de voz.
- **Experimento 4.2**: é semelhante ao experimento anterior, entretanto após o áudio ser sintetizado ele é pós-processado, utilizando a biblioteca de supressão de ruído RNNoise (VALIN, 2017).
- **Experimento 5.1**: neste experimento explora-se o uso do modelo Tacotron 1 (WANG et al., 2017b; PARK, 2018b), treinado no TTS-Portuguese Corpus.

- **Experimento 5.2**:é similar ao experimento anterior, porém utilizou-se um *dropout* de 5% e *layer normalization* (BA et al., 2016) em todas as camadas convolucionais e recorrentes do modelo.
- **Experimento 5.3**: é semelhante ao experimento 5.1, entretanto utiliza-se um modelo pré-treinado na língua inglesa, após carregar os pesos do modelo pré-treinado é continuado o treinamento utilizando a base TTS-Portuguese Corpus.
- **Experimento 5.4**: é semelhante ao experimento anterior, entretanto após a sintetização do áudio ele é pós-processado, utilizando a biblioteca de supressão de ruído RNNoise.
- **Experimento 6.1**: neste experimento explora-se o uso do modelo TTS proposto pela Mozilla (MOZILLA, 2019), treinado no TTS-Portuguese Corpus.
- **Experimento 6.2**: é semelhante ao experimento 6.1, porém após a sintetização do áudio ele é pós-processado, utilizando a biblioteca de supressão de ruído RNNoise.
- **Experimento 6.3**: é semelhante ao experimento 6.1, entretanto, utiliza-se um modelo pré-treinado na língua inglesa, após carregar os pesos do modelo pré-treinado foi continuado o treinamento utilizando a base TTS-Portuguese Corpus.
- **Experimento 6.4**: é semelhante ao experimento anterior, mas após a sintetização o áudio foi pós-processado, utilizando a biblioteca de supressão de ruído RNNoise.

Visando uma melhor compreensão dos experimentos a Tabela 5 apresenta uma breve descrição dos experimentos. Como pode ser observado na tabela o modelo TTS da Mozilla utilizou transcrição fonética. Entretanto esse modelo não utilizou-se da ferramenta PETRUS para converter grafema em fonema. Em vez disso, o modelo utilizou a ferramenta Phonemizer (BERNARD, 2019).

O DCC2WORLD consiste em duas redes, a primeira denominada Text2World que, a partir de um texto de entrada, prevê as características F0 (frequência fundamental), AP (aperiodicidade) e SP (envelope espectral) do WORLD *vocoder*, e uma segunda rede chamada WSRN (WORLD Super-Resolution Network) que recebe como entrada os recursos do WORLD *vocoder* e aumenta o número de quadros, aumentando assim a qualidade do áudio sintetizado.

O modelo Text2World é semelhante ao modelo Text2Mel do modelo DCTTS, apenas a saída foi alterada para prever os recursos do *vocoder* WORLD em vez de prever um espectrograma de Mel, além disso, foi adicionado um *dropout* de 5 % e *layer normalization* (BA et al., 2016) em todas as camadas, adicionalmente também é usado a taxa de aprendizado de 0,001 como o valor inicial, durante o treinamento a mesma é decaída utilizando o esquema de decaimento da taxa de aprendizado de Noam (VASWANI et al., 2017).

A topologia do WSRN pode ser vista na Figura 15. Considerando que Conv denota uma camada convolucional (AMARI et al., 2003), Deconv denota uma camada

Tabela 5 – Descrição dos experimentos propostos.

Experimento	Modelo	*Vocoder*	Entrada	Transferência de Aprendizado	RNNoise
1.1	DCTTS	RTISI-LA	Grafema	Não	Não
1.2	DCTTS	RTISI-LA	Grafema	Não	Sim
2.1	DCTTS	RTISI-LA	Fonema	Não	Não
2.2	DCTTS	RTISI-LA	Fonema	Não	Sim
3.1	DCTTS	WaveRNN	Grafema	Não	Não
3.2	DCTTS	WaveRNN	Grafema	Não	Sim
4.1	DCTTS	WORLD	Grafema	Não	Não
4.2	DCTTS	WORLD	Grafema	Não	Sim
5.1	Tacotron 1	RTISI-LA	Grafema	Não	Não
5.2	Tacotron 1 + Norm	RTISI-LA	Grafema	Não	Não
5.3	Tacotron 1	RTISI-LA	Grafema	Sim	Não
5.4	Tacotron 1	RTISI-LA	Grafema	Sim	Sim
6.1	TTS Mozilla	RTISI-LA	Fonema	Não	Não
6.2	TTS Mozilla	RTISI-LA	Fonema	Não	Sim
6.3	TTS Mozilla	RTISI-LA	Fonema	Sim	Não
6.4	TTS Mozilla	RTISI-LA	Fonema	Sim	Sim

Fonte: Autoria Própria

de deconvolução, HC denota a camada convolucional seguida por ativação HighWay (TACHIBANA et al., 2017) inspirada em Highway Networks (SRIVASTAVA et al., 2015b), R:3 indica que a convolução dilatada (YU; KOLTUN, 2015; KALCHBRENNER et al., 2016) é usado na camada com o fator de dilatação igual a três, RC denota uma camada convolucional seguida por ativação ReLU (NAIR; HINTON, 2010) e SC denota uma camada convolucional seguida por ativação Sigmoide Logística (GOODFELLOW et al., 2016a). Em todas as camadas do modelo, é usado um *dropout* de 5% e *layer normalization* (BA et al., 2016). Todas as convoluções e deconvoluções são unidimensionais.

As configurações dos parâmetros para extrair os recursos do WORLD *vocoder*(F0, AP, SP) nos experimentos 4 e 6 são apresentados na Tabela 6, valores diferentes são usados para o parâmetro período de *frames*, para extrair as características do *vocoder* WORLD, optou-se por usar um valor maior para o modelo DCC2WORLD, reduzindo assim o número de *frames* a serem sintetizados, para compensar essa diminuição é usado o modelo WSRN que aumenta o número de *frames* no áudio, consequentemente aumentando sua qualidade.

Para o modelo WaveRNN utilizou-se os áudios em formato *raw* e a distribuição gaussiana foi aplicada á saída do modelo, como proposto no *vocoder* Clarinet (PING et al., 2018), optou-se por essa configuração pois nos experimentos propostos por Ping et al. (2018) a abordagem permitiu uma melhoria em comparação com outras abordagens.

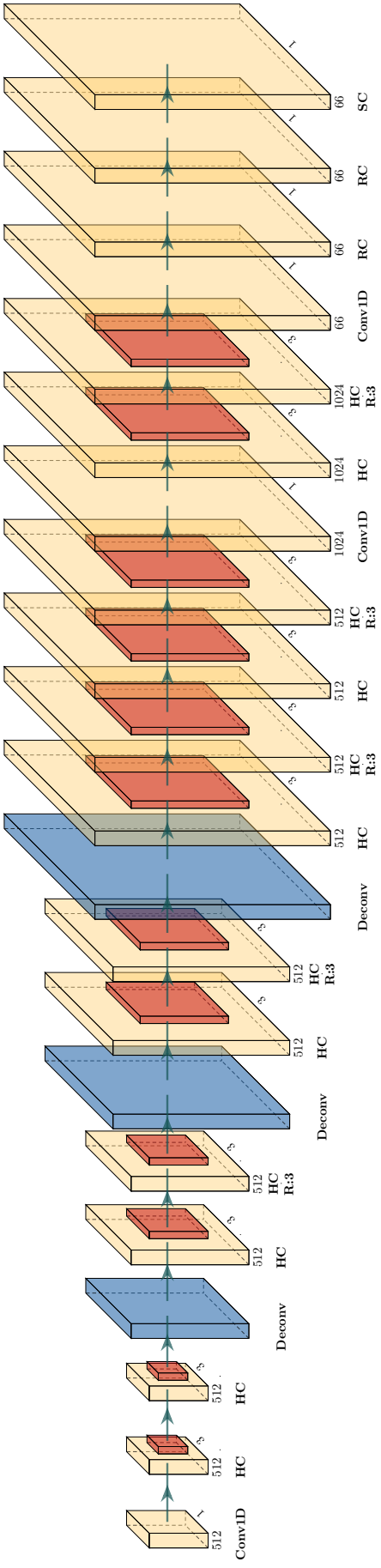

Figura 15 – Topologia do modelo WSRN

Fonte: Autoria Própria

Tabela 6 – Parâmetros utilizados para a extração das características do WORLD *vocoder*

Sampling rate	48000
FFT Size	2048
Período de *frame* para o Text2WORLD (DCC2WORLD)	48
Período de *frame* para o WSRN (DCC2WORLD)	8
F0 Floor	71
F0 Ceil	800
Velocidade	1

Fonte: Autoria Própria

Os experimentos do grupo 1, 2 e 4 foram treinados no computador 2 enquanto os demais experimentos foram executados utilizando o computador 1. As especificações dos computadores utilizados para treinamento estão dispostas na Tabela 3.

Os experimentos do grupo 1 e 2 utilizam a mesma função para treinamento. A parte Text2Mel que converte uma sentença em um espectrograma utiliza uma função de perda que é composta pela função L1, também conhecida como *Least Absolute Deviations* (LAD) e *Least Absolute Errors* (LAE) (GOODFELLOW et al., 2016a), pela entropia cruzada multi-classe (GOODFELLOW et al., 2016a) e pela função de perda da atenção guiada (TACHIBANA et al., 2017). Já a rede de super resolução (SSRN) utiliza uma função de perda que é constituída apenas pelas funções L1 e entropia cruzada multi-classe. Nos experimentos do grupo 3 utiliza-se as mesmas funções de perda, utilizadas no grupo 1 e 2, para o Text2Mel e a SSRN. Entretanto, o *vocoder* WaveRNN utiliza a função de perda Gaussiana (*Gaussian Loss*) (PING et al., 2018), que permite valores negativos.

Nos experimentos do grupo 4, os quais exploram o uso do WORLD *vocoder*, é utilizado para a rede Text2World como função de perda o erro quadrático médio (*Mean Squared Error*) em conjunto com a função de perda da atenção guiada. A escolha dos erros quadráticos médios para a função de perda da rede foi realizada em experimentos preliminares onde observou-se que a atenção convergia de forma mais rápida com a sua utilização. Já para a WSRN utilizou-se como função de perda apenas os erros quadráticos médios.

Nos experimentos do grupo 5 e 6 não utiliza-se atenção guiada, portanto, a função de perda não incluiu o custo da atenção. Além disso, o cálculo da perda não depende de apenas uma saída devido a uma rede similar a SSRN, denominada módulo CBHG (1-D *convolution bank + highway network + bidirectional GRU*), ser treinada em conjunto a parte que converte o texto em espectrograma de Mel, desta forma a perda depende da saída das duas redes.

Como forma de avaliação objetiva para a qualidade do áudio sintetizado, observar-se que a medida MOS é muito cara em recursos humanos. Desta forma, é proposto comparar as saídas desejadas e obtidas utilizando *Root Mean Square Error* (RMSE) (TAMAMORI et al.,

2017b), *Structural Similarity Index Method* (SSIM) (WANG et al., 2004) e a distância *Dynamic Time Warping* (DTW) (MÜLLER, 2007).

A RMSE é uma medida geralmente utilizada para comparar erros de previsão de modelos distintos. A medida mensura os erros e penaliza erros graves, quanto menor a medida mais parecidas são as previsões/imagens. Nesse trabalho, calculou-se um somatório da RMSE de cada *frame* do espectrograma STFT e após calculou-se a media do somatório considerando o total de *frames* do espectrograma (TAMAMORI et al., 2017b).

A SSIM é uma medida utilizada para calcular a similaridade entre duas imagens, ela varia entre 1 e -1, onde o valor 1 indica que a estrutura das imagens são idênticas, 0 aponta que as imagens não possuem similaridade estrutural e valores negativos indicam que a estrutura das imagens estão invertidas (WANG et al., 2004; BROOKS et al., 2008).

Para comparar duas séries temporais pode-se utilizar de uma medida de distância muito simples, como a distância euclidiana. Entretanto, é comum o caso em que duas sequências com aproximadamente as mesmas formas não se alinham no eixo X, por exemplo na área da fala duas pessoas falando a mesma frase, entretanto ambas começam em um tempo diferente e/ou falam em velocidades diferentes. Para encontrar a similaridade entre essas sequências, deve-se deformar o eixo temporal de uma (ou ambas) sequências para obter um melhor alinhamento (KEOGH; PAZZANI, 2001). A *Dynamic Time Warping* (DTW), é uma técnica para alcançar com eficiência essa deformação, após o alinhamento das sequências, é possível calcular a distância entre o espectrograma esperado e predito, a qual nesse trabalho será denominada como distância DTW, quanto menor a distância DTW mais parecidas são as sequências temporais. A DTW foi utilizada para diversas aplicações como na mineração de dados (KEOGH; PAZZANI, 2000), no reconhecimento de gestos (GAVRILA et al., 1995), no processamento de fala (RABINER et al., 1993) e na medicina (CAIANI et al., 1998).

4 RESULTADOS E DISCUSSÕES

4.1 BASE DE ÁUDIO DESENVOLVIDA

A base de fala desenvolvida nesse trabalho tem aproximadamente 10 horas e 28 minutos de fala de um único locutor (o autor do trabalho), gravado com uma taxa de amostragem de 48.000 Hz e um *Bit Depth* de 32-bit ponto flutuante (*flut*). A base contém um total de 3.632 arquivos de áudio no formato Wave. Os arquivos de áudio variam de 0,67 a 50,08 segundos. O número total de palavras da base é 71.358, sendo 13.311 palavras distintas.

Duas análises são propostas sobre os modelos induzidos a partir da base de áudio: uma avaliação subjetiva (descrita na Seção 4.2); e uma avaliação objetiva (descrita na Seção 4.3). Para ambas avaliações, é proposta a síntese de frases extraídas de dois conjuntos foneticamente balanceadas (com 10 frases cada) retiradas do trabalho proposto por Seara (1994). As frases utilizadas são:

1. A inauguração da vila é quarta ou quinta-feira
2. Vote se você tiver o título de eleitor
3. Hoje é fundamental encontrar a razão da existência humana
4. A temperatura é mais amena à noite
5. Em muitas cidades a população está diminuindo.
6. Nunca se deve ficar em cima do morro
7. Para as pessoas estranhas o panorama é desolador
8. É bom te ver colhendo flores menino
9. Eu finjo me banhar num lago ao amanhecer
10. Sua sensibilidade mostrará o caminho
11. A Amazônia é a reserva ecológica do globo
12. O ministério mudou demais com a eleição
13. Novas metas surgem na informática

14. O capital de uma empresa depende de sua produção
15. Se não fosse ela tudo teria sido melhor
16. A principal personagem no filme é uma gueixa
17. Espere seu amigo em casa
18. A juventude tinha que revolucionar a escola
19. A cantora terá quatro meses para ensaiar seu canto
20. Esse tema foi falado no congresso.

4.2 ANÁLISE SUBJETIVA

Devido ao escopo do trabalho, a análise dos resultados em um material escrito é dificultada já que os resultados dos áudios não podem ser transportados de maneira ideal para o texto escrito. Para facilitar essa análise, diversos recursos foram disponibilizados na Internet, conforme descritos na Tabela 7. Isso inclui a base de áudio desenvolvida nesse trabalho, amostras de áudio sintetizadas em cada experimento, os repositórios e *forks* (garfos) do GitHub com as implementações dos modelos, os *forks* são necessários devido às mudanças realizadas nos modelos. Adicionalmente, foram disponibilizados IPython Notebooks[1], hospedados no Google Colab[2], para demonstração dos modelos.

A Tabela 8 apresenta o *ranking* entre os experimentos propostos no trabalho na análise subjetiva, o *ranking* foi realizado utilizando a opinião do autor do trabalho. Para a realização do *ranking* o autor do trabalho analisou a sintetização das 20 frases apresentadas anteriormente na Seção 4.1.

4.3 ANÁLISE OBJETIVA

Os experimentos do grupo 1 replicam o modelo DCTTS, e utilizam o mesmo *vocoder* utilizado pelos autores (RTISI-LA). Entretanto, para o modelo convergir, foi necessário o uso

[1] Site oficial: https://ipython.org/notebook.html
[2] Site oficial: https://colab.research.google.com/

Tabela 7 – Resultados para análise subjetiva.

TTS-Portuguese Corpus	https://github.com/Edresson/TTS-Portuguese-Corpus
Áudios sintetizados	https://edresson.github.io/TTS-Portuguese-Corpus
Implementação do modelo DCTTS	https://github.com/Edresson/TTS-Conv
Fork da implementação do vocoder WaveRNN	https://github.com/Edresson/WaveRNN-Pytorch
Fork da implementação do Tacotron 1	https://github.com/Edresson/tacotron
Fork da implementação do TTS da Mozilla	https://github.com/Edresson/TTS
Demo Colab experimento 1.1	http://bit.do/demo-sintese-DCTTS-text
Demo Colab experimento 3.1	http://bit.do/demo-sintese-DCTTS-WaveRNN
Demo Colab experimento 5.3	http://bit.do/demo-sintese-tacotron1
Demo Colab experimento 6.1	http://bit.do/demo-tts-mozilla-no-transfer
Demo Colab experimento 6.3	http://bit.do/demo-tts-mozilla

Fonte: Autoria Própria

Tabela 8 – *Ranking* dos experimentos na análise subjetiva.

Posição	Experimento
1º	6.3 e 6.4
2º	6.1 e 6.2
3º	1.1 e 1.2
4º	2.1 e 2.2
5º	5.3 e 5.4
6º	3.2
7º	3.1
8º	4.1, 4.2, 5.1 e 5.2

Fonte: Autoria Própria

de *layer normalization* e *dropout* em todas as suas camadas.

Os experimentos do grupo 2 também replicam o modelo DCTTS como descrito nos experimentos do grupo 1, entretanto é utilizado como entrada do modelo as transcrições fonéticas dos textos em vez dos textos originais.

Nos experimentos do grupo 3, explora-se o uso do *vocoder* WaveRNN com o modelo DCTTS em vez do *vocoder* RTISI-LA. Nesses experimentos o modelo DCTTS foi modificado, a rede de super resolução (SSRN), foi alterada para predizer espectrograma de Mel em vez do espectrograma STFT completo.

Nos experimentos do grupo 4 explora-se o uso do WORLD *vocoder* em conjunto com o modelo DCTTS. O modelo DCTTS foi modificado para sintetizar as características do WORLD *vocoder* (SP, AP, F0), em vez do espectrograma STFT. O modelo proposto nesse experimento foi denominada de DCC2WORLD, descrito anteriormente na Seção 3.3.

Os experimentos do grupo 5 e 6 exploram, respectivamente, o uso do modelo Tacotron 1 (WANG et al., 2017b) e do modelo TTS da Mozilla (MOZILLA, 2019). Nesses grupos explora-se ainda o uso de transferência de aprendizagem entre o inglês e o português. Nesses experimentos são demonstradas as vantagens da transferência de aprendizado para a convergência do modelo.

A Tabela 9 apresenta os dados do treinamento para cada modelo dividido em grupos e experimentos, já que vários experimentos de um mesmo grupo utilizam o mesmo modelo e alguns experimentos utilizam modelos treinados em circunstâncias diferentes. Os dados de treinamento presentes na tabela são: número de passos (*steps*) do treinamento, o tempo necessário para o treinamento e a perda (*loss*) no passo final do treinamento.

Tabela 9 – Dados do treinamento dos modelos.

Grupo de experimento	**Passos de treinamento**	**Tempo**	**Perda**
Grupo 1 (Text2Mel/SSRN)	2115k/2019k	4d19h/5d22h	0,5134/0,4537
Grupo 2 (Text2Mel/SSRN)	1734k/2019k	4d20h/5d22h	0,4975/0,4537
Grupo 3 (Text2Mel/SSRN/ WaveRNN)	2653k/2688k/ 340k	5d14h/6d3h/ 3d5h	0.51/0.529/ -4,3556
Grupo 4 (Text2World/WSRN)	3893k/360k	6d2h/1d4h	0.0136/0.0983
Experimento 5.1	197k	3d23h	0.0995
Experimento 5.2	600k	14d17h	0,0949
Experimento 5.3 e 5.4	57k	23h	0,0905
Experimento 6.1 e 6.2	135k	6d2h	0,10172
Experimento 6.3 e 6.4	70k	2d22h	0,10126

Fonte: Autoria Própria

Como pode ser observado na Tabela 9, os passos de treinamento são diferentes entre cada experimento, até quando utilizam o mesmo modelo. Optou-se por essa alternativa devido a diferentes velocidades de convergência dos experimentos devido às modificações propostas. Por exemplo, nos experimentos do grupo 1 e 2, a rede Text2Mel do modelo DCTTS dos experimentos do grupo 2 foi treinada com menos passos, pois com a utilização de fonemas, o modelo convergiu mais rápido, ou seja, obteve uma perda menor em menos passos. Como já discutido anteriormente na Seção 3.3, as funções de perda do modelo nos experimentos do grupos 1 e 2 são idênticas, logo, nesses experimentos pode-se comparar os valores. Entretanto nos demais experimentos as funções de perda são diferentes, logo não se pode fazer comparação acerca de seus valores.

No experimento 5.3, foi utilizado um modelo pré-treinado na língua inglesa, treinado

Tabela 10 – Resultados da avaliação objetiva.

Experimentos	SSIM	RMSE	distância DTW
Experimento 1.1	0,2738	0,1934	54,2142
Experimento 1.2	0,2715	0,1943	54,5058
Experimento 2.1	0,2349	0,1992	58,2975
Experimento 2.2	0,2343	0,1990	57,8169
Experimento 3.1	0,2271	0,2600	97,2225
Experimento 3.2	0,2314	0,2295	86,9489
Experimento 4.1	0,2615	0,2529	443,4806
Experimento 4.2	0,2609	0,2277	76,7852
Experimento 5.1	**0,3651**	0,2010	85,4572
Experimento 5.2	0,3590	0,1931	112,9805
Experimento 5.3	0,3048	0,2263	65,5966
Experimento 5.4	0,2938	0,2181	59,1855
Experimento 6.1	0,2551	0,2118	67,734
Experimento 6.2	0,2525	0,2173	69,3479
Experimento 6.3	0,2508	0,1915	**53,9115**
Experimento 6.4	0,2585	**0,1855**	55,5195

Fonte: Autoria Própria

utilizando o corpus LJ Speech[3], com 200 Mil (200k) passos. Após, o modelo é treinado mais 57 mil passos na base TTS-Portuguese Corpus. O mesmo processo se aplica ao experimento 5.4.

No experimento 6.3 utiliza-se um modelo pré-treinado 185 mil passos no corpus LJ Speech. Após continuou-se o treinamento do modelo na base de fala proposta nesse trabalho. O mesmo se aplica ao experimento 6.4

A Tabela 10 apresenta os resultados da avaliação objetiva para cada experimento proposto. As medidas SSIM, RMSE e distância DTW são utilizadas como métricas de avaliação objetiva. A avaliação objetiva foi realizada comparando-se o espectrograma STFT completo extraído do áudio sintetizado com o espectrograma STFT completo extraído do áudio de referência, considerando-se as mesmas frases para síntese e comparação.

Para facilitar a visualização da diferença entre os áudios sintetizados em cada experimento, optou-se por apresentar os espectrogramas para a pronuncia da frase “A temperatura é mais amena à noite”. Na Figura 16 está disposto o espectrograma STFT completo de referência para a pronuncia da frase. As Figuras 17, 18 , 19, 20, 21 e 22, respectivamente, apresentam os espectrogramas STFT completos da pronuncia da frase dos experimentos dos grupos 1, 2, 3, 4, 5 e 6.

[3]Site oficial: https://keithito.com/LJ-Speech-Dataset/

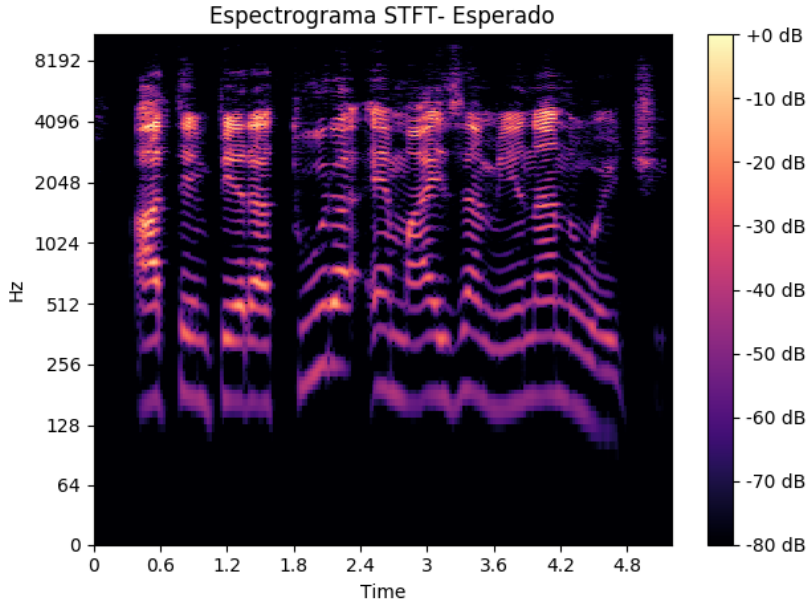

Figura 16 – Espectrograma STFT esperado da pronuncia da frase "A temperatura é mais amena à noite".

Fonte: Autoria Própria

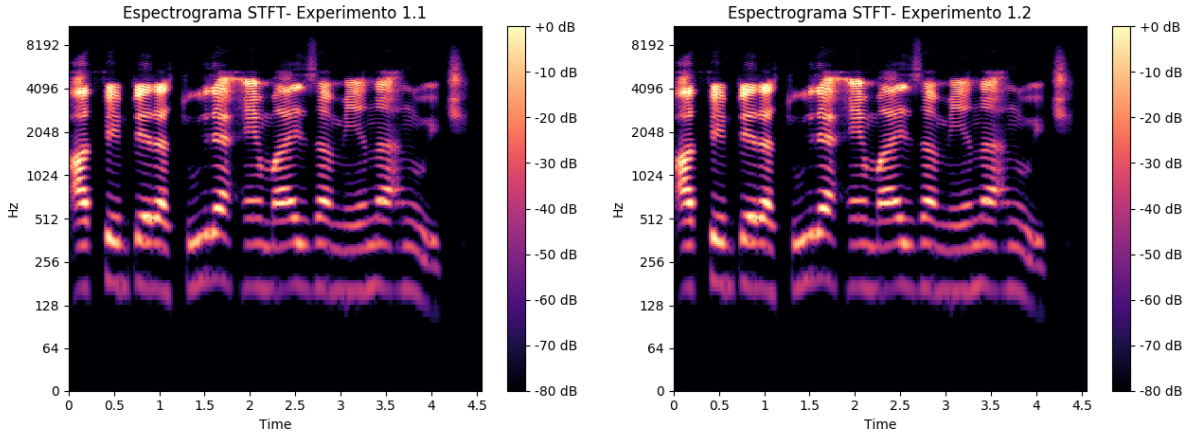

Figura 17 – Espectrogramas STFT da pronuncia da frase "A temperatura é mais amena à noite" dos experimentos do grupo 1.

Fonte: Autoria Própria

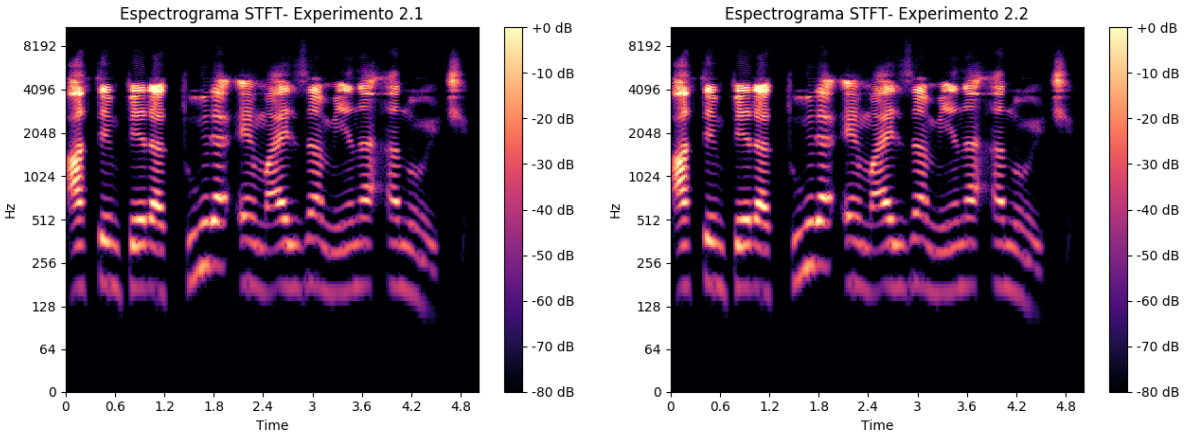

Figura 18 – Espectrogramas STFT da pronuncia da frase "A temperatura é mais amena à noite" dos experimentos do grupo 2.

Fonte: Autoria Própria

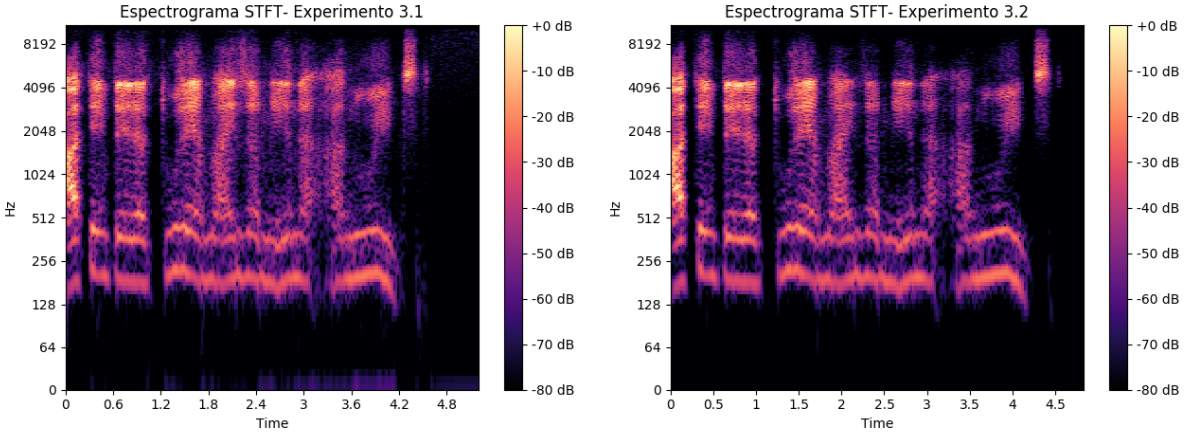

Figura 19 – Espectrogramas STFT da pronuncia da frase "A temperatura é mais amena à noite" dos experimentos do grupo 3.

Fonte: Autoria Própria

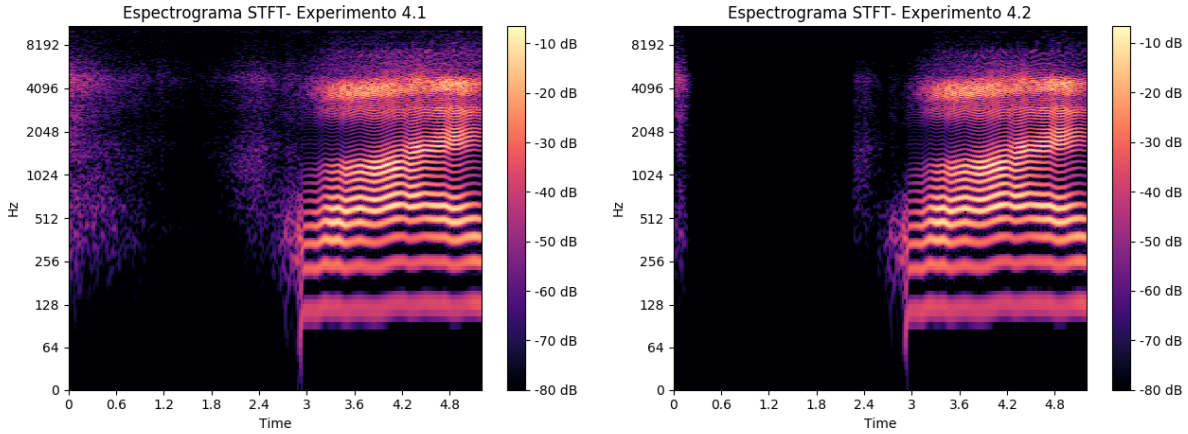

Figura 20 – Espectrogramas STFT da pronuncia da frase "A temperatura é mais amena à noite" dos experimentos do grupo 4.

Fonte: Autoria Própria

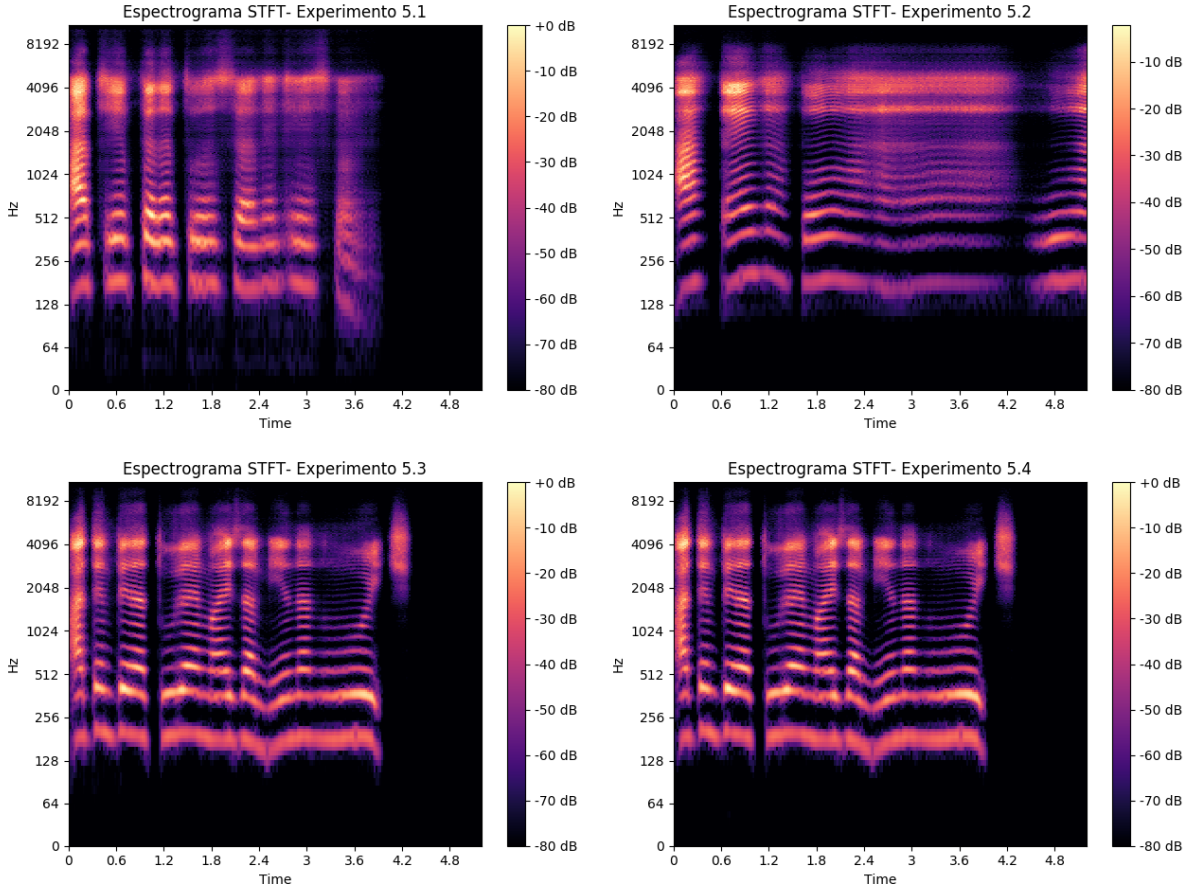

Figura 21 – Espectrogramas STFT da pronuncia da frase "A temperatura é mais amena à noite" dos experimentos do grupo 5.

Fonte: Autoria Própria

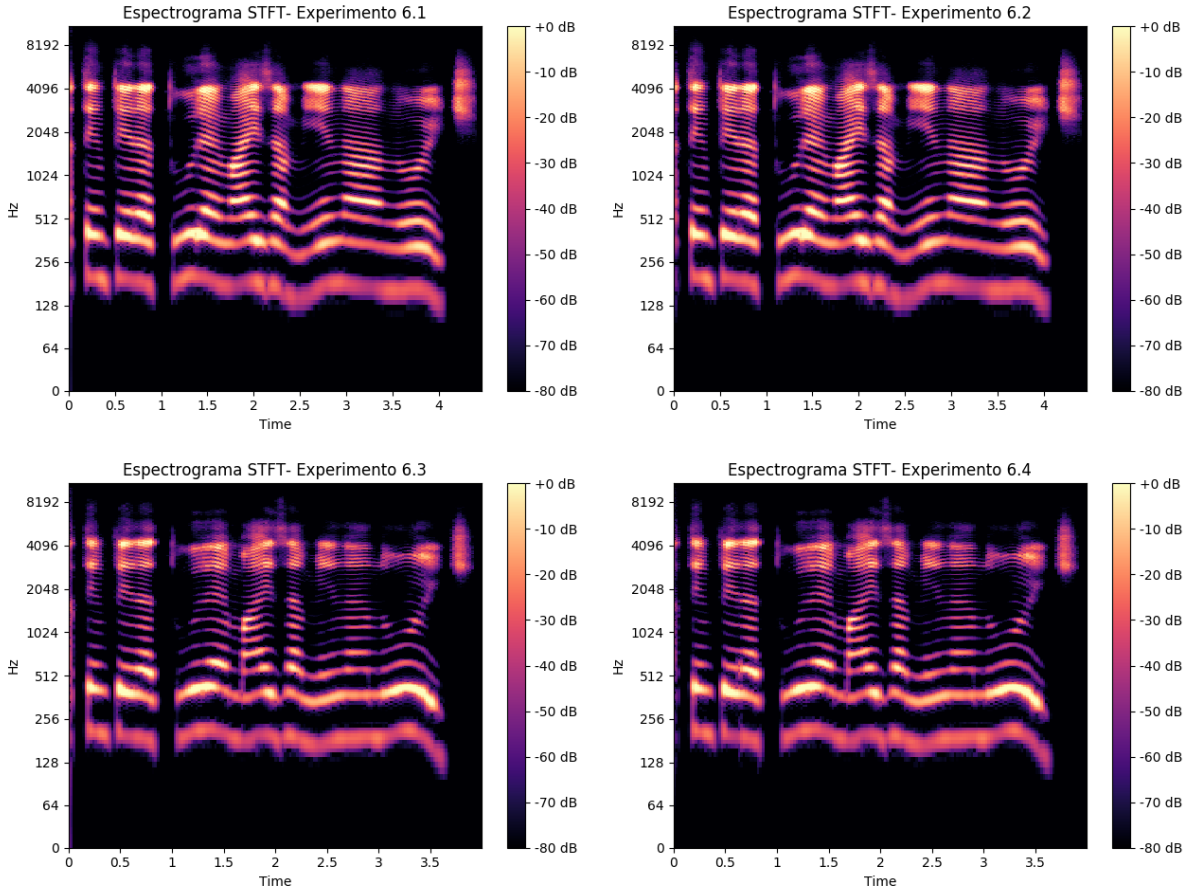

Figura 22 – Espectrogramas STFT da pronuncia da frase "A temperatura é mais amena à noite" dos experimentos do grupo 6.

Fonte: Autoria Própria

4.4 DISCUSSÃO DOS RESULTADOS

Como pode ser observado na Tabela 10, os experimentos do grupo 1 obtiveram bons resultados. O experimento 1.1 na avaliação objetiva utilizando as métricas SSIM e RMSE foi o quarto melhor experimento obtendo uma SSIM de 0,2738 (quanto maior o valor da SSIM mais similar estruturalmente são os áudios) e uma RMSE de 0,1934 (quanto menor a medida mais parecidos são os áudios). Já na distância DTW o experimento 1.1 foi o segundo melhor com uma distância de 54,2142 (quanto menor a distância mais similares são os áudios).

O experimento 1.2 foi o quinto melhor experimento para as métricas SSIM e RMSE, obtendo respectivamente os valores 0,2715 e 0,1943. Para a métrica distância DTW obteve o terceiro menor valor, 54,5058. Este experimento utiliza o mesmo modelo do experimento 1.1 e apenas é aplicada uma Rede Neural Artificial de supressão de ruido (RNNoise) após o áudio ser sintetizado. Contudo, os resultados desse experimento são inferiores ao experimento 1.1 em todas as métricas objetivas, o que pode indicar que a supressão de ruído para esse experimento não é necessária. Entretanto, a diferença é muito pequena e não é perceptível em avaliação subjetiva.

Nos experimentos do grupo 3, obteve-se um resultado inferior aos experimentos do grupo 1 e 2, isso ocorreu pois o *vocoder* neural (WaveRNN) não convergiu. Acredita-se que a principal causa é o tamanho do corpus de treinamento, um *vocoder* neural como o WaveRNN exige um maior número de instâncias de treinamento devido a sua profundidade. Percebe-se ainda que o *vocoder* aprendeu a gerar ruídos juntamente com a voz sintetizada, uma característica indesejável. Ao aplicar a biblioteca de supressão de ruído ao experimento 3.1 (resultando no experimento 3.2), obteve-se uma melhora na qualidade do áudio, passando de uma SSIM 0,2271 para 0.2314, de um RMSE 0,2600 para 0,2295 e de uma distância DTW 97,2225 para 86,9489. Essa melhora também pode ser percebida em avaliação subjetiva.

Os experimentos do grupo 4 não alcançaram bons resultados em avaliação subjetiva (os áudios sintetizados não são compreensíveis), entretanto, obtiveram um bom em resultado em uma das métricas de avaliação objetiva. O experimento 4.1 obteve um SSIM de 0,2615, superando por exemplo os experimentos 2.1, 2.2, 3.1 e 3.2, e obteve um RMSE de 0,2529, superando o experimento 3.1, e uma distância DTW de 443,4806, o pior experimento para essa métrica. O experimento 4.2 obteve respectivamente uma SSIM, RMSE e uma distância DTW de 0,2609, 0,2277 e 76,7852. Percebe-se uma melhora nas métricas de avaliação objetiva, exceto na SSIM após aplicar a biblioteca de supressão de ruídos (experimento 4.2). Acredita-se que os experimentos desse grupo obtiveram uma melhor SSIM que outros experimentos aos quais

os áudios sintetizados são compreensíveis, devido a medida SSIM não ser muito adequada a avaliação de arquivos de áudio. A estrutura do espectrograma é importante, porém como o áudio é uma série temporal, essa pode variar suas características no decorrer do tempo. Por exemplo, uma pessoa ao repetir uma mesma frase irá pronuncia-la em velocidades diferentes. O modelo neural fala em uma velocidade diferente da pronunciada pelo locutor, a velocidade da fala varia em cada experimento/modelo. Desta forma, os melhores resultados na SSIM podem estar diretamente relacionada com a velocidade da pronuncia do modelo e não na compreensão do que está sendo falado.

Os experimentos do grupo 5 exploraram o uso do modelo Tacotron 1. O modelo treinado diretamente na base TTS-Portuguese Corpus (experimento 5.1 e 5.2) não convergiu e gera apenas áudios não compreensíveis. Nos experimentos 5.3 e 5.4 exploram-se o uso da transferência de aprendizagem do inglês para o português, eles alcançaram melhores resultados. Nos experimentos 5.1 e 5.2, percebe-se que apesar da perda ser baixa, 0,0995 e 0,0949 respectivamente, e muito próxima da obtida nos experimentos 5.3 e 5.4 (0,0905), o modelo não convergiu e sintetiza apenas ruídos, enquanto os experimentos 5.3 e 5.4 possuem um bom desempenho. A principal causa da não convergência do modelo nesses experimentos é o mal alinhamento do mecanismo de atenção, como pode ser visualizado nas Figuras 23 e 24, percebe-se que praticamente não há alinhamento entre a frase “A cantora terá quatro meses para ensaiar seu canto” e os *frames* sintetizados. Observa-se que o eixo *x* da figura corresponde aos *frames* do arquivo sintetizado e o eixo *y* corresponde aos caracteres da frase de entrada na ordem em que estão dispostos.

Na Figura 25, pode ser visualizado o alinhamento do mecanismo de atenção para o experimento 5.3. Percebe-se um forte alinhamento entre a frase e o áudio sintetizado, a atenção aplicada a síntese de voz requer um alinhamento praticamente linear já que cada caractere da frase deve gerar alguns *frames* no áudio e a ordem dos caracteres é muito relevante. Observa-se que é sintetizado silêncio depois de toda a frase ser sintetizada. Isso pode ser observado principalmente no canto superior direito da imagem na faixa de tonalidade esverdeada.

No experimento 5.1 obteve-se, respectivamente, uma SSIM, RMSE e uma distância DTW de 0,3651, 0,2010 e 85,4572. Enquanto, no experimento 5.2 o modelo alcançou uma SSIM de 0,3590, um RMSE de 0m1931 e uma distância DTW de 112,9805. Nos resultados dos experimentos 5.1 e 5.2, percebe-se após aplicar a normalização de camadas (experimento 5.2), o RMSE melhorou enquanto a SSIM e a distância DTW tiveram um valor inferior ao experimento 5.1. Após a utilização da transferência de aprendizado, o modelo convergiu e passou a sintetizar áudios compreensíveis. Essa melhora pode ser evidenciada pela mudança da distância DTW passando de 85,4572 e 112,9805, experimento 5.1 e 5.2 respectivamente,

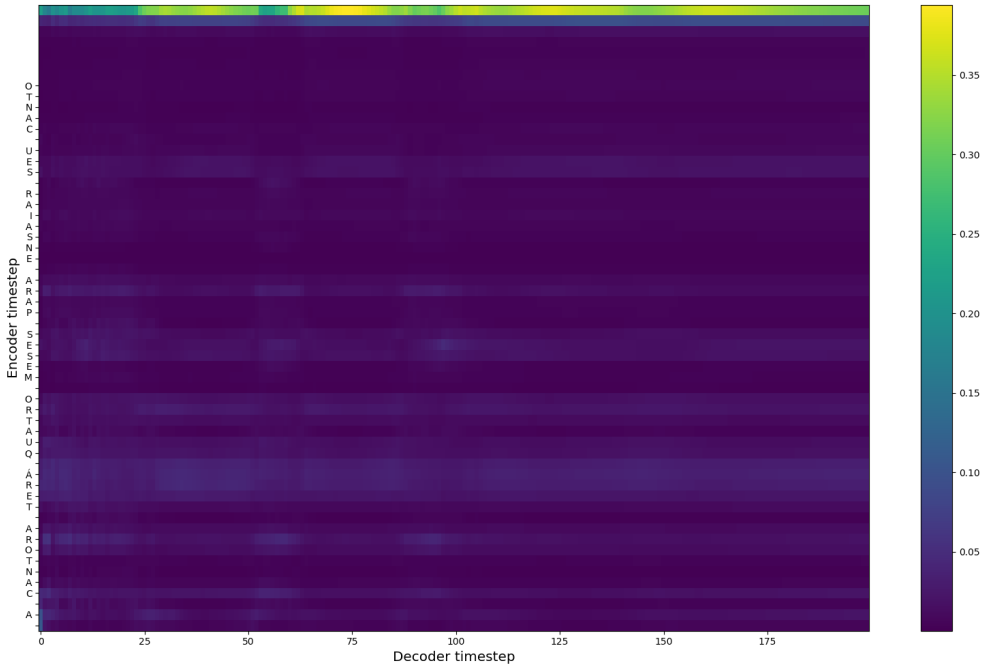

Figura 23 – Alinhamento do mecanismo de atenção para o experimento 5.1

Fonte: Autoria Própria

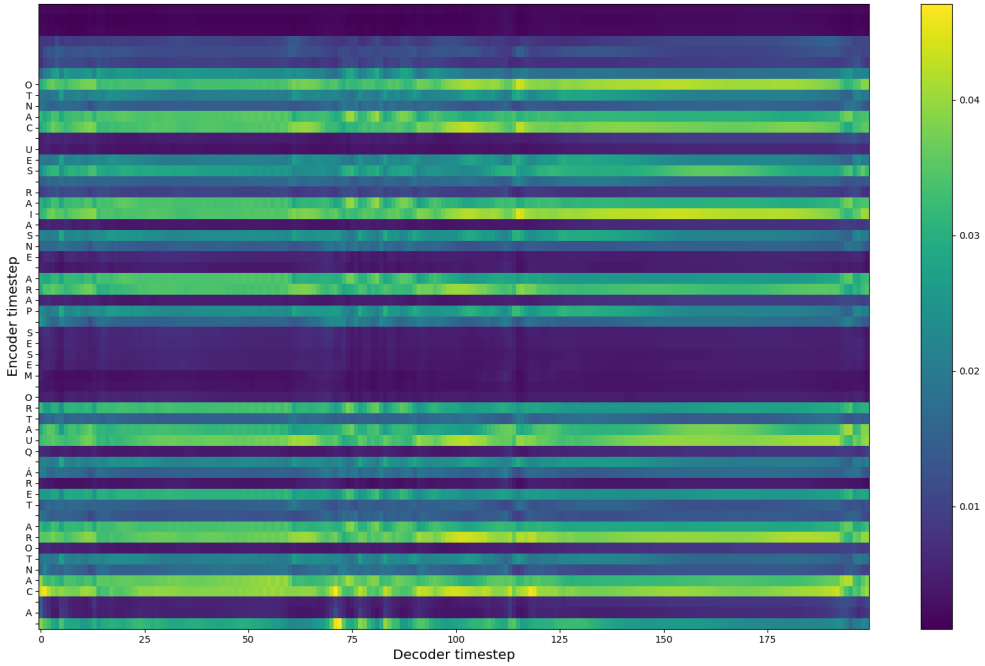

Figura 24 – Alinhamento do mecanismo de atenção para o experimento 5.2

Fonte: Autoria Própria

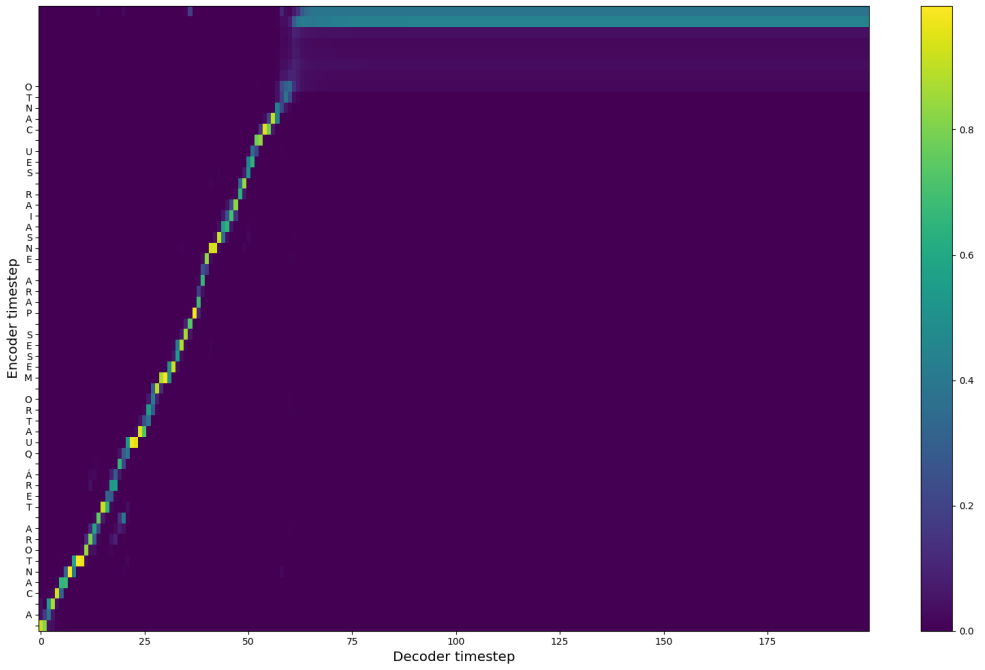

Figura 25 – Alinhamento do mecanismo de atenção para os experimentos 5.3 e 5.4

Fonte: Autoria Própria

para 65,5966 (experimento 5.3) e após aplicar a biblioteca de supressão de ruídos RNNoise para 59,1855 (experimento 5.4). Observa-se que as medidas SSIM e RMSE pioraram após o uso da transferência de aprendizado, o que não era esperado. Entretanto, o baixo desempenho da SSIM para a avaliação de áudios já foi discutida anteriormente e acredita-se que em alguns casos o RMSE pode sofrer com o mesmo problema do SSIM de não considerar a variação da série temporal como realizado pela distância DTW.

Nos experimentos do grupo 6, explora-se o uso do modelo TTS da Mozilla. No experimento 6.1, explora-se o treinamento do modelo diretamente na base TTS-Portuguese Corpus. Nesse experimento, o modelo teve um bom resultado, os áudios são compreensíveis e a pronúncia é razoavelmente boa. Esse experimento alcançou uma SSIM de 0,2551, um RMSE de 0,2118 e uma distância DTW de 67,734. Aplicando a biblioteca de supressão de ruídos ao experimento 6.1 (experimento 6.2), os resultados para as medidas objetivas foram piores, passando para uma SSIM de 0,2525, um RMSE de 0,2173 e uma distância DTW de 69,3479. Acredita-se que o experimento não melhorou após aplicar a biblioteca de supressão de ruído devido a sua utilização não ser necessária. Nos experimentos 6.3 e 6.4, explora-se o uso da transferência de aprendizagem, o modelo convergiu, sintetiza áudios compreensíveis e a pronúncia é a mais natural dentre todos os experimentos. O experimento 6.3 alcançou a melhor distância DTW dentre todos os experimentos. O experimento obteve uma SSIM de 0,2508, um

RMSE de 0,1915 e uma distância DTW de 53,9115. Após aplicar a biblioteca de supressão de ruídos RNNoise ao experimento 6.3 (experimento 6.4) o modelo perdeu nas medidas objetivas SSIM e distância DTW para o experimento 6.3 e alcançou o melhor valor para o RMSE dentre todos os experimentos.

A Tabela 8 apresentou o ranking dos experimentos na análise subjetiva. Dentre todos os experimentos realizados, aqueles que utilizaram o modelo TTS da Mozilla foram os melhores. Em primeiro lugar, estão os experimentos 6.3 e 6.4 que utilizam a transferência de aprendizado e soam mais natural que os outros experimentos. Em segundo lugar estão os experimento 6.1 e 6.2 que também utilizam o modelo TTS da Mozilla, porém, nesses experimentos o modelo é treinado diretamente na base desenvolvida nesse trabalho. Desta forma, não utilizou-se a transferência de aprendizagem. Os experimento 6.1 e 6.2 soam um pouco menos naturais que os experimento 6.3 e 6.4, a transferência de aprendizagem acelerou a convergência do modelo nos experimentos 6.3 e 6.4. A perda com o uso da transferência de aprendizagem também foi menor, passando de 0,10172 para 0,10126 (Tabela 9). Com a utilização da transferência aprendizagem (experimento 6.3 e 6.4), o modelo necessitou de apenas 70 mil passos para convergir, seu treinamento foi realizado em apenas 2 dias e 22 horas enquanto os experimento 6.1 e 6.2 necessitaram de 6 dias e 2 horas de treinamento.

Os experimentos que utilizaram o modelo DCTTS também ficaram bem posicionados no *ranking*. Os experimentos 1.1 e 1.2 ficaram no terceiro lugar. A síntese nesses experimentos é próxima a gerada nos experimentos 6.1 e 6.2, entretanto, a qualidade do áudio gerado é levemente inferior, pois percebe-se uma leve robotização na voz sintetizada. Os experimentos 2.1 e 2.2 ficaram na quarta posição. Nestes experimentos, é notável que com a utilização da transcrição fonética do texto para a entrada da rede, a mesma convergiu mais rápido que o experimento 1.1. Entretanto, o modelo enfrentou alguns problemas de pronuncia como a repetição de palavras. Por exemplo na frase “Novas metas surgem na informática” o modelo repete duas vezes a palavra “metas”. Acredita-se que isso acontece, pois com o uso de fonemas o problema se torna um pouco menos complexo, assim a rede neural consegue aprender *outliers* indesejados da base.

Por sua vez, os experimentos 5.3 e 5.4 ficaram na quinta posição. Esses experimentos exploram o uso da transferência de aprendizagem com o modelo Tacotron 1. A partir dos áudios sintetizados nesses experimentos, é notável a presença de erros de pronunciação das palavras e presença de rouquidão na voz do locutor. Os erros de pronúncia são causados principalmente pela falta de alguns acentos da língua portuguesa no vocabulário do modelo, de 42 símbolos o modelo foi treinado com apenas 32. Tal falta foi necessária devido ao vocabulário do modelo original ser pequeno e não haver espaço para adicionar os acentos, se adicionar novos caracteres

ao vocabulário não é possível carregar os pesos do modelo pré-treinado no idioma inglês. Uma alternativa a esse problema é não carregar os pesos da camada de *embedding* do modelo e deve ser explorada em trabalhos futuros. Após essas mudanças, acredita-se que o modelo possa obter uma melhor pronunciação das palavras sintetizadas.

Os experimentos 3.1 e 3.2 ficaram na sétima e sexta posição, respectivamente. Nesses experimentos, é notável que o *vocoder* WaveRNN sintetiza ruídos juntamente com a fala, após aplicar a biblioteca de supressão de ruídos (experimento 3.2), observa-se uma melhora na qualidade do áudio, porém a voz do locutor soa um pouco rouca. Além disso, os experimentos apresentam problemas de pronuncia principalmente no final das frases.

Os piores experimentos, ficando na oitava posição, foram os experimentos 4.1, 4.2, 5.1 e 5.2. Nesses experimentos, a partir dos áudios sintetizados, não é possível compreender o texto que está sendo falado. Acredita-se que os experimentos do grupo 4 obtiveram um desempenho ruim devido a não convergência do mecanismo de atenção, isso pode ser observado na Figura 26. Além disso o WORLD *vocoder* é mais sensível a variações dos valores que o RTISI-LA/Griffin-Lim, desta forma uma pequena variação nos valores preditos influencia muito o resultado final da síntese. Nos experimentos 5.1 e 5.2, o modelo Tacotron 1 não convergiu ao ser treinado diretamente com a base TTS-Portuguese Corpus (sem o uso da transferência de aprendizagem). Como já discutido anteriormente acredita-se que a principal causa da não convergência do modelo foi o mal alinhamento do mecanismo de atenção.

4.5 CONSIDERAÇÕES FINAIS

Nos experimentos realizados, o modelo que levou aos melhores resultados foi o TTS da Mozilla, isto pôde ser observado tanto na avaliação subjetiva onde ele é o primeiro do *ranking* quanto na objetiva, onde alcançou os melhores resultados para as medidas SSIM e distância DTW. Percebe-se ainda que a transferência de aprendizado possibilitou a convergência do modelo Tacotron, o qual sem o uso de tal técnica não sintetizou áudios compreensíveis, e permitiu uma melhora significativa no desempenho do modelo TTS da Mozilla.

Os modelos Tacotron sem a utilização de transferência de aprendizado e DCC2WORLD que explora a utilização do *vocoder* WORLD não apresentaram bons resultados na base de estudo, o que sugere que tais modelos podem convergir com a utilização de uma base maior.

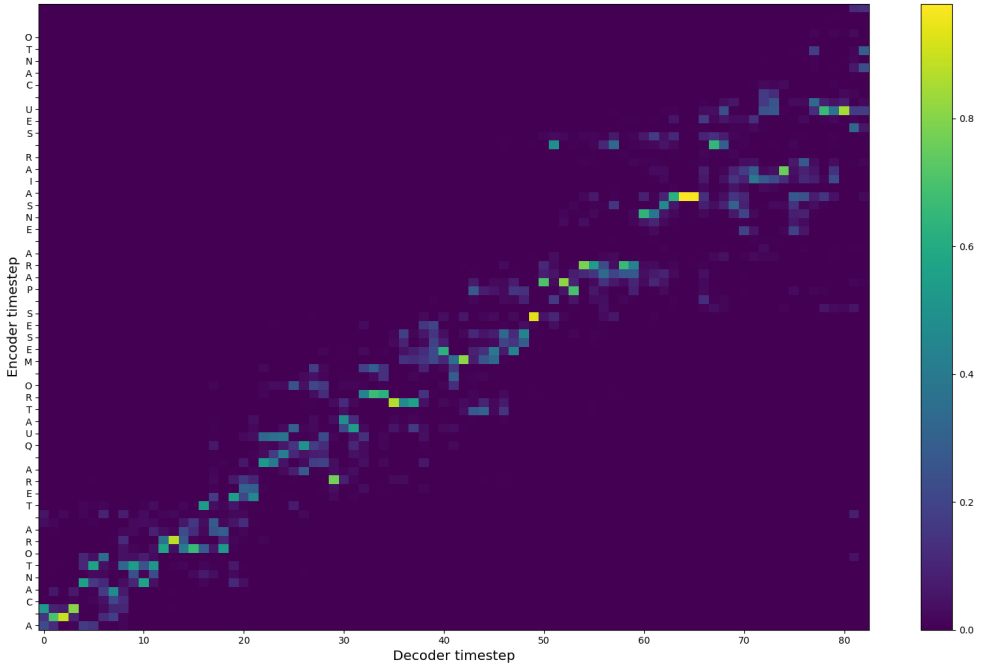

Figura 26 – Alinhamento do mecanismo de atenção no experimento 4.1 para a pronuncia da frase "A cantora terá quatro meses para ensaiar seu canto"

Fonte: Autoria Própria

5 CONCLUSÃO

Neste trabalho explorou-se o estado da arte da síntese de voz para o Português Brasileiro. Todos os objetivos propostos nesse trabalho foram atingidos, desta forma, pretende-se fornecer contribuições para a área de pesquisa da fala. As principais contribuições são:

- Desenvolvimento de uma base de fala para síntese de voz no idioma Português Brasileiro com aproximadamente 10 horas e 28 minutos de áudio e totalmente aberta. A base contribui de forma direta para a área diante da escassez de bases de fala para síntese de voz no idioma Português;
- Implementação (codificação) de um modelo convolucional para síntese de voz baseado no modelo DCTTS e uma comparação entre o uso dos *vocoders* RTISI-LA, WaveRNN e WORLD. Desta forma, o trabalho contribui para a escolha de um *vocoder* ideal dependendo da base a ser utilizada para o treinamento dos modelos.
- Modelos pré-treinados para o Português Brasileiro utilizando a base TTS-Portuguese Corpus. Os modelos providos nesse trabalho podem ser utilizados em vários projetos envolvendo síntese de voz. Por exemplo, projetos relacionados a acessibilidade digital de pessoas com deficiência visual, bem como auxiliar pessoas com deficiência de fala a se comunicar.
- Exploração de transferência de aprendizado, entre dois idiomas distintos, para síntese de voz, demonstrando que com sua utilização os modelos podem convergir mais rapidamente. Adicionalmente, demonstrou-se também que um modelo que não convergiu na base de fala pôde convergir com a utilização de transferência de aprendizado.

5.1 TRABALHOS FUTUROS

É possível, em trabalhos futuros:

- Alterar o vocabulário do modelo Tacotron 1 adicionando todos os acentos da língua

portuguesa, após treinar o modelo utilizando o modelo pré-treinado do idioma inglês, entretanto, não carregar os pesos da camada de *embeddings* do modelo;

- Treinar os modelos TTS Cube (TIBERIU, 2019), DeepVoice 3 (PING et al., 2017) e Tacotron 2 (SHEN et al., 2017; MAMA, 2018) na base desenvolvida nesse trabalho;
- Explorar a utilização dos *vocoders* LPCNet (VALIN; SKOGLUND, 2019), WaveNet (TAMAMORI et al., 2017a), Clarinet (PING et al., 2018), GELP (JUVELA et al., 2019) e WaveGlow (PRENGER et al., 2019);
- Explorar a transferência de aprendizado para os *vocoders* neurais, visando aumentar o desempenho dos modelos neurais, como o WaveRNN;
- Realizar como forma complementar de avaliação subjetiva uma análise com ajuda de avaliadores humanos, para tal pode utilizar-se da medida MOS onde os avaliadores devem avaliar a qualidade dos áudios sintetizados conforme descrito anteriormente na Tabela 1.

REFERÊNCIAS

AMARI, S. et al. **The handbook of brain theory and neural networks**. [S.l.]: MIT press, 2003.

AMODEI, D. et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In: **International Conference on Machine Learning**. [S.l.: s.n.], 2016. p. 173–182.

ANTHIMOPOULOS, M. et al. Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. **IEEE transactions on medical imaging**, IEEE, v. 35, n. 5, p. 1207–1216, 2016.

ARIK, S. O. et al. Deep voice: Real-time neural text-to-speech. **arXiv preprint arXiv:1702.07825**, 2017.

ARIK, S. O. et al. Deep voice 2: Multi-speaker neural text-to-speech. **arXiv preprint arXiv:1705.08947**, 2017.

BA, J. L.; KIROS, J. R.; HINTON, G. E. Layer normalization. **arXiv preprint arXiv:1607.06450**, 2016.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014.

BALDUZZI, D.; GHIFARY, M. Strongly-typed recurrent neural networks. **arXiv preprint arXiv:1602.02218**, 2016.

BENESTY, J.; CHEN, J.; HABETS, E. A. **Speech enhancement in the STFT domain**. [S.l.]: Springer Science & Business Media, 2011.

BERNARD, M. **Simple text to phonemes converter for multiple languages**. 2019. Https://github.com/bootphon/phonemizer.

BEZERRA, E. Introdução à aprendizagem profunda. **Tópicos em Gerenciamento de Dados e Informações. Porto Alegre: Sociedade Brasileira de Computação**, p. 57–86, 2016.

BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: **Proceedings of COMPSTAT'2010**. [S.l.]: Springer, 2010. p. 177–186.

BRADBURY, J. et al. Quasi-recurrent neural networks. **arXiv preprint arXiv:1611.01576**, 2016.

BREDIN, H. TRISTOUNET: TRIPLET LOSS FOR SPEAKER TURN EMBEDDING. n. 1, 2017. Disponível em: <https://arxiv.org/pdf/1609.04301.pdf>.

BROOKS, A. C.; ZHAO, X.; PAPPAS, T. N. Structural similarity quality metrics in a coding context: Exploring the space of realistic distortions. **IEEE Transactions on image processing**, IEEE, v. 17, n. 8, p. 1261–1273, 2008.

CAIANI, E. G. et al. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. In: IEEE. **Computers in Cardiology 1998. Vol. 25 (Cat. No. 98CH36292)**. [S.l.], 1998. p. 73–76.

CARVALHO, A. P. de Leon F. de. **Redes Neurais Artificiais**. 2017. Disponível em: <http://conteudo.icmc.usp.br/pessoas/andre/research/neural/index.htm>.

CASANOVA, E. **A TensorFlow Implementation of DCTTS**. 2019. Https://github.com/Edresson/TTS-Conv.

CHO, K. et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014.

CHOROWSKI, J. K. et al. Attention-based models for speech recognition. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2015. p. 577–585.

CHRIS, L. **Why Cortana Assistant Can Help Microsoft in the Smartphone Market. The Street**. 2014.

CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv preprint arXiv:1412.3555**, 2014.

CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). **arXiv preprint arXiv:1511.07289**, 2015.

DAI, J. et al. R-fcn: Object detection via region-based fully convolutional networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2016. p. 379–387.

DAVIS, S. B.; MERMELSTEIN, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. In: **Readings in speech recognition**. [S.l.]: Elsevier, 1990. p. 65–74.

DECKER, D. M. et al. **Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet**. [S.l.]: Cambridge University Press, 1999.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**. [S.l.], 2009. p. 248–255.

FARFADE, S. S.; SABERIAN, M. J.; LI, L.-J. Multi-view face detection using deep convolutional neural networks. In: ACM. **Proceedings of the 5th ACM on International Conference on Multimedia Retrieval**. [S.l.], 2015. p. 643–650.

FAYEK, H. **Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between**. 2018. Disponível em: <http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>.

FIRAT, O.; CHO, K.; BENGIO, Y. Multi-way, multilingual neural machine translation with a shared attention mechanism. **arXiv preprint arXiv:1601.01073**, 2016.

GAVRILA, D. M.; DAVIS, L. S. et al. Towards 3-d model-based tracking and recognition of human movement: a multi-view approach. In: CITESEER. **International workshop on automatic face-and gesture-recognition**. [S.l.], 1995. p. 272–277.

GEHRING, J. et al. Convolutional sequence to sequence learning. **arXiv preprint arXiv:1705.03122**, 2017.

GIBBON, D.; MERTINS, I.; MOORE, R. K. **Handbook of multimodal and spoken dialogue systems: Resources, terminology and product evaluation**. [S.l.]: Springer Science & Business Media, 2012.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In: **Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics**. [S.l.: s.n.], 2011. p. 315–323.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning (adaptive computation and machine learning series). **Adaptive Computation and Machine Learning series**, p. 800, 2016.

GOODFELLOW, I. et al. **Deep learning**. [S.l.]: MIT press Cambridge, 2016.

GORDILLO, C. D. A. **Reconhecimento de Voz Contınua Combinando os Atributos MFCC e PNCC com Métodos de Robustez SS, WD, MAP e FRN**. Tese (Doutorado) — PUC-Rio, 2013.

GRAVES, A. Supervised sequence labelling. In: **Supervised sequence labelling with recurrent neural networks**. [S.l.]: Springer, 2012. p. 5–13.

GREFF, K.; SRIVASTAVA, R. K.; SCHMIDHUBER, J. Highway and residual networks learn unrolled iterative estimation. **arXiv preprint arXiv:1612.07771**, 2016.

GRIFFIN, D.; LIM, J. Signal estimation from modified short-time fourier transform. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, IEEE, v. 32, n. 2, p. 236–243, 1984.

GRUBER, T. R. **Siri, a Virtual Personal Assistant—Bringing Intelligence to the Interface**. [S.l.]: Jun, 2009.

GUDMUNDSON, M.; ANDERSON, P.-O. Adjacent channel interference in an ofdm system. In: IEEE. **Vehicular Technology Conference, 1996. Mobile Technology for the Human Race., IEEE 46th**. [S.l.], 1996. v. 2, p. 918–922.

GUO, J. BackPropagation Through Time. **Manuscript**, n. 1, p. 1–6, 2013.

HAYKIN, S. S. et al. **Neural networks and learning machines**. [S.l.]: Pearson Upper Saddle River, NJ, USA:, 2009.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2015. p. 1026–1034.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HIJAZI, S.; KUMAR, R.; ROWEN, C. Using convolutional neural networks for image recognition. **Cadence Design Systems Inc.: San Jose, CA, USA**, 2015.

HOLSCHNEIDER, M. et al. A real-time algorithm for signal analysis with the help of the wavelet transform. In: **Wavelets**. [S.l.]: Springer, 1990. p. 286–297.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. **Neural networks**, Elsevier, v. 4, n. 2, p. 251–257, 1991.

HÜBSCHLE, C. B.; SHELDRICK, G. M.; DITTRICH, B. Shelxle: a qt graphical user interface for shelxl. **Journal of applied crystallography**, International Union of Crystallography, v. 44, n. 6, p. 1281–1284, 2011.

JURAFSKY, D.; MARTIN, J. H. **Speech and language processing**. [S.l.]: Pearson London:, 2014.

JUVELA, L. et al. Gelp: Gan-excited liner prediction for speech synthesis from mel-spectrogram. **arXiv preprint arXiv:1904.03976**, 2019.

KALCHBRENNER, N. et al. Efficient neural audio synthesis. **arXiv preprint arXiv:1802.08435**, 2018.

KALCHBRENNER, N. et al. Neural machine translation in linear time. **arXiv preprint arXiv:1610.10099**, 2016.

KEOGH, E. J.; PAZZANI, M. J. Scaling up dynamic time warping for datamining applications. In: ACM. **Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.], 2000. p. 285–289.

KEOGH, E. J.; PAZZANI, M. J. Derivative dynamic time warping. In: SIAM. **Proceedings of the 2001 SIAM international conference on data mining**. [S.l.], 2001. p. 1–11.

KLAMBAUER, G. et al. Self-normalizing neural networks. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2017. p. 972–981.

KOMINEK, J.; BLACK, A. W. The cmu arctic speech databases. In: **Fifth ISCA workshop on speech synthesis**. [S.l.: s.n.], 2004.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1097–1105.

KROGH, A.; HERTZ, J. A. A simple weight decay can improve generalization. In: **Advances in neural information processing systems**. [S.l.: s.n.], 1992. p. 950–957.

LECUN, Y.; BENGIO, Y. The handbook of brain theory and neural networks. **chapter Convolutional Networks for Images, Speech, and Time Series**, p. 255–258, 1998.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. **The handbook of brain theory and neural networks**, v. 3361, n. 10, p. 1995, 1995.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, MIT Press, v. 1, n. 4, p. 541–551, 1989.

LECUN, Y.; KANTER, I.; SOLLA, S. A. Second order properties of error surfaces: Learning time and generalization. In: **Advances in neural information processing systems**. [S.l.: s.n.], 1991. p. 918–924.

MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: **Proc. ICML**. [S.l.: s.n.], 2013. v. 30, n. 1.

MACLEAN, K. **VoxForge**. 2018. Disponível em: <http://www.voxforge.org/home>.

MAMA, R. **DeepMind's Tacotron-2 Tensorflow implementation**. 2018. Https://github.com/Rayhane-mamah/Tacotron-2.

MAZZA, L. O. Aplicacao de redes neurais convolucionais densamente conectadas no processamento digital de imagens para remocao de ruído gaussiano. 2017.

MCFEE, B. et al. librosa: Audio and music signal analysis in python. In: . [S.l.: s.n.], 2015.

MIKOLOV, T. et al. Recurrent neural network based language model. In: **Eleventh Annual Conference of the International Speech Communication Association**. [S.l.: s.n.], 2010.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2013. p. 3111–3119.

MINSKY, M.; SELFRIDGE, O. G. **Learning in random nets**. [S.l.]: MIT Lincoln Laboratory, 1960.

MOHAMED, A.-r. **Deep neural network acoustic models for asr**. Tese (Doutorado), 2014.

MORISE, M.; YOKOMORI, F.; OZAWA, K. World: a vocoder-based high-quality speech synthesis system for real-time applications. **IEICE TRANSACTIONS on Information and Systems**, The Institute of Electronics, Information and Communication Engineers, v. 99, n. 7, p. 1877–1884, 2016.

MOZILLA. **Deep learning for Text to Speech**. 2019. Https://github.com/mozilla/TTS.

MÜLLER, M. Dynamic time warping. **Information retrieval for music and motion**, Springer, p. 69–84, 2007.

MURTOZA, S. et al. Phonetically balanced bangla speech corpus. In: **Proc. Conference on Human Language Technology for Development 2011**. [S.l.: s.n.], 2011. p. 87–93.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th international conference on machine learning (ICML-10)**. [S.l.: s.n.], 2010. p. 807–814.

NETO, N. et al. Spoltech and ogi-22 baseline systems for speech recognition in brazilian portuguese. In: SPRINGER. **International Conference on Computational Processing of the Portuguese Language**. [S.l.], 2008. p. 256–259.

NIELSEN, M. A. **Neural networks and deep learning**. [S.l.]: Determination Press, 2015.

OLAH, C. Understanding lstm networks. **GITHUB blog, posted on August**, v. 27, p. 2015, 2015.

OORD, A. V. D. et al. Wavenet: A generative model for raw audio. **arXiv preprint arXiv:1609.03499**, 2016.

OORD, A. v. d.; KALCHBRENNER, N.; KAVUKCUOGLU, K. Pixel recurrent neural networks. **arXiv preprint arXiv:1601.06759**, 2016.

PARK, K. **A TensorFlow Implementation of DC-TTS**. 2018. Https://github.com/kyubyong/dc_tts.

PARK, K. **A TensorFlow Implementation of Tacotron: A Fully End-to-End Text-To-Speech Synthesis Model**. 2018. Https://github.com/kyubyong/tacotron.

PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. In: **International Conference on Machine Learning**. [S.l.: s.n.], 2013. p. 1310–1318.

PENHA, D. de P.; CASTRO, A. R. G. Convolutional neural network applied to the identification of residential equipment in non-intrusive load monitoring systems. In: **3rd International Conference on Artificial Intelligence and Applications**. [S.l.: s.n.], 2017. p. 11–21.

PING, W.; PENG, K.; CHEN, J. Clarinet: Parallel wave generation in end-to-end text-to-speech. **arXiv preprint arXiv:1807.07281**, 2018.

PING, W. et al. Deep voice 3: 2000-speaker neural text-to-speech. **arXiv preprint arXiv:1710.07654**, 2017.

PONTI, M. A.; COSTA, G. B. P. da. Como funciona o deep learning. 2017.

PRENGER, R.; VALLE, R.; CATANZARO, B. Waveglow: A flow-based generative network for speech synthesis. In: IEEE. **ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2019. p. 3617–3621.

PURINGTON, A. et al. Alexa is my new bff: social roles, user satisfaction, and personification of the amazon echo. In: ACM. **Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems**. [S.l.], 2017. p. 2853–2859.

QUINTANILHA, I. M. **End-to-End Speech Recognition Applied to Brazilian Portuguese Using Deep Learning**. Tese (Doutorado) — MSc dissertation, PEE/COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, 2017.

RABINER, L. R.; JUANG, B.-H.; RUTLEDGE, J. C. **Fundamentals of speech recognition**. [S.l.]: PTR Prentice Hall Englewood Cliffs, 1993.

RAO, K. et al. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In: IEEE. **Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on**. [S.l.], 2015. p. 4225–4229.

RAUBER, T. W. Redes neurais artificiais. **Universidade Federal do Espírito Santo**, 2005.

RAZAVIAN, A. S. et al. Cnn features off-the-shelf: an astounding baseline for recognition. In: IEEE. **Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on**. [S.l.], 2014. p. 512–519.

RIBANI, M. et al. Validation for chromatographic and electrophoretic methods. **Quimica Nova**, SciELO Brasil, v. 27, n. 5, p. 771–780, 2004.

RIBEIRO, F. et al. Crowdmos: An approach for crowdsourcing mean opinion score studies. In: IEEE. **Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on**. [S.l.], 2011. p. 2416–2419.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

RUSSAKOVSKY, O. et al. Imagenet large scale visual recognition challenge. **International Journal of Computer Vision**, Springer, v. 115, n. 3, p. 211–252, 2015.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Malaysia; Pearson Education Limited,, 2016.

SEARA, I. C. Estudo estatístico dos fonemas do português falado na capital de santa catarina para elaboração de frases foneticamente balanceadas. 1994.

SERRANI, V. M. Ambiente web de suporte à transcrição fonética automática de lemas em verbetes de dicionários do português do brasil. Universidade Estadual Paulista (UNESP), 2015.

SEVILLA-LARA, L. et al. Optical flow with semantic segmentation and localized layers. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2016. p. 3889–3898.

SHANG, W. et al. Understanding and improving convolutional neural networks via concatenated rectified linear units. In: **International Conference on Machine Learning**. [S.l.: s.n.], 2016. p. 2217–2225.

SHEN, J. et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. **arXiv preprint arXiv:1712.05884**, 2017.

SILVA, T. C. **Fonética e fonologia do português: roteiro de estudos e guia de exercícios**. [S.l.]: Contexto, 1999.

SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. Training very deep networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2015. p. 2377–2385.

SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. Training very deep networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2015. p. 2377–2385.

SUMMERFIELD, M. **Rapid GUI programming with Python and Qt: the definitive guide to PyQt programming**. [S.l.]: Pearson Education, 2007.

TACHIBANA, H.; UENOYAMA, K.; AIHARA, S. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. **arXiv preprint arXiv:1710.08969**, 2017.

TAMAMORI, A. et al. Speaker-dependent wavenet vocoder. In: **Proceedings of Interspeech**. [S.l.: s.n.], 2017. p. 1118–1122.

TAMAMORI, A. et al. Speaker-dependent wavenet vocoder. In: **Proc. Interspeech**. [S.l.: s.n.], 2017. v. 2017, p. 1118–1122.

TAYLOR, P. **Text-to-speech synthesis**. [S.l.]: Cambridge university press, 2009.

TIBERIU, B. **End-2-end speech synthesis with recurrent neural networks**. 2019. Https://github.com/tiberiu44/TTS-Cube.

VALIN, J.-M. A hybrid dsp/deep learning approach to real-time full-band speech enhancement. **arXiv preprint arXiv:1709.08243**, 2017.

VALIN, J.-M.; SKOGLUND, J. Lpcnet: Improving neural speech synthesis through linear prediction. In: IEEE. **ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2019. p. 5891–5895.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: **Proceedings of the XXIX Conference on Graphics, Patterns and Images**. [S.l.: s.n.], 2016. p. 1–4.

VASWANI, A. et al. Attention is all you need. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2017. p. 5998–6008.

WANG, G. **Fatcord's Alternative WaveRNN**. 2018. Https://github.com/G-Wang/WaveRNN-Pytorch.

WANG, P. et al. Understanding convolution for semantic segmentation. **arXiv preprint arXiv:1702.08502**, 2017.

WANG, Y. et al. Tacotron: A fully end-to-end text-to-speech synthesis model. **arXiv preprint arXiv:1703.10135**, 2017.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. **IEEE transactions on image processing**, IEEE, v. 13, n. 4, p. 600–612, 2004.

YANG, Z. et al. Improved variational autoencoders for text modeling using dilated convolutions. **arXiv preprint arXiv:1702.08139**, 2017.

YU, D.; DENG, L. **Automatic Speech Recognition: A Deep Learning Approach**. Springer London, 2014. (Signals and Communication Technology). ISBN 9781447157793. Disponível em: <https://books.google.com.br/books?id=rUBTBQAAQBAJ>.

YU, D.; DENG, L. **AUTOMATIC SPEECH RECOGNITION.** [S.l.]: Springer, 2016.

YU, D. et al. Feature learning in deep neural networks-studies on speech recognition tasks. **arXiv preprint arXiv:1301.3605**, 2013.

YU, F.; KOLTUN, V. Multi-scale context aggregation by dilated convolutions. **arXiv preprint arXiv:1511.07122**, 2015.

ZEN, H.; SAK, H. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. In: IEEE. **Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on**. [S.l.], 2015. p. 4470–4474.

ZHU, X.; BEAUREGARD, G. T.; WYSE, L. L. Real-time signal estimation from modified short-time fourier transform magnitude spectra. **IEEE Transactions on Audio, Speech, and Language Processing**, IEEE, v. 15, n. 5, p. 1645–1653, 2007.