UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

Reconhecimento de Fala Utilizando Aprendizado Hierárquico Multitarefa

Trabalho de Conclusão de Curso submetido ao curso de Engenharia Eletrônica da Universidade Federal de Santa Catarina como requisito para aprovação da disciplina EEL7806 Projeto Final TCC.

André Lucas Schlichting

Orientador: Rui Seara Júnior
Coorientador: Richard Demo Souza

Florianópolis, 12 de dezembro de 2024.

ANDRÉ LUCAS SCHLICHTING

RECONHECIMENTO DE FALA UTILIZANDO APRENDIZADO HIERÁRQUICO MULTITAREFA

Trabalho de Conclusão de Curso submetido ao curso de Engenharia Eletrônica da Universidade Federal de Santa Catarina como requisito para aprovação da disciplina EEL7806 Projeto Final TCC.
Orientador: Rui Seara Júnior.
Coorientador: Richard Demo Souza.

FLORIANÓPOLIS
2024

Schlichting, André Lucas
Reconhecimento de Fala Utilizando Aprendizado Hierárquico Multitarefa / André Lucas Schlichting ; orientador, Rui Seara Júnior, coorientador, Richard Demo Souza, 2024.
80 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2024.

Inclui referências.

1. Engenharia Eletrônica. 2. Reconhecimento de Fala. 3. Aprendizado Hierárquico. 4. Aprendizado Multitarefa. I. Seara Júnior, Rui. II. Souza, Richard Demo. III. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. IV. Título.

André Lucas Schlichting

RECONHECIMENTO DE FALA UTILIZANDO APRENDIZADO HIERÁRQUICO MULTITAREFA

Este Trabalho de Conclusão de Curso foi julgado adequado no contexto da disciplina EEL7806 Projeto Final TCC, e aprovado em sua forma final pelo Departamento de Engenharia Elétrica e Eletrônica da Universidade Federal de Santa Catarina.

Florianópolis, 12 de dezembro de 2024.

Prof[a]. Daniela Ota Hisayasu Suzuki, Dra.

Coordenadora do Curso

Rui Seara Júnior, Me.

Orientador

Banca examinadora:

Prof. Richard Demo Souza, Dr.

Universidade Federal de Santa Catarina - UFSC

Coorientador

Prof. Eduardo Luiz Ortiz Batista, Dr.

Universidade Federal de Santa Catarina - UFSC

Prof. Mário de Noronha Neto, Dr.

Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina - IFSC

Agradecimentos

Desejo expressar meu reconhecimento a todos que, de uma maneira ou outra, colaboraram na realização deste trabalho, em especial aos meus pais e familiares, pelo apoio incondicional e incentivo constante durante toda esta jornada; aos amigos que fiz durante a graduação, pelos momentos compartilhados de estudo e descontração; e aos colegas do Laboratório de Circuitos e Processamento de Sinais (LINSE), pela parceria, troca de conhecimentos e ambiente colaborativo que proporcionaram.

Não se preocupe apenas com o repertório, mas como o novo repertório nos leva a melhores questões. (Stuart Firenstein)

RESUMO

Neste trabalho, foi investigada a incorporação de informações fonéticas no processo de treinamento de um sistema de reconhecimento automático de fala (ASR) utilizando aprendizado hierárquico multitarefa, com foco em um encoder do tipo Zipformer. A partir de um modelo baseline composto por este encoder e um decoder RNN-T (*Recurrent Neural Network Transducer*) treinado conjuntamente com um decoder CTC (*Connectionist Temporal Classification*), foi proposta uma arquitetura estendida que integra informações fonéticas canônicas em camadas intermediárias do Zipformer através de um decodificador CTC adicional. Experimentos em diversos conjuntos de teste revelaram uma redução consistente na taxa de erro de palavras (*Word Error Rate* - WER) quando as informações fonéticas foram incorporadas na segunda camada do encoder. A incorporação em camadas superiores ou na primeira resultou, em geral, no aumento da taxa de erro. Os resultados abrem caminho para explorar outras formas de representações fonéticas auxiliares em tarefas que podem beneficiar o treinamento de sistemas ASR com uma arquitetura Zipformer de encoder.

Palavras-chave: Reconhecimento Automático de Fala. Aprendizado Hierárquico. Aprendizado Multitarefa.

ABSTRACT

In this work, we investigated the incorporation of phonetic information in the training process of an automatic speech recognition (ASR) system using hierarchical multitask learning, focusing on a Zipformer-type encoder. Starting from a baseline model composed of this encoder and an RNN-T (Recurrent Neural Network Transducer) decoder jointly trained with a CTC (Connectionist Temporal Classification) decoder, we proposed an extended architecture that integrates canonical phonetic information into intermediate layers of the Zipformer through an additional CTC decoder. Experiments on various test sets revealed a consistent reduction in Word Error Rate (WER) when phonetic information was incorporated into the second layer of the encoder. Incorporation into higher or the first layers generally resulted in increased error rates. The results pave the way for exploring other forms of auxiliary phonetic representations in tasks that can benefit the training of ASR systems with a Zipformer encoder architecture.

Keywords: Automatic Speech Recognition. Hierarchical Learning. Multitask Learning.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Símbolo	Definição
$\mathbf{X}$	Sinal de áudio parametrizado.
$\mathbf{Y}$	Sequência de símbolos da transcrição.
$\mathbf{H}$	Representação intermediária após o *encoder*.
D	Dimensão dos parâmetros dos frames de $\mathbf{X}$.
T	Dimensão temporal de $\mathbf{H}$.
T'	Dimensão temporal de $\mathbf{X}$.
L	Número de símbolos em $\mathbf{Y}$.
M	Número de símbolos em $\mathbf{Y}$ na sequência fonética.
$\boldsymbol{\Sigma}$	Conjunto de todos os símbolos que podem estar contidos em $\mathbf{Y}$.
$\langle b \rangle$	Símbolo especial usado para o alinhamento.
$\boldsymbol{\Sigma}$	Conjunto de todos os símbolos válidos que podem estar contidos em $\mathbf{Y}$.
$\boldsymbol{\Sigma}'$	Conjunto estendido $\boldsymbol{\Sigma} \cup \langle b \rangle$
L'	Número de símbolos de $\boldsymbol{\Sigma}$.
A	Alinhamento entre $\mathbf{X}$ e $\mathbf{Y}$.

Símbolo	Definição
$\mathbf{A}$	Conjunto de todos os alinhamentos A.
$\hat{\mathbf{Y}}$	Hipótese de $\mathbf{Y}$.

Lista de Siglas

Sigla	Definição
ASR	*Automatic Speech Recognition.*
E2E	*End-to-end.*
RNN	*Recurrent Neural Network.*
RNN-T	*Recurrent Neural Network - Transducer.*
CTC	*Connectionist Temporal Classification.*
HMM	*Hidden Markov Model.*
GMM	*Gaussian Mixture Model.*
SGD	*Stochastic Gradient Descent.*
RPE	*Relative Positional Encoding.*
GLU	*Gated Linear Unit.*
MHA	*Multi-Head Attention.*
MHA(SW)	*Multi-Head Attention (Shared Weights).*
WER	*Word Error Rate.*
GPU	*Graphic Processor Unit*
VRAM	*Video Random Access Memory.*

Sumário

CAPÍTULO 1

Introdução

O reconhecimento automático de fala (*Automatic Speech Recognition* - ASR) é crucial na computação moderna, enfrentando desafios e com grande potencial de avanço. Posicionado entre o processamento de sinais, aprendizado de máquina e linguística computacional, facilita a comunicação homem-máquina pela transcrição precisa da fala.

Nos últimos anos, a evolução neste campo tem sido notável, especialmente com o surgimento dos sistemas fim a fim (*End-to-End* - E2E) [1]. Esta nova abordagem, impulsionada pelos avanços em aprendizado profundo, tem-se mostrado particularmente eficaz em superar as limitações inerentes às arquiteturas tradicionais baseadas em modelos estatísticos híbridos. Diferentemente dos sistemas convencionais, que necessitam de múltiplos componentes independentes, os sistemas E2E oferecem uma solução mais integrada e elegante para a complexa tarefa de converter sinais acústicos em texto.

As aplicações de sistemas ASR são vastas, abrangendo desde assistentes virtuais em *smartphones* e dispositivos domésticos inteligentes [2] até sistemas avançados de transcrição de áudio e comandos por voz em veículos e equipamentos industriais. No setor automotivo, por exemplo, permitem o controle por voz de sistemas de navegação e entretenimento,

Figura 1.1: Diagrama de um Sistema ASR que converte áudio **X** em texto **Y**

Fonte: do Autor.

aumentando a segurança do motorista [3]. Na área médica, facilitam a documentação clínica, permitindo a profissionais de saúde ditarem relatórios com maior eficiência [4]. Além disso, o ASR é fundamental para a acessibilidade, auxiliando pessoas com deficiências através da geração de legendas automáticas e permitindo que indivíduos com mobilidade reduzida controlem dispositivos por comandos de voz. Mais recentemente, o ASR tem se tornado um componente de interesse para a integração de conteúdo de áudio em modelos de linguagem de grande escala (*Large Language Models* - LLMs), permitindo a análise e geração de respostas contextuais baseadas em conteúdo falado [5]. O constante aprimoramento dos sistemas ASR, portanto, tem um impacto direto e significativo em diversos aspectos da sociedade, ampliando as possibilidades de interação humana com sistemas de inteligência artificial.

Um sistema ASR tem como objetivo encontrar, a partir de um sinal de áudio parametrizado $\mathbf{X}$, a sequência de símbolos $\mathbf{Y}$ que representa a transcrição do conteúdo falado no áudio. Um diagrama de alto nível desse sistema é mostrado na Figura 1.1. O sinal de entrada $\mathbf{X}$ representa o áudio parametrizado em atributos acústicos como coeficientes cepstrais [6]. Os símbolos de saída podem ser, por exemplo, caracteres ou sílabas da língua. Os modelos acústicos, sejam os E2E ou tradicionais, são ferramentas que fornecem a distribuição de probabilidade condicional a posteriori $P(\mathbf{Y}|\mathbf{X})$, ou seja, observam os atributos de entrada do sistema e fornecem a sequência de saída mais provável através do modelo treinado.

Os modelos tradicionais de ASR, na Figura 1.2, caracterizam-se por uma abordagem modular que integra componentes distintos de conhecimento de domínio, onde cada módulo é responsável por modelar aspectos específicos do processo de conversão de fala em texto. Esses módulos são o **modelo acústico**, o **modelo de linguagem** e o **lé-**

xico [7]. O modelo acústico tradicional é tipicamente implementado com Modelos Ocultos de Markov (*Hidden Markov Model* - HMM) e Mistura de Gaussianas (*Gaussian Mixture Models* - GMM) e é capaz de mapear características acústicas do sinal de entrada em unidades fonéticas [8]. O modelo de linguagem, por sua vez, incorpora conhecimentos linguísticos, na forma de n-gramas para estimar a probabilidade de sequências de palavras ($P(w_i|w_{i-1}, ..., w_0)$) [9] onde um n-grama é uma subsequência de n palavras em um dado texto. No contexto do processamento de linguagem natural, w_i representa a i-ésima palavra na sequência, sendo w_i a palavra atual e $w_{i-1}, ..., w_0$ as palavras anteriores. O léxico atua como uma ponte entre os domínios acústico e linguístico, definindo o conjunto de palavras conhecidas do sistema com todas as suas respectivas variantes fonéticas. Esta estrutura tripartite, embora eficaz, requer um extenso conhecimento de domínio para sua implementação e otimização. A necessidade de treinamento individual de cada componente, a anotação manual de pronúncias no léxico e a integração cuidadosa desses módulos impõem desafios, especialmente em termos de escalabilidade e adaptabilidade a novos domínios. Assim, esta abordagem compartimentalizada pode limitar a capacidade do sistema em capturar dependências complexas do sinal de fala, como variações de sotaque, variações intra e interlocutores, e diferentes cenários ambientais, motivando assim a busca por paradigmas mais integrados e modernos como os sistemas E2E.

Em um sistema de reconhecimento automático de fala baseado em modelos ocultos de Markov e modelos de misturas Gaussianas, o processo de transcrição do sinal de fala em texto ocorre da seguinte maneira: o sinal de fala é processado pelo modelo acústico, que é composto por HMMs. Cada HMM representa um conjunto de fonemas da língua. Os estados dos HMMs modelam a evolução temporal dos fonemas e as transições entre esses estados, ou transições entre fonemas, têm suas probabilidades de emissão modeladas por parâmetros dos GMMs, médias e variâncias. O decodificador então usa o algoritmo de Viterbi [10] para encontrar a sequência mais provável de estados dos HMMs, dado o sinal de entrada, sendo essa busca guiada pelo léxico, que mapeia as palavras em sequências de fonemas, e pelo modelo de linguagem, que fornece as probabilidades de transição entre palavras. Por fim, a sequência de palavras mais provável é fornecida como a transcrição final

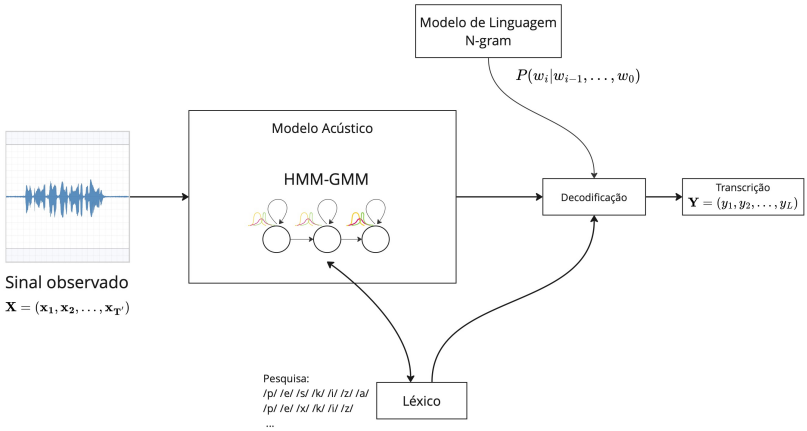

Figura 1.2: Diagrama de um Sistema ASR tradicional baseado em HMMs e GMMs.

Fonte: do Autor.

do sinal de fala. Através desse processo, o sistema é capaz de encontrar a sequência de palavras mais provável dado o sinal de fala, realizando a transcrição em texto.

Os sistemas ASR E2E se destacam por sua capacidade de mapear diretamente os sinais acústicos em sequências textuais, sem a necessidade de utilizar os módulos individuais das arquiteturas tradicionais. Essa abordagem simplificada e orientada por dados tem demonstrado notável eficácia [11]. A combinação de técnicas avançadas de aprendizado de máquina e a natureza fim a fim desses sistemas permitem um treinamento eficiente e uma adaptação mais flexível a diferentes domínios e condições acústicas, tornando-os uma escolha atraente para o desenvolvimento de tecnologias de ASR modernas e de alto desempenho.

De central na revolução metodológica trazida por modelos E2E encontra-se a estrutura *encoder-decoder* ilustrada na Figura 1.3, um *framework* que tem se mostrado particularmente adequado para tarefas de processamento de sequências [12] [13] em sistemas de aprendizado de máquina. Esta estrutura, ao empregar técnicas avançadas de aprendizado profundo, consegue capturar de forma eficaz as complexidades inerentes à fala humana, incluindo variabilidades acústicas e linguís-

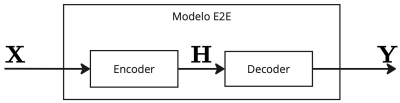

Figura 1.3: Diagrama de um Sistema ASR E2E.
Fonte: do Autor.

ticas anteriormente tratadas individualmente pelos modelos acústico e de linguagem, e também outras formas de variabilidade, como sotaques e estilos de fala tratadas anteriormente pelo léxico. O *encoder* e *decoder* são sistemas implementados com o uso de Redes Neurais onde o *encoder* codifica o sinal de entrada para uma representação de mais alto nível e o decoder processa esta representação transformando-a na forma da saída projetada.

Embora a estrutura *encoder-decoder* tenha alcançado sucesso notável em sistemas ASR E2E, ainda há espaço para aprimoramentos, sendo preenchidos por novas arquiteturas ou pelo sugerimento de melhorias em arquiteturas existentes [14] [15] [16] [17]. No âmbito do *encoder*, por exemplo, as redes neurais convolucionais se destacam por sua eficácia na captura de padrões locais e na promoção de invariância temporal. Essa propriedade das redes convolucionais tem impulsionado avanços significativos no desempenho de sistemas ASR, como evidenciado pela arquitetura *Conformer* [18]. Tal arquitetura inovou ao incorporar blocos convolucionais nas camadas do encoder *Transformer* [19], combinando os benefícios de ambas as técnicas e estabelecendo um novo patamar de desempenho no campo do reconhecimento de fala E2E.

Apesar dos avanços nos sistemas ASR E2E, ainda há lacunas na otimização dos encoders. A arquitetura encoder-decoder, embora poderosa, não resolve todos os desafios do ASR, como robustez a ruídos, generalização a sotaques e eficiência em tempo real. Essas limitações destacam a necessidade de mais pesquisas e novas técnicas de otimização. Neste contexto, o aprendizado hierárquico [20] [21] emerge como uma abordagem interessante. Esta técnica explora as representações intermediárias geradas pelo *encoder*, incorporando tarefas auxiliares em diferentes níveis da arquitetura. Ao estruturar o aprendizado em múltiplos níveis e integrar tarefas relacionadas, busca-se capturar re-

presentações mais ricas e robustas dos sinais tratados, melhorando o desempenho global do sistema ASR. Além disso técnicas dedicadas de otimização, como o uso de funções custo auxiliares treinadas em conjunto, demonstram que é possível aprimorar o desempenho do sistema sem alterar demasiadamente sua arquitetura [22].

Este trabalho propõe investigar, a partir de um modelo de referência, a incorporação de informações fonéticas canônicas do português brasileiro como uma tarefa auxiliar de forma hierárquica no *encoder*. Com isso, este estudo busca avaliar como a abordagem hierárquica multitarefa pode otimizar o *encoder* sem alterar sua arquitetura de forma substancial, levando a um sistema ASR mais robusto e preciso.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho tem como objetivo principal investigar uma abordagem de aprendizado de máquina hierárquica para reconhecimento automático de fala, focando na otimização do *encoder* do sistema através da integração de técnicas de aprendizado conjunto multitarefas, utilizando um *decoder* CTC.

1.1.2 Objetivos Específicos

- Elaborar uma análise da eficácia da incorporação de informações fonéticas em camadas intermediárias de um *encoder* com o objetivo de melhorar a transcrição do áudio.
- Fazer uma avaliação comparativa do desempenho do modelo avaliado em relação ao modelo de referência estabelecido sem essa incorporação.
- Buscar possíveis diretrizes para a otimização de arquiteturas ASR E2E em cenários de aprendizado multitarefa.

1.2 Estrutura do Documento

A estrutura deste documento está organizada da seguinte forma: O Capítulo 2 apresenta uma revisão teórica abrangente, cobrindo os fundamentos do ASR E2E, descrevendo as arquiteturas CTC, RNN-T e

Zipformer, o aprendizado hierárquico multitarefa e treinamento conjunto. O Capítulo 3 detalha o desenvolvimento do modelo investigado, estabelecendo um modelo de referência (*baseline*), incluindo a metodologia experimental, descrição dos conjuntos de dados utilizados e protocolos de avaliação, mostrando e discutindo os resultados obtidos. O Capítulo 4 conclui o trabalho, sintetizando as principais conclusões e apontando pesquisas futuras.

CAPÍTULO 2

Fundamentação Teórica

O reconhecimento automático de fala representa um desafio significativo no campo do processamento de sinais e aprendizado de máquina, envolvendo a transformação de sinais acústicos contínuos em sequências discretas de unidades linguísticas. Esta é uma tarefa complexa devido à natureza variável e assíncrona da relação temporal entre os sinais de entrada e as sequências de saída.

Este capítulo descreve duas arquiteturas independentes de *decoder* E2E para encontrar esse alinhamento, os decodificadores CTC e RNN-T. Em seguida, descreve como a arquitetura CTC pode ser treinada conjuntamente com a arquitetura RNN-T de forma eficiente e, então, são descritas algumas formas otimizadas do *decoder* RNN-T, o podado e sem estado. Por último, o *encoder* E2E é abordado, descrevendo o *Zipformer* e como informações de suas camadas intermediárias podem ser usadas para um treinamento de forma hierárquica.

2.1 Encontrando o Alinhamento Temporal Entrada e Saída em Sistemas ASR

Um dos problemas fundamentais no desenvolvimento de sistemas ASR é o desafio do alinhamento temporal entre a entrada acústica e a saída textual [23]. Pode-se definir o problema da seguinte maneira:

(i) $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{T'}) \in \mathbb{R}^{D \times T'}$ uma sequência de T' *frames* de áudio parametrizados em características acústicas em vetores de dimensão D.

(ii) $\mathbf{Y} = (y_1, y_2, ..., y_L) \in \mathbf{\Sigma}^{L'}$ uma sequência de L símbolos textuais que fazem parte do conjunto $\mathbf{\Sigma}$ com L' símbolos.

O conjunto $\mathbf{\Sigma}$ é composto de todas as possíveis saídas do sistema e pode ser formado, por exemplo, por um conjunto finito de palavras, pelo conjunto de letras de um alfabeto ou até mesmo um conjunto de sílabas. O desafio central no sistema ASR E2E está no fato de que $T' \neq L$. Além disso, a relação entre o tamanho dos sinais de entrada e de saída é altamente variável, principalmente no que diz respeito, por exemplo, a:

- Taxa de Locução: A velocidade da fala varia significativamente intra-locutor e inter-locutores, ou seja, para um mesmo número de símbolos L há diferentes valores de T' que mapeiam para a mesma sequência.
- Características Fonéticas: Diferentes sons têm durações intrínsecas variadas. Alguns fonemas plosivos como o som de /p/ e /b/ são constantemente curtos enquanto vogais (/a/, /e/, /i/...) têm maior variação de duração.
- Pausas e hesitações: disfluências na fala introduzem variabilidade no sinal $\mathbf{X}$.

Dessa forma a relação entre $\mathbf{X}$ e $\mathbf{Y}$ não é uma correspondência um-para-um. Essa relação pode ser representada por uma função $\mathbf{f}$ que mapeia $\mathbf{X}$ para $\mathbf{Y}$ de forma não injetora e não sobrejetora:

$$\mathbf{f} : \mathbb{R}^{D \times T'} \rightarrow \ \Sigma^{L'} \tag{2.1}$$

Por não ser injetora, podem existir valores distintos $x_i \neq x_j$ que apontam para o mesmo símbolo y_k. Por ser não sobrejetora, pode existir uma série de sequências de símbolos em $\mathbf{Y}$ que não são mapeadas a partir de um sinal de entrada, seja ele por exemplo um sinal aleatório ou devido às próprias capacidades do modelo de não reconhecer corretamente sinais válidos. Essas limitações estão ligadas a características do problema de transcrição em si e influenciam na robustez e na generalização a partir de um conjunto de dados suficientemente representativo.

Esta complexidade torna muito difícil a criação de um conjunto de dados para treinamento com alinhamentos explícitos *frame-a-frame* entre $\mathbf{X}$ e $\mathbf{Y}$ anotados manualmente. Consequentemente, técnicas de aprendizado supervisionado, como a minimização da entropia cruzada, não são diretamente aplicáveis. A ausência de alinhamentos explícitos nos dados de treinamento motivou o desenvolvimento de técnicas especializadas que permitem o aprendizado nesse cenário sem a necessidade da existência destes alinhamentos. Tais abordagens buscam "aprender" implicitamente o alinhamento durante o processo de treinamento.

Entre as abordagens para o problema descrito acima destacam-se o CTC [24] e RNN-T [23]. Essas arquiteturas funcionam como o *decoder* da imagem da Figura 1.3. Ambos os métodos fazem uso de um símbolo especial $\langle b \rangle$, chamado de *blank*, estendendo o conjunto $\mathbf{\Sigma}$ para $\mathbf{\Sigma}'$. Este símbolo é utilizado durante o treinamento do sistema para aprendizado do alinhamento e tem tratamento especial durante a inferência.

$$\mathbf{\Sigma}' = \mathbf{\Sigma} \cup \langle b \rangle \tag{2.2}$$

De forma geral, para encontrar o alinhamento entre entrada e saída, é necessário tratar da expressão $P(\mathbf{Y}|\mathbf{X})$. Lembrando que o sinal de entrada $\mathbf{X}$ é processado pelo *encoder* gerando a representação intermediária $\mathbf{H}(\mathbf{X})$ e que a sequência de saída $\mathbf{Y}$ é encontrada pelo decoder ao processar $\mathbf{H}(\mathbf{X})$. Então

$$P(\mathbf{Y}|\mathbf{X}) = P(\mathbf{Y}|H(\mathbf{X})). \tag{2.3}$$

O alinhamento entre $\mathbf{X}$ e $\mathbf{Y}$ por estes métodos é encontrado ao definir um conjunto com todos os possíveis alinhamentos $\mathbf{A}$ onde cada $A = (a_1, a_2, ..., a_{T'}) \in \mathbf{A}$ que representa $\mathbf{Y}$, é uma sequência de símbolos

de $\mathbf{\Sigma}'$ e permite marginalizar $P(\mathbf{Y}|H(\mathbf{X}))$ sobre todos estes alinhamentos. Dessa forma, o problema passa a considerar os alinhamentos na forma

$$P(\mathbf{Y}|H(\mathbf{X})) = \sum_{\mathbf{A}} P(\mathbf{Y}, A|H(\mathbf{X})). \tag{2.4}$$

Decompondo a probabilidade conjunta acima, tem-se a probabilidade marginal de todos os alinhamentos [11]:

$$P(\mathbf{Y}|H(\mathbf{X})) = \sum_{\mathbf{A}} P(\mathbf{Y}|A, H(\mathbf{X}))P(A|H(\mathbf{X})). \tag{2.5}$$

Nas Seções 2.2 e 2.3 é explorado como o uso do símbolo especial *blank* do conjunto $\mathbf{\Sigma}'$ permite o processo de aprendizado dos alinhamentos entre entrada e saída para os decodificadores CTC e RNN-T, respectivamente.

2.2 Decodificador CTC

O decodificador CTC [24] processa a representação intermediária $\mathbf{H}$ para gerar a sequência $\mathbf{Y}$ aprendendo o alinhamento através da presunção de independência entre cada símbolo individual de $\mathbf{Y}$ decodificado na saída.

Durante o processo de treinamento, além de fazer uso do símbolo $\langle b \rangle$, o CTC permite a repetição de símbolos válidos no seu alinhamento. Para obter a sequência final $\mathbf{Y}$, os símbolos repetidos são colapsados em um único símbolo e, em seguida, os símbolos $\langle b \rangle$ são removidos. Ou seja, ao considerar todas as prováveis repetições e as posições do $\langle b \rangle$, existe um conjunto de alinhamentos que representa uma única sequência $\mathbf{Y}$. Um exemplo é mostrado a seguir: considerando $T = 15$, $\mathbf{Y} = (P, E, S, Q, U, I, S, A)$ e o conjunto de símbolos $\mathbf{\Sigma}$ sendo composto pelas letras do alfabeto, tem-se dois exemplos de alinhamentos $A_{1_{\text{CTC}}}$ e $A_{2_{\text{CTC}}}$ que representam esse $\mathbf{Y}$:

$$A_{1_{\text{CTC}}} = (P, E, E, \langle b \rangle, S, S, \langle b \rangle, \langle b \rangle, Q, U, \langle b \rangle, I, S, A, A)$$

e

$$A_{2_{\text{CTC}}} = (P, E, \langle b \rangle, S, S, S, \langle b \rangle, \langle b \rangle, \langle b \rangle, Q, U, I, S, \langle b \rangle, A).$$

Dessa forma, ao colapsar as repetições de símbolos como, por exem-

plo, (E, E), (S, S, S) e $(\langle b \rangle, \langle b \rangle)$ e remover todos os símbolos $\langle b \rangle$, tem-se de volta a sequência $\mathbf{Y}$. Ao considerar todos os alinhamentos e marginalizar sobre eles para obter a probabilidade final, tem-se

$$P_{\text{CTC}}(\mathbf{Y}|H(\mathbf{X})) = \sum_{A \in \mathbf{A}} P(A|H(\mathbf{X})) \tag{2.6}$$

onde cada $P(A|H(\mathbf{X}))$ é calculado como o produto das probabilidades de cada símbolo no alinhamento em cada instante de tempo, somando todos os alinhamentos $\mathbf{A}$ [11]:

$$P_{\text{CTC}}(\mathbf{Y}|H(\mathbf{X})) = \sum_{A \in \mathbf{A}} \prod_{t=0}^{T} P(a_t|\mathbf{h}_t). \tag{2.7}$$

Em suma, o modelo CTC consiste em um sistema que modela a probabilidade de um símbolo do alinhamento (a_t) ocorrer em um determinado instante de tempo (t) observando um sinal de entrada ($\mathbf{h}_t$). Essa distribuição de probabilidades em cada t é gerada através de uma camada *Softmax* sobre a dimensão de $\mathbf{\Sigma}'$. A arquitetura CTC é exemplificada na Figura 2.1. É importante notar que o CTC assume independência entre as previsões de símbolos em diferentes instantes de tempo para cada entrada. Essa suposição de independência simplifica os cálculos, mas também representa uma limitação do modelo, pois não captura explicitamente dependências entre símbolos na sequência de saída. O CTC também tem o requisito de que $L \leq T$, ou seja, o número de símbolos de saída não pode ser maior que o número de frames processados da entrada.

O CTC utiliza o algoritmo *forward-backward* [24] com uma técnica de programação dinâmica para calcular eficientemente as probabilidades considerando todos os alinhamentos. Este algoritmo opera em duas etapas na sequência de entrada. No primeiro passo *forward*, é calculada a probabilidade de chegar a cada estado intermediário do alinhamento, movendo-se do início para o fim da sequência. No segundo passo *backward*, é calculada a probabilidade de completar a sequência a partir de cada estado intermediário, movendo-se do fim para o início. Combinando estas duas etapas, o algoritmo pode calcular a probabilidade total de todos os alinhamentos válidos sem ter que enumerá-los explicitamente, o que seria computacionalmente inviável devido ao grande

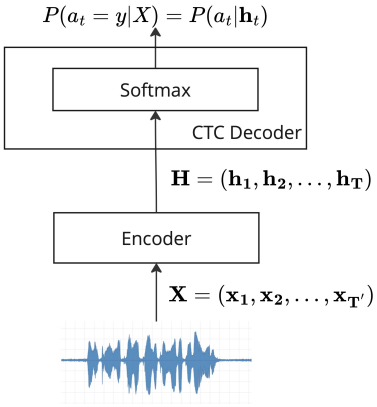

Figura 2.1: Estrutura de um Modelo CTC. Fonte.
Fonte: do Autor.

número de alinhamentos. Esta abordagem de programação dinâmica permite que o CTC compute os gradientes eficientemente durante o treinamento, aproveitando os cálculos intermediários das duas etapas do algoritmo.

A função custo do CTC é definida como o negativo do logaritmo da probabilidade condicional de $\mathbf{Y}$ em $\mathbf{X}$:

$$\mathcal{L}_{\text{CTC}} = -\log(P_{\text{CTC}}(\mathbf{Y}|H(\mathbf{X}))). \tag{2.8}$$

Esta formulação transforma o problema de maximização de probabilidade em um problema de minimização de custo, mais adequado para técnicas de otimização em aprendizado de máquina. Assim, o modelo pode ser treinado para minimizar a função custo usando técnicas de otimização baseadas em gradiente, como *Stochastic Gradient Descent* - SGD [25]. Tal abordagem permite que o CTC aprenda os alinhamentos mais prováveis entre a entrada e a saída durante o processo de treinamento, sem a necessidade de alinhamentos explícitos nos dados de treinamento, sendo eles compostos apenas da transcrição ortográfica do conteúdo falado nos trechos selecionados.

Por fim, apesar de suas vantagens, o CTC apresenta limitações importantes. A suposição de independência entre as saídas em diferentes instantes de tempo, embora simplifique a abordagem, ignora as de-

pendências linguísticas existentes na fala. Além disso, a falta de um mecanismo para modelar o contexto linguístico pode levar à geração de sequências de saída improváveis ou incoerentes do ponto de vista da linguagem, como repetições de sílabas ou palavras grafadas de maneira incorreta. Essas limitações motivaram o desenvolvimento de abordagens mais avançadas, capazes de capturar melhor as complexidades inerentes à fala e à linguagem, como o decodificador RNN-T.

2.3 Decodificador RNN-T

O decodificador RNN-T [23] surge como uma evolução natural do CTC, introduzindo uma arquitetura mais sofisticada que permite a modelagem das dependências entre os elementos da saída, diferentemente do *decoder* CTC. Isso é feito através do uso dois módulos chamados de ***Rede Preditora*** e ***Rede de Junção*** ilustrados na Figura 2.2.

A rede preditora é responsável por adicionar a dependência dos símbolos já previstos anteriormente através de uma variável de estado preditor $\mathbf{p}$ para prever novos símbolos, enquanto a rede de junção, de forma conjunta, observa o estado preditor da rede preditora e os sinais de entrada da saída *encoder* $\mathbf{h}_t$ para prever um novo símbolo. Dessa forma, a arquitetura RNN-T consegue adicionar em suas predições informações acústicas provenientes do *encoder* e informações linguísticas provenientes da rede de predição.

O RNN-T utiliza o símbolo $\langle b \rangle$ descrito na seção anterior, porém de forma diferente. O RNN-T não admite o processo de repetição e colapso de símbolos válidos. Para fornecer o alinhamento, a rede de junção entende o símbolo $\langle b \rangle$ como um "símbolo nulo", ou seja, decide por fornecer na saída ou um símbolo válido de $\mathbf{\Sigma}$, que altera o estado preditor, ou nenhum símbolo, através da decisão por *blank*, não alterando o estado preditor. Esse símbolo não é utilizado pela rede de predição, que observa apenas os símbolos válidos.

Em um alinhamento RNN-T para um determinada posição $\tau \leq T$ na sequência de saída, existe um certo número i_τ de símbolos válidos da sequência parcial do alinhamento. Dessa forma o número de símbolos $\langle b \rangle$ no alinhamento parcial é $\tau - i_\tau - 1$. Um exemplo de alinhamento, $A_{1_{\text{RNNT}}}$, é mostrado abaixo considerando $T = 15$

$$A_{1_{\text{RNNT}}} = (\langle b \rangle, P, \langle b \rangle, E, S, Q, \langle b \rangle, \langle b \rangle, U, \langle b \rangle, I, S, \langle b \rangle, A, \langle b \rangle)$$

Neste caso de exemplo em um determinado instante, por exemplo $\tau = 10$, o número símbolos $\langle b \rangle$ que ocorreram na sequência parcial é 4 porque até o décimo símbolo houveram 5 símbolos válidos, a sequência parcial é (P, E, S, Q, U) e a sequência completa até $\tau = 10$ é $A'_{1_{\text{RNNT}}} = (\langle b \rangle, P, \langle b \rangle, E, S, Q, \langle b \rangle, \langle b \rangle, U, \langle b \rangle)$.

Na arquitetura RNN-T, ao marginalizar sobre todos os alinhamentos, a probabilidade final $P_{\text{RNNT}}(\mathbf{Y}|H(\mathbf{X}))$ é dada de forma geral, considerando os alinhamentos a_τ por

$$P_{\text{RNNT}}(\mathbf{Y}|H(\mathbf{X})) = \sum_{A \in \mathbf{A}} \prod_{\tau=0}^{T} P(a_\tau | a_{\tau-1}, ..., a_0, H(\mathbf{X})). \tag{2.9}$$

Após considerar apenas os símbolos válidos, desconsiderando os símbolos *blank* dentro dos símbolos do alinhamento A tem-se:

$$P_{\text{RNNT}}(\mathbf{Y}|H(\mathbf{X})) = \sum_{A \in \mathbf{A}} \prod_{\tau=0}^{T} P(a_\tau | y_{i_\tau}, y_{i_{\tau-1}}, ..., y_0, \mathbf{h}_{\tau - i_\tau}). \tag{2.10}$$

Dessa forma a probabilidade final fica condicionada, além da entrada transformada $\mathbf{X}$, também pelos símbolos anteriores válidos de saída $y_{i_{\tau \ldots 0}}$, de forma que essa condicionante é representada pelo sinal latente $\mathbf{p}_{i_\tau}$ da saída da rede preditora. Assim a rede preditora, em conjunto com a rede de junção, consegue decidir por fornecer um símbolo válido, avançando o estado preditor e a entrada $\mathbf{h}$ ou o símbolo nulo $\langle b \rangle$, avançando apenas na entrada e seguindo até o fim da sequência. Dessa forma:

$$P_{\text{RNNT}}(\mathbf{Y}|H(\mathbf{X})) = \sum_{A \in \mathbf{A}} \prod_{\tau=0}^{T} P(a_\tau | \mathbf{p}_{i_\tau}, \mathbf{h}_{\tau - i_\tau}). \tag{2.11}$$

A função custo do RNN-T é definida como o negativo do logaritmo da probabilidade condicional $\mathcal{L}_{\text{RNNT}} = -\log(P_{\text{RNNT}}(\mathbf{Y}|H(\mathbf{X})))$. O uso do negativo do logaritmo, converte o problema de maximização da probabilidade em um problema de minimização do custo. O cálculo de $P(\mathbf{Y}|H(\mathbf{X}))$ no RNN-T deve considerar todos os possíveis alinhamentos entre a sequência de entrada $\mathbf{X}$ e a sequência de saída $\mathbf{Y}$, além das emissões do símbolo especial $\langle b \rangle$, que representa um avanço no tempo sem geração de um símbolo de saída válido.

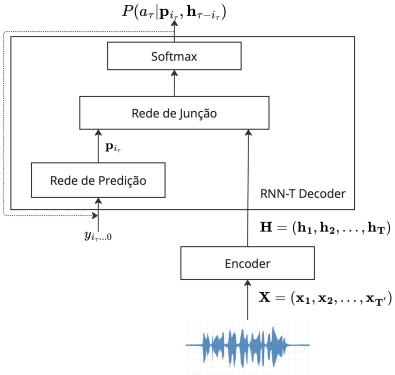

Figura 2.2: Estrutura de um Modelo RNN-T.
Fonte: do Autor.

Para calcular eficientemente esta probabilidade, o RNN-T também utiliza um algoritmo *forward-backward* de programação dinâmica. Este algoritmo constrói uma variável $\alpha(t, l)$, onde t representa o índice do *frame* da entrada e l o índice do símbolo na saída parcial. Cada elemento $\alpha(t, l)$ desta variável encapsula a probabilidade de ter gerado os primeiros l símbolos da sequência de saída utilizando os primeiros t *frames* da entrada.

O algoritmo do RNN-T é mais complexo do que o do CTC, pois precisa considerar duas possibilidades em cada passo: emitir um símbolo de saída válido ou emitir um $\langle b \rangle$, indicando apenas um avanço no tempo. Segundo descrito em [23], isso é representado como

$$\alpha(t, l) = \alpha(t-1, l)P(\langle b \rangle | t-1, l) + \alpha(t, l-1)P(y_l|t, l-1). \quad (2.12)$$

Nesta expressão, $P(\langle b \rangle | t-1, l)$, calculada pela rede de junção, representa a probabilidade de não emitir um símbolo válido no estado atual e avançar para o próximo *frame* de entrada, ilustrado com setas verdes na Figura 2.3 (a). $P(y_l|t, l-1)$ é a probabilidade de emitir o símbolo válido y_l, ilustrado com uma seta vermelha na Figura 2.3 (a), no contexto atual da rede preditora até o símbolo $l-1$. Esta formulação permite ao modelo RNN-T equilibrar de forma flexível o progresso através da sequência de entrada com a geração da sequência de saída, uma vantagem em relação ao modelo CTC. Em resumo, o RNN-T pode decidir

permanecer no mesmo estado de saída e processar o próximo *frame* de entrada, ou gerar um novo símbolo de saída para avançar no tempo e no estado atual. O cálculo final da probabilidade $P(\mathbf{Y}|H(\mathbf{X}))$ é obtido do elemento final da variável $\alpha(T, L)$. O algoritmo de *backward* calcula e propaga os gradientes e permite a otimização dessas probabilidades.

O RNN-T, embora ofereça uma modelagem mais sofisticada que o CTC, apresenta um custo computacional significativamente maior. Devido à presença da rede de predição e à necessidade de considerar probabilidades para cada combinação de estado acústico e linguístico, exige recursos computacionais substancialmente maiores. Para abordar esse desafio, formas otimizadas foram desenvolvidas [26] [27] [28], oferecendo implementações eficientes para diferentes *hardwares*. Além disso, otimizações da arquitetura como o RNN-T podado [29] e o RNN-T sem estado [30] foram propostas. Estas técnicas são cruciais para tornar o RNN-T viável em aplicações práticas com recursos de *hardware* limitados equilibrando a sofisticação do modelo com a eficiência computacional necessária para sistemas de reconhecimento de fala.

2.3.1 Decodificador RNN-T Podado e Sem Estado

A especialização do decodificador RNN-T podado [29] é uma técnica que visa reduzir o custo computacional e uso de memória associados ao treinamento de modelos para ASR E2E com decodificadores RNN-T. A ideia principal é reduzir o espaço de buscas durante o cálculo da função custo no treinamento na variável $\alpha(T, L)$ considerando um subconjunto dos símbolos da sequência de saída.

No RNN-T tradicional, o espaço de busca é representado por uma matriz de dimensões (T, L), onde T representa o tamanho da entrada e L o tamanho da sequência de saída. Para cada posição (t, l) nesta matriz, é necessário calcular probabilidades para todos os símbolos do vocabulário, resultando em um objeto com dimensões (T, L, L') onde L' é a quantidade de símbolos em $\mathbf{\Sigma}'$. Isso leva a um uso intensivo de memória e computação, especialmente para modelos com $\mathbf{\Sigma}$ com muitos símbolos. Por exemplo, um dado de treinamento de um sinal com 10 segundos de duração possui aproximadamente $T = 1000$ *frames* e tem sua transcrição representada por $L = 132$ símbolos, dessa forma, considerando um vocabulário de $L' = 1000$ símbolos a quantidade de

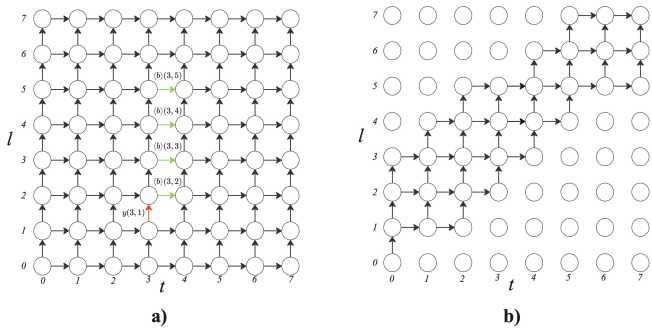

Figura 2.3: Espaço de busca para cálculo da função custo RNN-T. a) RNN-T tradicional. b) RNN-T Podado.
Fonte: Fonte: [29]

itens nesta matriz é:

$$T \times L \times L' = 1000 \times 132 \times 1000 = 132.000.000.$$

Se cada item é representado por 4 bytes, isso representa 528.000.000 bytes ou $0,5$GB ao considerar uma representação em *float32*. Isso representa apenas um tensor de apenas um exemplo na saída da rede de junção. Ao adicionar à conta o tamanho de *batch* ou considerar outros tensores de outros módulos o uso de memória é ainda maior.

O RNN-T podado reduz este espaço de busca para (T, S, L'), onde $S \ll L$ é um número fixo e pequeno (por exemplo, 4 ou 5) de posições em L consideradas durante o cálculo da função custo. Esse novo intervalo de buscas é exemplificado na Figura 2.3 (b). O algoritmo de poda, criado com informações das características da fala e com testes empíricos, seleciona S posições para cada t, e apenas para essas posições calcula as probabilidades completas usando a rede de junção do modelo. Esta abordagem não só reduz drasticamente o uso de memória e o tempo de computação, mas também tem a capacidade de manter a generalização do modelo.

A especialização do decodificador RNN-T sem estado [30] é mais uma técnica que busca reduzir a complexidade computacional. O objetivo principal nessa abordagem é remover a recorrência completa da rede de predição. A rede de predição no RNN-T tradicional é composta

por uma estrutura recorrente (RNN) que processa todo o histórico de saída com símbolos válidos da rede de junção para gerar **p**. Ao remover as recorrências o espaço computacional passa a ser reduzido, uma vez que, como demonstrado pelos autores, ao observar apenas o último símbolo de saída, no lugar da sequência inteira, a técnica proposta alcança desempenho comparável ao RNN-T tradicional completo. Além disso supera os modelos CTC, que não têm dependência de saídas anteriores. Essas descobertas sugerem que, para muitas tarefas de reconhecimento de fala, um único símbolo de contexto de saída é suficiente para alcançar um bom desempenho em *decoders* RNN-T. Isso oferece uma alternativa eficiente e simplificada comparada ao RNN-T tradicional, mantendo a qualidade das transcrições em muitos cenários práticos.

2.4 Aprendizado Multitarefas e Treinamento Conjunto CTC

O aprendizado conjunto de modelos CTC e modelos baseados em atenção tem se mostrado uma abordagem promissora para melhorar o desempenho e a eficiência do treinamento em sistemas de ASR [22]. Esta técnica visa superar as limitações individuais de cada modelo, combinando suas características em uma estrutura de aprendizado chamada aprendizado multitarefa de treinamento conjunto CTC.

Os modelos de atenção *Transformer* possuem dificuldades em lidar com sinais de áudio para treinar um sistema ASR. Isso acontece porque os módulos de atenção tem dificuldade em lidar com sinais que possuem muita variabilidade, que é a característica de sinais de áudio para transcrever. Diferentemente de quando se está lidando com palavras e textos, que a informação está na forma estruturada dos dados, com sinais de fala, além da ordem do alinhamento, existem também as variabilidades do sinal. O autor em [22] propõe que treinar conjuntamente um modelo CTC, mais simples, com outro *decoder* mais complexo, acelera e facilita a convergência do treinamento. Essa estrutura é exemplificada na Figura 2.4.

A ideia central propõe que a função custo total multitarefa $\mathcal{L}_{\text{Total}}$ seja uma combinação das funções custo $\mathcal{L}_{\text{CTC}}$ e $\mathcal{L}_{\text{RNNT}}$, dado que o outro decoder seja do tipo RNN-T, ponderadas por fatores α e β que controlam o quanto o gradiente de cada parte do sistema influencia na

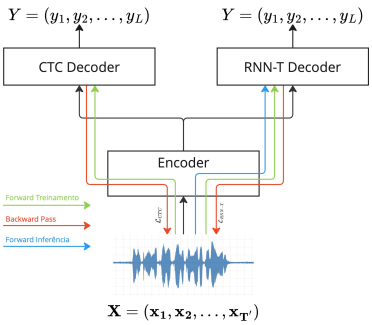

Figura 2.4: Treinamento Conjunto de um Decodificador RNN-T com CTC em Aprendizado Multitarefas.
Fonte: do Autor.

otimização dos módulos conjuntos.

$$\mathcal{L}_{\text{Total}} = \alpha \mathcal{L}_{\text{CTC}} + \beta \mathcal{L}_{\text{RNN-T}} \tag{2.13}$$

Nesta estrutura o encoder é compartilhado entre os decodificadores CTC e RNN-T. Este componente comum processa a entrada para que seja utilizada por ambos os decodificadores. O processo de treinamento, indicado pelas setas verdes (*forward*) e vermelhas (*backward*) na Figura 2.4, ocorre em dois fluxos de dados:

- ***Forward Pass*** (Treinamento): Durante esta fase, o sinal de entrada é processado pelo encoder compartilhado. A saída do encoder com a representação de alto nível do sinal alimenta simultaneamente o decodificador CTC e o decodificador RNN-T. Cada decodificador gera sua própria sequência de saída $\hat{\mathbf{Y}}$, que é usada para gerar $\mathcal{L}_{\text{CTC}}$ e $\mathcal{L}_{\text{RNNT}}$ individualmente.
- ***Backward Pass***: Nesta fase, os gradientes são calculados e propagados através da rede. A função custo combinada é utilizada para atualizar os pesos do modelo, incluindo os parâmetros compartilhados do encoder.

Segundo os autores o uso dessa arquitetura conjunta possui alguns propósitos:

(*i*) Aceleração do treinamento: O CTC, com sua natureza de alinhamento monotônico, ajuda a guiar o treinamento, especialmente nos estágios iniciais, facilitando a convergência.

(*ii*) Regularização: Atua como uma forma de regularização, potencialmente melhorando a generalização do modelo.

(*iii*) Alinhamento auxiliar: Fornece um mecanismo auxiliar de alinhamento que pode ser útil durante a inferência para tarefas que requerem informações de temporização.

Durante a inferência, indicada pela seta azul na imagem, apenas o decodificador RNN-T é utilizado, aproveitando sua capacidade superior de modelagem para produzir saídas mais coerentes linguisticamente. O decodificador CTC é usado apenas para o treinamento.

2.5 Codificador Zipformer

O *encoder* Zipformer [31] transforma o sinal de entrada $\mathbf{X}$ em uma representação intermediária $\mathbf{H}$ de alto nível que será posteriormente processada pelo *decoder* para gerar a sequência de saída. Ele surge como uma alternativa para abordar os desafios de complexidade computacional presentes em arquiteturas baseadas em *Transformer* [19], particularmente em tarefas de processamento de sequências longas como é o caso do reconhecimento de fala. Assim como o *Transformer*, o *Zipformer* utiliza mecanismos de atenção para codificar dependências do sinal de entrada. No entanto, o *Zipformer* introduz modificações na arquitetura para torná-la mais eficiente e eficaz especificamente para tarefas de reconhecimento de fala.

Uma característica importante significativa dos modelos *Transformer* tradicionais é sua complexidade quadrática em relação ao comprimento da sequência de entrada. Esta característica torna-se especialmente problemática ao lidar com sinais de áudio, que frequentemente são sequências com muitos *frames*. O *Zipformer* aborda essa questão através de uma estrutura que incorpora operações de *downsampling* e *upsampling* em todas as camadas internas do *encoder*. Isso modifica o tamanho do sinal, reduzindo a questão da complexidade.

O *Zipformer* adota uma arquitetura em forma de U, inspirado na estrutura U-Net utilizada em tarefas de visão computacional [32]. Esta

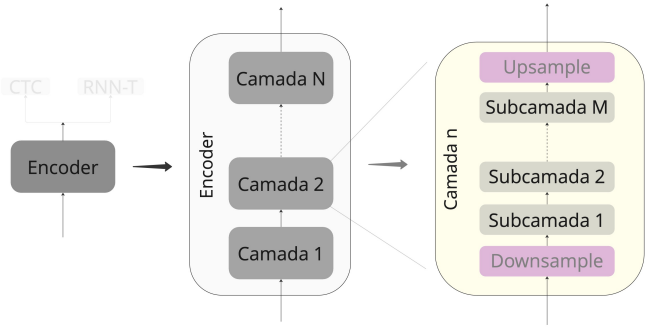

Figura 2.5: Esquema de *Downsampling* e *Upsampling* do *encoder Zipformer*.
Fonte: do Autor.

estrutura permite que o modelo processe a informação em múltiplas resoluções processando a sequência de entrada por uma série de estágios que operam em resoluções temporais diferentes. Entre esses estágios, módulos de *downsample* reduzem progressivamente a resolução temporal, permitindo que as camadas intermediárias operem em taxas de amostragem mais baixas. Módulos de *upsampling* entre subcamadas restauram parcialmente a resolução. Este esquema é ilustrado na Figura 2.5. Esta abordagem de processamento em múltiplas escalas oferece como vantagens:

- Redução da complexidade computacional: Ao processar a sequência em resoluções mais baixas, o *Zipformer* reduz substancialmente o custo computacional em comparação com modelos que operam na resolução máxima durante todo o processamento como os modelos *Transformer*.

- Captura eficiente de dependências de longo alcance: As camadas que operam em resoluções mais baixas têm um campo receptivo efetivamente maior, facilitando a modelagem de dependências de longo prazo no sinal.

- Representações multi-escala: A arquitetura permite que o modelo aprenda representações em múltiplas escalas temporais, potencialmente capturando tanto detalhes finos quanto estruturas mais globais do sinal de fala.

2.6 Aprendizado Hierárquico

O Aprendizado Hierárquico em sistemas de reconhecimento automático de fala representa um aprimoramento na arquitetura de modelos E2E, explorando a riqueza das representações intermediárias geradas pelos *encoders*. Esta abordagem baseia-se na premissa de que as camadas intermediárias de redes neurais profundas capturam informações relevantes e hierárquicas sobre o sinal de entrada, uma observação corroborada por estudos do aprendizado profundo como por exemplo na área de visão computacional [33].

No contexto específico de ASR, o trabalho de [34] demonstrou que as representações intermediárias em modelos E2E contêm informações fonéticas significativas. Analogamente às redes convolucionais em visão computacional, onde camadas iniciais detectam bordas e texturas, e camadas mais profundas identificam formas e objetos complexos, os *encoders* em sistemas ASR desenvolvem uma hierarquia similar de representações acústico-fonéticas. As camadas inferiores tendem a capturar características acústicas de baixo nível, enquanto as camadas superiores abstraem essas informações em representações mais complexas.

Aproveitando essas descobertas, os autores em [20] propuseram uma arquitetura de aprendizado ***hierárquico*** para ASR. Na abordagem, além da tarefa principal de transcrição como já descrito em detalhes neste trabalho, tarefas auxiliares são introduzidas em camadas intermediárias do *encoder*. Essas tarefas auxiliares, tipicamente relacionadas à classificação fonética, servem a dois propósitos principais:

(*i*) Fornecem supervisão adicional, guiando o modelo a aprender representações intermediárias mais robustas e informativas.

(*ii*) Atuam como uma forma de regularização, potencialmente melhorando a generalização do modelo.

Na prática a implementação do aprendizado hierárquico em ASR envolve a adição de classificadores auxiliares conectados a camadas intermediárias do *encoder*. Esses classificadores são treinados com um sinal de saída diferente da sequência da informação principal. A função custo global do modelo passa a ser uma combinação ponderada da perda da tarefa principal e das perdas das tarefas auxiliares. Um aspecto crucial dessa abordagem é a escolha das camadas para a aplicação das tarefas auxiliares e a determinação dos pesos relativos de

cada componente da função de perda. O autor sugere que a colocação ótima dessas tarefas auxiliares pode variar dependendo da arquitetura específica do modelo e da natureza das tarefas auxiliares escolhidas.

CAPÍTULO 3

Desenvolvimento

Este capítulo detalha o desenvolvimento e a avaliação de um sistema de reconhecimento automático de fala baseado em aprendizado hierárquico multitarefa. O estudo parte de um modelo *baseline encoder-decoder*, utilizando um *encoder Zipformer* e um *decoder* RNN-T treinado conjuntamente com um *decoder* CTC, e propõe avaliar uma arquitetura estendida que incorpora informações fonéticas nas camadas intermediárias do *encoder* através de um decodificador CTC adicional, complementando o objetivo principal de transcrição ortográfica. A implementação utiliza um conjunto de dados de treinamento, com 2000 horas em português brasileiro, e aplica técnicas de processamento de sinais de áudio e aumento de dados. A avaliação do sistema é realizada em quatro conjuntos de teste, onde as avaliações conduzidas visam determinar em qual camada intermediária do *encoder* com informações fonéticas o sistema treinado produz o menor WER.

No contexto do presente trabalho, a investigação se concentra na aplicação dessa abordagem hierárquica a um *encoder* moderno, especificamente o *Zipformer*. O foco está na incorporação de sequências fonéticas canônicas como uma tarefa auxiliar, ignorando o uso de variantes fonéticas. Uma sequência fonética canônica representa a pronúncia pa-

drão ou ideal de uma palavra conforme estabelecida em dicionários e descrições linguísticas formais, sem considerar as variações dialetais, regionais ou individuais que ocorrem naturalmente na fala. Esta escolha é motivada pela hipótese de que a introdução da sequência fonética explícita pode melhorar a capacidade do modelo de generalizar. Uma análise subsequente busca determinar em qual camada do *encoder* a adição dessa informação proporciona o maior benefício para a tarefa global de transcrição, contribuindo assim para uma compreensão mais profunda da interação entre diferentes níveis de representação em sistemas ASR modernos.

3.1 Baseline

A estrutura do modelo *baseline* para o sistema é ilustrada na Figura 2.4 onde o *encoder* é baseado no *Zipformer*.

Processamento da Entrada

O sinal de entrada $\mathbf{X}$ do sistema ASR é gerado a partir de sinais de áudio amostrados em 16kHz. Esses sinais são janelados em *frames* de 25ms (com passo de 10ms) dos quais são extraídos 83 atributos acústicos, sendo 80 referentes às energias de banco de filtros (*fbank*) [6] e 3 atributos de *pitch* [35]. Nesta etapa de parametrização, foram utilizadas as ferramentas do *Kaldi*[1] [36].

Encoder Zipformer

O *encoder Zipformer* é uma estrutura composta por 5 camadas empilhadas. Cada camada contém um número variável de subcamadas *Zipformer*, distribuídas da seguinte forma:

- **Camada 1**: 2 subcamadas;
- **Camada 2**: 4 subcamadas;
- **Camada 3**: 3 subcamadas;
- **Camada 4**: 2 subcamadas;

[1] `http://kaldi-asr.org`

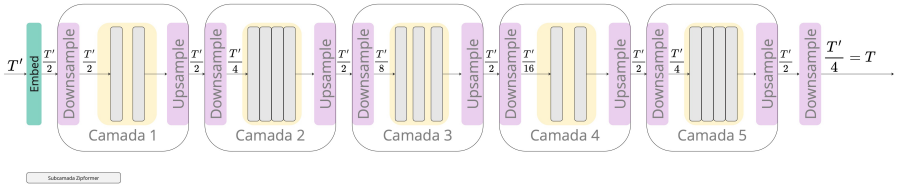

Figura 3.1: *Encoder* completo com blocos de *Downsample* e *Upsample* e respectivas taxas de amostragem.

Fonte: do Autor.

- **Camada 5**: 4 subcamadas.

No total, o *encoder* possui 15 subcamadas distribuídas ao longo das suas 5 camadas.

As camadas do *encoder* operam em dados subamostrados em sua dimensão temporal. Desta forma, faz-se necessária a utilização de blocos de *Downsample* na entrada e de *Upsample* na saída, sendo que os fatores de subamostragem da entrada em cada camada são respectivamente 1, 2, 4, 8 e 2. Na saída de cada bloco, a subamostragem é revertida.

Na arquitetura *Zipformer* utilizada, um bloco *Embed* pré-processa o sinal de entrada, reduzindo a dimensão temporal pela metade. No final do *encoder*, após a quinta camada, um último bloco *Downsample* reduz a dimensão temporal novamente pela metade. Considerando T' a dimensão temporal original do sinal de entrada parametrizado $\mathbf{X}$, após o bloco *Embed* a dimensão temporal passa a ser $\frac{T'}{2}$. Ao final do *encoder*, após todas as subcamadas e o último bloco *Downsample*, a dimensão temporal é reduzida para $\frac{T'}{4} = T$, que é a dimensão temporal de $H(\mathbf{X})$. Portanto, o *encoder* realiza uma redução total de 4 vezes na dimensão temporal do sinal. Um diagrama geral do *encoder* mostrando o fator de subamostragem em cada nível é mostrado na Figura 3.1.

O *Downsample* é feito extraindo a média de dois *frames* consecutivos ponderadas por um parâmetro aprendido do modelo. O *Upsample* é feito através da repetição de *frames*. .

Cada subcamada Zipformer é composta por três operações *Feed-Forward*, duas operações convolucionais e duas operações de atenção *multi-head* sendo uma tradicional *MHA* (*Multi-Head Attention*) e uma que utiliza os pesos de atenção compartilhados da operação anterior

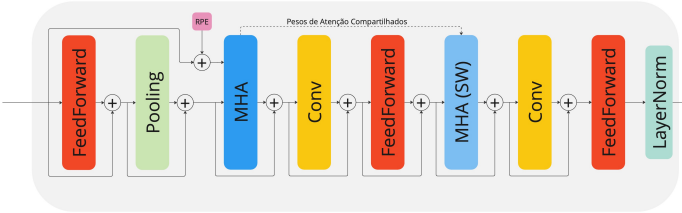

Figura 3.2: Arquitetura de uma Subcamada *Zipformer*.
Fonte: do Autor.

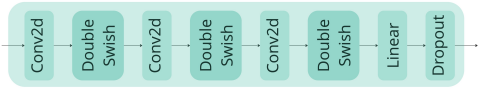

Figura 3.3: Bloco *Embed* em detalhe.
Fonte: do Autor.

MHA(SW) (*Shared Weight - SW*), além de *Pooling*, *Normalização* e uma camada de codificação de posição relativa (*Relative Positional Encoding - RPE*) [37] como mostrado na Figura 3.2.

Bloco Embed O bloco *Embed* da entrada do *encoder* é composto por uma série de operações de convolução que ao mesmo tempo que expandem os canais de entrada, reduzem a dimensão temporal. As estruturas convolucionais são formadas por três componentes principais que expandem os canais de entrada para 8, 32 e 128 respectivamente com ativações *DoubleSwish* entre elas e uma transformação linear na saída com *dropout* que fecha o bloco que é mostrado em detalhe na Figura 3.3.

Bloco FeedForward Cada bloco *FeedForward* é composto por duas transformações lineares com ativação *DoubleSwish* e *Dropout*. O diagrama em detalhe é mostrado na Figura 3.4.

Bloco Conv No bloco *Conv* as camadas de convolução são estruturadas em um padrão conhecido como convolução separável em profundidade (*depthwise separable convolution*). Neste padrão, uma operação

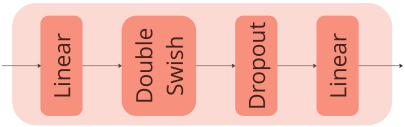

Figura 3.4: Bloco FeedForward em detalhe.
Fonte: do Autor.

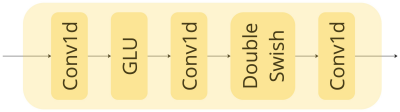

Figura 3.5: Bloco Conv em detalhe.
Fonte: do Autor.

é feita por cada canal de entrada e uma nova operação pontual agrupa as informações de todos os canais em uma única saída. As estruturas convolucionais são formadas pelos componentes abaixo e são mostradas na Figura 3.5.

- Convolução Pontual Inicial: Um bloco de entrada utiliza um *kernel* de tamanho 1x1 para expandir a entrada para 768 canais.

- Mecanismo GLU (*Gated Linear Unit*): O GLU divide o tensor resultante ao meio ao longo da dimensão dos canais, aplicando uma função de ativação sigmoide a uma metade e multiplicando elemento a elemento com a outra metade.

- Convolução *Depthwise*: Aplica a convolução separadamente a cada canal de entrada, usando um *kernel* de tamanho 31.

- Ativação dupla do tipo *Swish*.

- Convolução Pontual Final: Uma segunda convolução pontual (1x1) é aplicada para projetar o tensor de volta para a dimensão original.

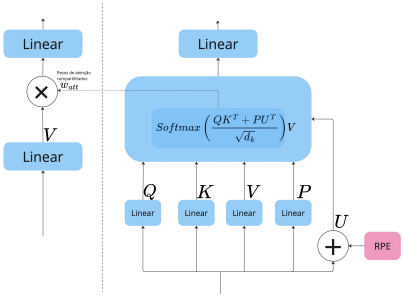

Figura 3.6: Bloco MHA(SW) (esq.) e MHA (dir.) em detalhes
Fonte: do Autor.

Blocos De Atenção O bloco de atenção é composto por uma operação de atenção Attention(Q, K, V, P) seguida de uma operação linear. A operação de atenção é *multi-head* e possui informação de posição relativa codificada adicionada através de U. Q, K, V e P são diferentes transformações da mesma entrada $\mathbf{X}$.

$$\text{Attention}(Q, K, V, P) = \text{softmax}\left(\frac{QK^T + PU^T}{\sqrt{d_k}}\right)V \tag{3.1}$$

No bloco com pesos de atenção compartilhados a operação de atenção Attention(w_{att}, V) é feita apenas com o operador V que é calculado da mesma forma. Os pesos de atenção w_{att}, da operação *softmax* do bloco MHA, são utilizados aqui. Também há uma operação linear na saída do bloco.

$$\text{Attention}(w_{\text{att}}, V) = \text{Linear}(w_{\text{att}}, V) \tag{3.2}$$

Bloco de Pooling O bloco de *Pooling* consiste em uma transformação linear da entrada que faz a média na dimensão temporal e projeta um vetor de dimensão idêntica a entrada.

Bloco de LayerNorm A normalização de camadas ajusta as ativações da rede neural com base na média e na variância dos valores ao longo de todas as características de uma entrada, porém utiliza um parâmetro aprendível como valor *eps*.

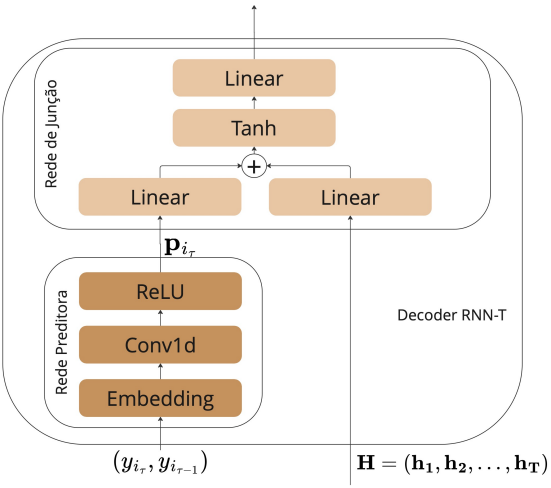

Figura 3.7: Decodificador RNN-T em detalhe
Fonte: do Autor.

Decoder RNN-T Podado sem Estado

O decodificador RNN-T é composto por dois blocos principais, mostrados na Figura 2.2: a rede de predição e a rede de junção. A rede de predição não possui nenhum fator de recorrência e é implementada utilizando apenas uma camada de *embedding* com um contexto de tamanho fixo de 2 símbolos e uma camada convolucional que agrega os *embeddings* dos símbolos em um único sinal de saída $\mathbf{p}$ de dimensão fixa.

A rede de junção processa a saída da rede de predição e do *encoder* através de uma projeção linear da soma das projeções de cada um desses fatores. O cálculo da função custo considera 5 símbolos em cada *frame* para efetuar o cálculo no espaço de busca podado. Um diagrama com detalhes das redes de predição e de junção são mostrados nas Figuras 3.7.

Decoder CTC

O decodificador CTC consiste em uma única camada de transformação linear que mapeia diretamente a representação intermediária $\mathbf{H}$

da saída do *encoder* para o espaço dos símbolos de saída na dimensão do tamanho do vocabulário. Na saída deste bloco é adicionada uma operação *softmax* para obter as probabilidades de cada símbolo do vocabulário para cada *frame* da entrada.

Vocabulário de Saída

O vocabulário de saída $\mathbf{Y}$, compartilhado entre os decodificadores CTC e RNN-T, é composto por $L' = 1000$ símbolos *word-pieces*, e são encontrados usando uma tokenização probabilística com regularização [38], que tem boa capacidade de lidar com a ocorrência de palavras fora do vocabulário. O treinamento do tokenizador foi realizado usando o corpus de texto das transcrições do conjunto de treinamento, contendo 18.099.331 palavras e 103.217.269 caracteres utilizando a ferramenta SentencePiece[2]. Por exemplo, a frase "agora terei todos os domingos livres" é tokenizada com 14 símbolos da seguinte maneira: [_agora, _te, re, i, _todos, _os, _do, mi, n, go, s, _li, v, res]. O caracter "_" é utilizado para adicionar a informação de início de palavra na tokenização. Dessa forma, ao agrupar todos os símbolos, basta remover este caractere e substituí-lo por um espaço para obter a frase novamente.

Entre os 1000 símbolos estão três símbolos especiais para diferentes blocos do modelo: (i) "*<sos>/<eos>*" representa início e fim de sequência para a rede preditora do RNN-T. (ii) "$\langle b \rangle$" utilizado tanto pelo CTC quanto pelo RNN-T no processo de alinhamento. (iii) "<unk>" para representar possíveis situações de fora de vocabulário nos conjuntos de validação e testes. Assim, a lista de símbolos de saída é dada por

$$\mathbf{\Sigma}' = \{\langle b \rangle, \text{<sos>/<eos>}, \text{<unk>}, word - pieces \ldots\}. \tag{3.3}$$

Conjunto de Dados de Treinamento

O conjunto de dados utilizado para treinamento do modelo compreende 2046 (2k) horas de áudio transcrito com trechos vozeados e em diferentes condições acústicas e com uma variedade de características linguísticas. Um histograma da duração dos trechos de áudio é mostrado na Figura 3.8. Um processo de *data augmentation* é feito adicio-

[2] https://github.com/google/sentencepiece

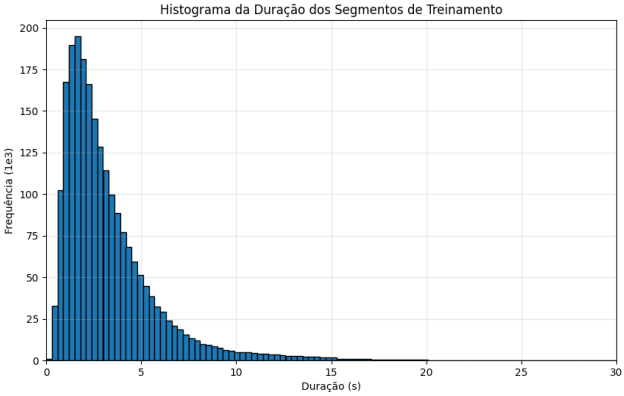

Figura 3.8: Histograma da duração dos trechos de áudio do conjunto de treinamento

Fonte: do Autor.

nando duas cópias do *dataset* com ±10% da velocidade de reprodução do áudio, respectivamente, além do uso de *SpecAugment* [39] um método de aumento de dados adicionados diretamente em $\mathbf{X}$ que insere distorções e mascaramentos na entrada.

O conjunto de validação utilizado para avaliar o desempenho do modelo durante o treinamento consiste em 9981 trechos de áudio totalizando 9 horas.

Treinamento

As dimensões utilizadas em cada camada e dimensões de arquitetura do *encoder* são mostradas na Tabela 3.1. As dimensões do decodificador RNN-T se resumem à dimensão da saída da rede preditora e das transformações internas da rede de junção que são 512 e 512, respectivamente. As funções custo foram ponderadas com pesos $\alpha = 0.3$ e $\beta = 0.3$.

O processo de treinamento do modelo *baseline* foi conduzido por 40 épocas. Os hiperparâmetros específicos incluíram um tamanho de *batch* de 600 segundos e uma taxa de aprendizado base de 0,05. Foi utilizado um esquema de *warmup* onde a taxa de aprendizado sobe

Camada	1	2	3	4	5
FeedForward	1024	1024	2048	2048	1024
Encoder	384	384	384	384	384
Atenção	192	192	192	192	192
Kernel Conv	31	31	31	31	31
Número de Cabeças	8	8	8	8	8
Número de Subcamadas	2	4	3	2	4
Fator de Downsample	1	2	4	8	2

Tabela 3.1: Dimensões do modelo *baseline*
Fonte: do Autor.

linearmente por 3000 passos e após cai exponencialmente seguindo o *scheduler* Eden [31]. Para otimizar o uso de recursos computacionais e acelerar o treinamento, foi empregada a técnica de precisão mista [40]. Uma GPU NVidia RTX 4090 com 24 GB de VRAM foi utilizada no treinamento. Um resumo desses dados é mostrado na Tabela 3.2.

Parâmetro	**Valor**
Épocas	40
Otimizador	ScaledAdam [31]
Scheduler	Eden [31]
Warmup	3000 passos
Função de custo	CTC (0,3) + RNN-T (0,3)
Batch size	600 segundos de áudio

Tabela 3.2: Parâmetros de treinamento
Fonte: do Autor.

As funções custo de treinamento e validação CTC e RNN-T são mostradas nas Figuras 3.9 e 3.10 respectivamente.

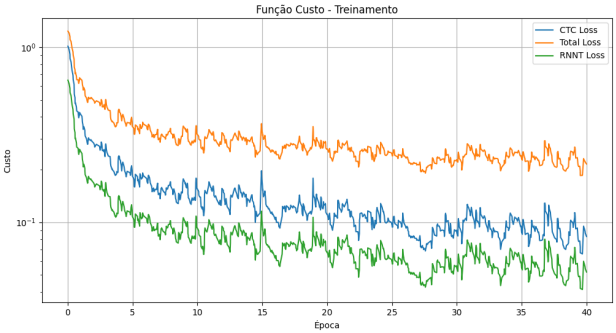

Figura 3.9: Funções Custo de treinamento do modelo *baseline*.
Fonte: do Autor.

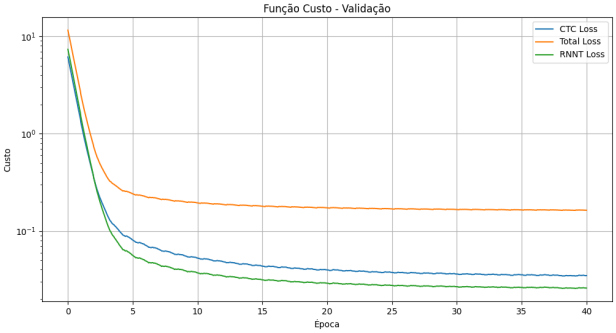

Figura 3.10: Funções Custo de validação do modelo *baseline*.
Fonte: do Autor.

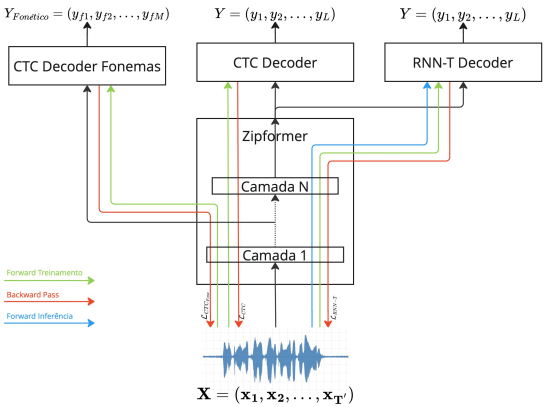

Figura 3.11: Arquitetura Investigada com CTC Fonético conectado a uma saída intermediária do *encoder*

Fonte: do Autor.

3.2 Arquitetura Investigada

A arquitetura investigada neste trabalho é uma extensão do modelo *baseline* descrito, incorporando princípios de aprendizado hierárquico multitarefa. A modificação consiste na adição de um decodificador CTC com símbolos que representam a transcrição fonética canônica do áudio. Isso está conectado em camadas intermediárias do *encoder Zipformer*, como ilustrado na Figura 3.11. Durante o treinamento, além do *decoder* CTC otimiza conjuntamente o *encoder* com RNN-T; um novo *decoder* CTC otimiza parcialmente o *encoder* enquanto usa a informação fonética fornecida.

O decodificador CTC fonético é conectado em uma camada intermediária do *encoder Zipformer*. Na intenção de observar o comportamento entre as camadas internas, foram realizados experimentos em todas elas para avaliar qual configuração oferece o melhor desempenho do sistema quando observada a taxa de erro de palavras. A inferência, mostrada no fluxo em azul na Figura 3.11, é realizada apenas na rede RNN-T.

Neste trabalho foi utilizada uma representação fonética composta por 51 fonemas do português brasileiro. As transcrições fonéticas foram obtidas através de uma ferramenta de conversão grafema-fonema [41] que faz a conversão utilizando uma série de regras fonológicas. Além

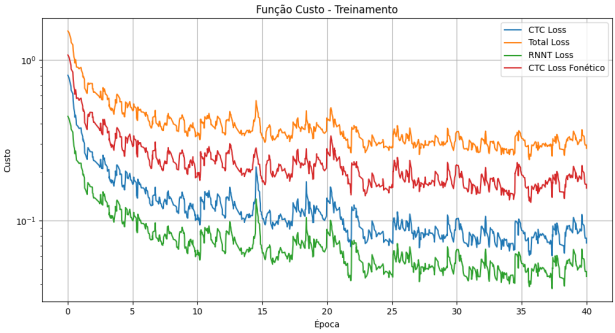

Figura 3.12: Funções custo de treinamento do modelo treinado na segunda camada

Fonte: do Autor.

dos símbolos fonéticos, há o símbolo $\langle b \rangle$ para o alinhamento CTC. As sequências $\mathbf{Y}_{Fonético}$ que representam os sinais de saída são compostas pela lista dos fonemas da conversão grafema-fonema; dessa forma, não há informações sobre separação de palavras como os espaços codificados pelo modelo *word-piece*. A frase usada para exemplificar a tokenização *word-piece*, na sequência fonética, é representada com a seguinte sequência de símbolos: [a, g, ó, r, a, t, e, r, ê, y, t, ô, d, ō, z, ō, z, d, o, m, ĩ, g, ō, z, l, í, v, r, ē, S].

A função de custo total do sistema foi modificada para incorporar o componente fonético, resultando na seguinte formulação:

$$\mathcal{L} = \alpha\mathcal{L}_{\text{CTC}} + \beta\mathcal{L}_{\text{RNN-T}} + \gamma\mathcal{L}_{\text{CTC}_{\text{fone}}}. \tag{3.4}$$

onde $\alpha = \beta = \gamma = 0.3$, atribuindo pesos iguais para cada componente do sistema. Todos os outros parâmetros de treinamento são idênticos ao modelo *baseline*. A Figura 3.12 apresenta as curvas de treinamento de cada bloco do modelo avaliado mostrando o comportamento do treinamento. A Figura 3.13 apresenta as funções custo apenas do CTC Fonético no conjunto de validação para todas as camadas.

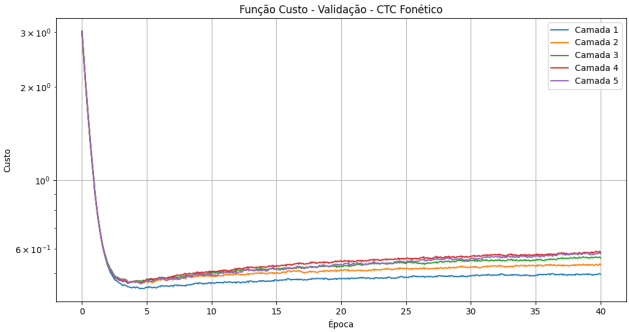

Figura 3.13: Função custo CTC Fonético no conjunto de validação
Fonte: do Autor.

3.2.1 Experimentos e Teste

Foram realizados cinco experimentos distintos, cada um correspondendo a uma das camadas do *encoder*. Em cada experimento, que corresponde a um treinamento completo, o *decoder* CTC fonético foi conectado a uma camada diferente. Sendo a camada 1 a mais próxima da entrada e a camada 5 a mais próxima da saída. Cada configuração foi treinada e avaliada separadamente, mantendo todos os outros parâmetros do modelo constantes. Os conjuntos de dados são os mesmos. A arquitetura *baseline*[3] e a arquitetura avaliada [4] estão disponíveis online.

Conjuntos de Teste

Para avaliar o desempenho do modelo em diferentes cenários, são utilizados quatro conjuntos de teste públicos disponíveis *online*:

MLS *Multilingual LibriSpeech* [42] é um grande *dataset* apropriado para tarefas relacionadas a fala. O *dataset* é composto de áudios livros de domínio público. O subconjunto de teste em português contém 851 segmentos e tem 3h44 de duração.

[3] https://github.com/k2-fsa/icefall
[4] https://github.com/alucassch/eel7806_tcc

TedX *Multilingual TEDx* [43] é um corpus multilíngua de reconhecimento e tradução de fala projetado para facilitar o treinamento de modelos de Reconhecimento Automático de Fala e Tradução de Fala em diversos idiomas. O corpus é composto por gravações de áudio e transcrições de palestras. O subconjunto de teste em português possui 1007 segmentos e tem 1h48 de duração.

CORAA ASR V1 CORAA ASR [44] é um *dataset* para Reconhecimento Automático de Fala em português brasileiro. Ele contém 290 horas de áudios e suas respectivas transcrições, totalizando mais de 400 mil segmentos. A parte de teste contém 7521 segmentos e tem 6h de duração.

GNeutralSpeech GNeutralSpeech [45] é uma base de dados desenvolvida para síntese de fala em português brasileiro. Ele consiste em aproximadamente 20 h de gravações de um único locutor masculino, lendo sentenças com voz neutra. O corpus é composto por gravações de alta qualidade.

Para avaliar o sistema foi utilizada a métrica WER (*Word Error Rate*), que mede a taxa de erro entre uma transcrição de referência e uma hipótese de transcrição. O WER é calculado através da distância de Levenshtein [46], ou distância de edição, que contabiliza o número mínimo de operações de edição necessárias para transformar a sequência de palavras hipotética na sequência de referência, considerando três tipos de erros: **I**nserções (palavras adicionadas), **D**eleções (palavras removidas) e **S**ubstituições (palavras trocadas) normalizados pelo número de palavras da referência. O cálculo da taxa de erro de palavras é feito somando as operações de edição descritas acima e normalizando pelo tamanho da referência na forma:

$$\text{WER} = 100\% \times \frac{I + D + S}{N}. \tag{3.5}$$

Para realizar a decodificação, o método de busca gananciosa (*greedy search*) foi utilizado como uma abordagem simplificada de decodificação, onde, em cada passo de tempo, é selecionado o símbolo com maior probabilidade na distribuição de saída, descartando os símbolos $\langle b \rangle$. Isso é feito apenas no decodificador RNN-T. Os decodificadores CTC

não são usados na inferência.

3.3 Resultados

Os experimentos realizados com a arquitetura investigada revelaram padrões consistentes sobre a incorporação de informações fonéticas nas diferentes camadas do *encoder*. A Figura 3.14 apresenta os resultados gerais obtidos para os quatro conjuntos de teste.

O dataset MLS apresentou a melhoria mais expressiva, com o WER reduzindo de 10,85% no *baseline* para 10,23% na segunda camada, representando uma melhoria relativa de 5,71%. Para o TEDx, observou-se uma melhoria moderada, com o WER diminuindo de 26,40% no *baseline* para 25,77% na segunda camada, resultando em uma melhoria relativa de 2,39%. No caso do CORAA, a melhoria foi mais sutil, com o WER reduzindo de 17,10% para 16,80%, correspondendo a uma melhoria relativa de 1,75%. O *dataset* GNEUTRAL, apesar de apresentar os menores valores absolutos de WER, mostrou uma melhoria relativa considerável de 3,47%, com o WER diminuindo de 3,75% para 3,62%. Estes dados estão sumarizados na Tabela 3.3

Dataset	Camada 1	Camada 2	Camada 3	Camada 4	Camada 5
TEDx	+0,11%	**-2,39**%	-1,17%	-1,70%	-1,85%
CORAA	+8,07%	**-1,75**%	+5,44%	+4,33%	-0,35%
MLS	+1,01%	**-5,71**%	-3,69%	-2,49%	-0,37%
GNeutral	+5,60%	**-3,47**%	-3,47%	+8,27%	+0,80%

Tabela 3.3: Melhora relativa do WER (%) em relação ao *baseline* para cada conjunto de teste e camada

Fonte: do Autor.

Os resultados obtidos nos diferentes *datasets* avaliados apontam para um padrão consistente, no qual a incorporação de informações fonéticas na segunda camada do *encoder* avaliado levou a uma melhora maior na taxa de erro de palavra do sistema como um todo, independentemente das características de cada teste. Observou-se também que a incorporação dessas informações em camadas superiores à segunda resultou, em alguns casos, em um aumento do WER. Isso também é observado de forma ainda mais intensa na primeira camada.

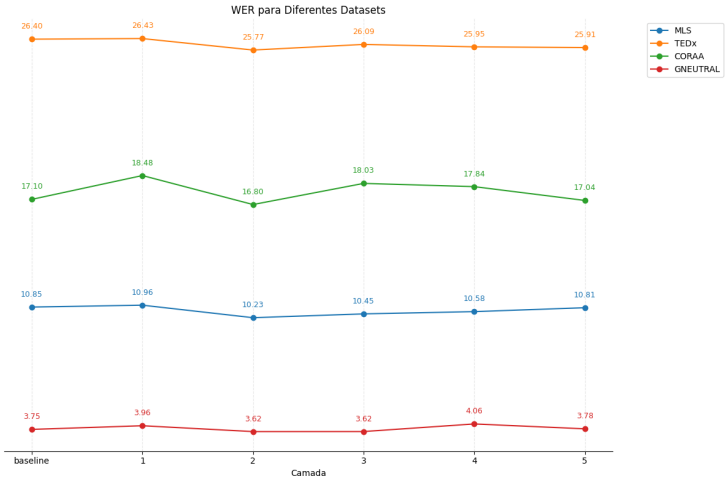

Figura 3.14: Resultados mostrando WER comparando o modelo *baseline* com os modelos com o CTC fonético em diferentes camadas (lambda) para três *datasets* públicos

Fonte: do Autor.

A Figura 3.15 apresenta um gráfico de caixas que ilustra a mudança relativa na taxa de erro de palavras em relação ao *baseline*. Cada caixa representa a distribuição dos resultados para uma determinada camada. A linha laranja no centro de cada caixa indica a mediana da mudança relativa do WER para aquela camada. A caixa azul representa o intervalo interquartil, que contém 50% dos valores centrais. As linhas pretas se estendem até os valores máximos e mínimos.

A segunda camada se destaca por apresentar a menor mediana de mudança relativa do WER. Além disso, a caixa da segunda camada é relativamente compacta e está inteiramente abaixo de zero, o que indica uma redução consistente da taxa de erro em todos os testes.

Entre a terceira e quinta camadas, observa-se uma tendência de aumento na mediana e um aumento na variabilidade dos resultados, evidenciado pelo alargamento das caixas. Isso sugere que a incorporação de informações fonéticas em camadas mais próximas da saída pode levar a benefícios menos consistentes e, em alguns casos, até mesmo prejudicar o desempenho do sistema. A primeira camada apresenta a maior variabilidade entre os testes e demonstra uma consistente piora

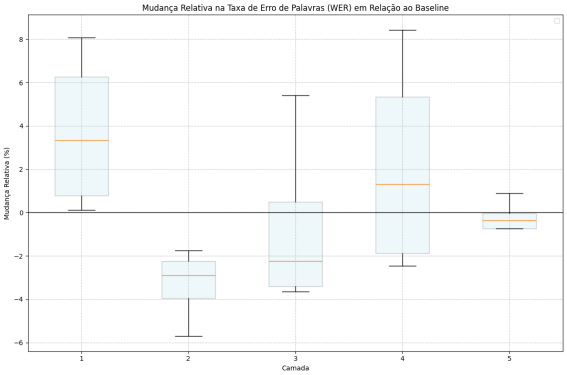

Figura 3.15: Gráfico de caixas ilustrando a melhoria relativa do WER para cada camada

Fonte: do Autor.

nos resultados.

É interessante destacar que ocorre um *overfitting* no conjunto de validação para o bloco do CTC fonético. Isso pode ser um indicativo de que existe espaço para melhoria na arquitetura investigada. No entanto, como a tarefa de predição do alinhamento fonético não é utilizada no processo de inferência, podemos considerar que a ocorrência deste *overfitting* no processo de treinamento não causa impacto relevante no desempenho do sistema.

CAPÍTULO 4

Conclusão

Este trabalho apresentou uma investigação sobre a incorporação de informações fonéticas em uma arquitetura moderna de reconhecimento automático de fala utilizando aprendizado hierárquico multitarefa, com foco específico na otimização de um encoder do tipo Zipformer. A partir de um modelo baseline composto por este encoder e um decoder RNN-T treinado conjuntamente com um decoder CTC, foi proposta uma arquitetura estendida que integra informações fonéticas canônicas nas camadas intermediárias do encoder através de um decodificador CTC adicional.

Os experimentos conduzidos em diversos conjuntos de teste revelaram uma melhora consistente na taxa de erro de palavras quando as informações fonéticas são incorporadas na segunda camada do encoder. Esse resultado sugere que, para esta arquitetura específica, a segunda camada representa o melhor ponto para a integração da transcrição fonética. No entanto, a incorporação de informações fonéticas em camadas superiores à segunda camada, além da primeira camada, resultou, em alguns casos, em um aumento da taxa de erro de palavras, indicando que, à medida que o encoder Zipformer progride para representações mais abstratas em camadas superiores, a integração de

informações fonéticas explícitas se torna menos benéfica.

A análise das curvas de treinamento revelou um overfitting na função custo CTC fonética. Esse overfitting pode ser atribuído a dois fatores principais: (1) a especialização excessiva do modelo nas informações fonéticas específicas fornecidas do conjunto de treinamento; e (2) a falta de regularização na camada correspondente.

Os resultados obtidos neste trabalho abrem caminho para explorar outras representações auxiliares e tarefas que podem beneficiar o treinamento de sistemas ASR com a arquitetura de encoder Zipformer. Futuros trabalhos podem investigar se os padrões observados no Zipformer se mantêm em outras arquiteturas de encoder similares. Além disso, é interessante explorar o uso de diferentes representações fonéticas, como o uso de variantes específicas de palavras, e outras tarefas auxiliares, como a predição de limites de palavras. Essas investigações adicionais podem contribuir para o desenvolvimento de sistemas ASR mais robustos, precisos e adaptáveis a diferentes domínios.

Trabalhos futuros

Com base nos resultados e limitações deste estudo, são sugeridos os seguintes trabalhos futuros:

- Extrapolar o uso das informações fonéticas canônicas através do uso do alinhamento forçado fornecido por um modelo tradicional com o uso de diversas variantes fonéticas no lugar da transcrição canônica. Embora isso adicione complexidade, pois a escolha das variantes influenciaria o processo, é possível que um alinhamento mais representativo do sinal de áudio resulte em um melhor aproveitamento dessa informação pelo modelo.

- Explorar diferentes formas de representar a transcrição fonética utilizando técnicas como *word-pieces*, tratando os fonemas como um texto similar à transcrição ortográfica, Isso seria similar ao uso de tri-fones, que são grupos de fonemas consecutivos, em sistemas tradicionais.

- Realizar uma engenharia de atributos mais aprofundada, testando diferentes escolhas de símbolos fonéticos e seu impacto no desempenho do sistema.

- Aprimorar o processo de treinamento através do teste de diferentes pesos para cada função custo, diferentes taxas de aprendizado e outras técnicas de otimização.

Referências bibliográficas

[1] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 1764–1772, Bejing, China, 22–24 Jun 2014. PMLR.

[2] Gustavo López, Luis Quesada, and Luis A. Guerrero. Alexa vs. siri vs. cortana vs. google assistant: A comparison of speech-based natural user interfaces. In Isabel L. Nunes, editor, *Advances in Human Factors and Systems Interaction*, pages 241–250, Cham, 2018. Springer International Publishing.

[3] Zhang Hua and Wei Lieh Ng. Speech recognition interface design for in-vehicle system. In *Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '10, page 29–33, New York, NY, USA, 2010. Association for Computing Machinery.

[4] Tobias Hodgson and Enrico Coiera. Risks and benefits of speech recognition for clinical documentation: A systematic review. *Journal of the American Medical Informatics Association*, 23, 11 2015.

[5] Jing Peng, Yucheng Wang, Yu Xi, Xu Li, Xizhuo Zhang, and Kai Yu. A survey on speech large language models. Technical report, Shanghai Jiao Tong University, Shanghai, China, 2023. MoE Key Lab of Artificial Intelligence, AI Institute and X-LANCE Lab, Department of Computer Science and Engineering.

[6] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[7] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304, January 2007.

[8] Mbarki Aymen, Ammari Abdelaziz, Sghaier Halim, and Hassen Maaref. Hidden markov models for automatic speech recognition. In *2011 International Conference on Communications, Computing and Control Applications (CCCA)*, pages 1–6, 2011.

[9] Daniel Jurafsky and James H. Martin. *N-gram Language Models*, chapter 3. 3 edition, 2024. Online manuscript released August 20, 2024.

[10] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[11] Rohit Prabhavalkar, Takaaki Hori, Tara N. Sainath, Ralf Schlüter, and Shinji Watanabe. End-to-end speech recognition: A survey. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:325–351, 2024.

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[13] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao,

Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[14] Yifan Peng, Siddharth Dalmia, Ian Lane, and Shinji Watanabe. Branchformer: Parallel MLP-attention architectures to capture local and global context for speech recognition and understanding. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17627–17643. PMLR, 17–23 Jul 2022.

[15] Kwangyoun Kim, Felix Wu, Yifan Peng, Jing Pan, Prashant Sridhar, Kyu J. Han, and Shinji Watanabe. E-branchformer: Branchformer with enhanced merging for speech recognition. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 84–91, 2023.

[16] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 29:3451–3460, October 2021.

[17] Zengwei Yao, Wei Kang, Xiaoyu Yang, Fangjun Kuang, Liyong Guo, Han Zhu, Zengrui Jin, Zhaoqing Li, Long Lin, and Daniel Povey. CR-CTC: Consistency regularization on CTC for improved speech recognition. `https://arxiv.org/abs/2410.05101`, 2024. Preprint. Under review.

[18] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech 2020*, pages 5036–5040, 2020.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[20] Kalpesh Krishna, Shubham Toshniwal, and Karen Livescu. Hierarchical multitask learning for CTC-based speech recognition. Technical report, University of Massachusetts Amherst, USA, 2024.

[21] Chunxi Liu, Frank Zhang, Duc Le, Suyoun Kim, Yatharth Saraf, and Geoffrey Zweig. Improving RNN transducer based ASR with auxiliary tasks. In *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021.

[22] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint CTC-attention based end-to-end speech recognition using multi-task learning, 2017. MERL CMU.

[23] Alex Graves. Sequence transduction with recurrent neural networks. In *International Conference on Machine Learning (ICML) Workshop on Representation Learning*, 2012.

[24] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 369–376, New York, NY, USA, 2006. Association for Computing Machinery.

[25] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[26] Mingkun Huang. warp-transducer. `https://github.com/HawkAaron/warp-transducer`, 2018.

[27] Ivan Sorokin. Cuda-warp rnn-transducer. `https://github.com/1ytic/warp-rnnt`, 2019.

[28] k2 fsa. k2. `https://github.com/k2-fsa/k2`, 2020.

[29] Fangjun Kuang, Liyong Guo, Wei Kang, Long Lin, Mingshuang Luo, Zengwei Yao, and Daniel Povey. Pruned rnn-t for fast, memory-efficient asr training. In *Interspeech 2022*, pages 2068–2072, 2022.

[30] Mohammad Reza Ghodsi, Xiaofeng Liu, James Alexander Apfel, Rodrigo Cabrera, and Eugene Weinstein. Rnn-transducer with stateless prediction network. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7049–7053, 2020.

[31] Zengwei Yao, Liyong Guo, Xiaoyu Yang, Wei Kang, Fangjun Kuang, Yifan Yang, Zengrui Jin, Long Lin, and Daniel Povey. Zipformer: A faster and better encoder for automatic speech recognition. In *The Twelfth International Conference on Learning Representations*, 2024.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[33] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pages 818–833. Springer, 2014.

[34] Yonatan Belinkov and James Glass. Analyzing hidden representations in end-to-end automatic speech recognition systems. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[35] Pegah Ghahremani, Bagher BabaAli, Daniel Povey, Korbinian Riedhammer, Jan Trmal, and Sanjeev Khudanpur. A pitch extraction algorithm tuned for automatic speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2494–2498, 2014.

[36] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011. IEEE Catalog No.: CFP11SRW-USB.

[37] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics.

[38] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[39] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019*, interspeech$_2$019.$ISCA, September$2019.

[40] Sharan Narang, Gregory Diamos, Erich Elsen, Paulius Micikevicius, Jonah Alben, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations (ICLR)*, 2018. Baidu Research NVIDIA.

[41] Izabel Christine Seara, Fernando Santana Pacheco, Sandra Ghizoni Kafka, Rui Seara, Jr., and Rui Seara. Morphosyntactic parser for Brazilian Portuguese: Methodology for development and assessment. In *Proceedings of the 9th International Conference on Computational Processing of the Portuguese Language (PROPOR)*, pages 1–6, Porto

Alegre, Brazil, 2010. Pontifícia Universidade Católica do Rio Grande do Sul.

[42] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. Mls: A large-scale multilingual dataset for speech research. *ArXiv*, abs/2012.03411, 2020.

[43] Elizabeth Salesky, Matthew Wiesner, Jacob Bremerman, Roldano Cattoni, Matteo Negri, Marco Turchi, Douglas W. Oard, and Matt Post. Multilingual tedx corpus for speech recognition and translation. In *Proceedings of Interspeech*, 2021.

[44] Arnaldo Candido Junior, Edresson Casanova, Anderson Soares, Frederico Santos de Oliveira, Lucas Oliveira, Ricardo Corso Fernandes Junior, Daniel Peixoto Pinto da Silva, Fernando Gorgulho Fayet, Bruno Baldissera Carlotto, Lucas Rafael Stefanel Gris, et al. Coraa asr: a large corpus of spontaneous and prepared speech manually validated for speech recognition in brazilian portuguese. *Language Resources and Evaluation*, pages 1–33, 2022.

[45] Media Technology Lab. Gneutralspeech. `https://www.kaggle.com/datasets/mediatechlab/gneutralspeech`, 2023.

[46] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.