

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA
CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA**

RAQUEL DUTRA KOTZIAS

**DESENVOLVIMENTO DE UM SISTEMA DE KARAOKÊ BASEADO EM
INTELIGÊNCIA ARTIFICIAL: separação automática de voz e geração
de legendas utilizando modelos pré-treinados**

FLORIANÓPOLIS, 2026.

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA
CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA**

RAQUEL DUTRA KOTZIAS

**DESENVOLVIMENTO DE UM SISTEMA DE KARAOKÊ BASEADO EM
INTELIGÊNCIA ARTIFICIAL: separação automática de voz e geração
de legendas utilizando modelos pré-treinados**

Trabalho de Conclusão de Curso submetido
ao Instituto Federal de Educação, Ciência e
Tecnologia de Santa Catarina como parte
dos requisitos para obtenção do título de
Engenheiro em Eletrônica.

Orientador:
Prof. Robinson Pizzio, Dr.

FLORIANÓPOLIS, 2026.

Ficha de identificação da obra elaborada pelo autor.

Kotzias, Raquel Dutra
DESENVOLVIMENTO DE UM SISTEMA DE KARAOKÊ BASEADO EM INTELIGÊNCIA ARTIFICIAL: separação automática de voz e geração de legendas utilizando modelos pré-treinados
/ Raquel Dutra Kotzias; orientação de Robinson Pizzio.
- Florianópolis, SC, 2026.

74 p.
Trabalho de Conclusão de Curso (TCC) - Instituto Federal de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Eletrônica. Departamento Acadêmico de Eletrônica.
Inclui Referências.

1. Separação de Fontes Musicais. 2. Inteligência Artificial. 3. Processamento Digital de Sinais. 4. Mel-Band RoFormer. 5. Whisper. I. Pizzio, Robinson. II. Instituto Federal de Santa Catarina. III. DESENVOLVIMENTO DE UM SISTEMA DE KARAOKÊ BASEADO EM INTELIGÊNCIA ARTIFICIAL.

DESENVOLVIMENTO DE UM SISTEMA DE KARAOKÊ BASEADO EM INTELIGÊNCIA ARTIFICIAL: separação automática de voz e geração de legendas utilizando modelos pré-treinados

RAQUEL DUTRA KOTZIAS

Este trabalho foi julgado adequado para obtenção do título de Engenheiro em Eletrônica e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 10 de agosto, 2026.

Banca Examinadora:

Documento assinado digitalmente
 **ROBINSON PIZZIO**
Data: 11/08/2026 23:18:14-0300
Verifique em <https://validar.iti.gov.br>

Robinson Pizzio, Dr.

Documento assinado digitalmente
 **HUGO MARCONDES**
Data: 12/08/2026 09:51:23-0300
Verifique em <https://validar.iti.gov.br>

Hugo Marcondes, M.Sc.
IFSC

Documento assinado digitalmente
 **LUIS CARLOS MARTINHAGO SCHLICHTING**
Data: 12/08/2026 14:16:06-0300
Verifique em <https://validar.iti.gov.br>

Luis Carlos M. Schlichting, Dr.
IFSC

Dedico este trabalho a todos
os professores que muito me ensinaram,
a minha família e amigos que sempre me impulsionaram,
e ao universo que me colocou neste caminho.

AGRADECIMENTOS

Agradeço a todas as pessoas que me ajudaram e fizeram parte da minha vida no decorrer destes anos: minha família, todos os professores, em especial ao professor Prof. Robinson Pizzio por ter aceitado ser meu orientador no trabalho de conclusão de curso e não ter me deixado desistir. Agradeço aos amigos. Agradeço por ter todos os atributos necessários para cumprir esta jornada. Enfim, agradeço a todos aqueles que passaram de alguma forma por minha vida e de alguma maneira me ajudaram a chegar até este momento.

“Don't find fault, find a remedy;
anybody can complain”.

Henry Ford

RESUMO

A convergência entre o Processamento Digital de Sinais e o aprendizado profundo tem possibilitado a automação de tarefas complexas na engenharia de áudio, como o desenvolvimento de sistemas de karaokê. Este trabalho aborda a implementação de um fluxo automatizado para a separação de fontes musicais e a geração de legendas sincronizadas a partir de gravações comerciais. O modelo Mel-Band RoFormer é utilizado para a decompor o sinal em faixas vocal e instrumental, empregando mecanismos de atenção hierárquica e a escala Mel para garantir fidelidade perceptiva. Para a transcrição e sincronização, aplica-se o modelo *Whisper*. Um aspecto central do desenvolvimento envolveu a configuração de ambientes computacionais distintos, a etapa de separação de fontes utilizou aceleração por hardware (GPU) via CUDA para otimizar o tempo de inferência, enquanto a geração automática de legendas foi executada exclusivamente em CPU. Os resultados indicam um ganho de desempenho (*speedup*) médio de 2,68 vezes na etapa de separação ao utilizar a GPU. As legendas geradas apresentaram taxas de erro (WER e CER) variáveis conforme a complexidade da faixa, mas mantiveram a legibilidade necessária para a aplicação de karaokê na maioria dos cenários. A integração de modelos pré-treinados de aprendizado profundo é uma alternativa viável para a automação da produção de conteúdos musicais, reduzindo drasticamente o esforço manual tradicional, embora o tempo de processamento ainda represente um desafio para aplicações voltadas ao tempo real.

Palavras-chave: Separação de Fontes Musicais. Inteligência Artificial. Processamento Digital de Sinais. Mel-Band RoFormer. Whisper.

ABSTRACT

The convergence between Digital Signal Processing and deep learning has enabled the automation of complex tasks in audio engineering, such as the development of karaoke systems. This work addresses the implementation of an automated workflow for music source separation and synchronized lyrics generation from commercial recordings. The Mel-Band RoFormer model is used to decompose the signal into vocal and instrumental tracks, employing hierarchical attention mechanisms and the Mel scale to ensure perceptual fidelity. For transcription and synchronization, the Whisper model is applied. A central aspect of the development involved the configuration of distinct computational environments, the source separation stage utilized hardware acceleration (GPU) via CUDA to optimize inference time, while the automated lyrics generation was executed exclusively on CPU. The results indicate an average performance gain (speedup) of 2.68 times in the separation stage when using the GPU. The generated lyrics presented varying error rates (WER and CER) depending on the track's complexity, yet maintained the necessary legibility for karaoke application in most scenarios. The integration of pre-trained deep learning models is a viable alternative for automating the production of musical content, drastically reducing traditional manual effort, although processing time still represents a challenge for real-time applications.

Keywords: Music Source Separation. Artificial Intelligence. Digital Signal Processing. Mel-Band RoFormer. Whisper.

LISTA DE FIGURAS

Figura 1 - Exemplo de Espectrograma.....	22
Figura 2 - Comparação das funções janela no domínio do tempo (azul) e suas respectivas respostas em frequência (vermelho), ilustrando o impacto da geometria da janela na resolução e no vazamento espectral	23
Figura 3 - Comparação da distribuição dos filtros do banco Mel nas escalas linear e Mel	26
Figura 4 - Comparação entre os espectrogramas STFT e Mel	27
Figura 5 - Modelo não-linear de neurônio	28
Figura 6 - Grafo arquitetural de um perceptron de múltiplas camadas com N camadas ocultas.....	30
Figura 7 - Função Sigmoid	30
Figura 8 - Função da Tangente Hiperbólica	31
Figura 9 - Função ReLU	31
Figura 10 - Estrutura geral da arquitetura Transformer	34
Figura 11 - Funcionamento do mecanismo de atenção escalada por produto interno (Scaled Dot-Product Attention).....	35
Figura 12 - Incorporação da codificação posicional à representação de entrada do Transformer.....	36
Figura 13 - Aplicação do Rotary Positional Embedding (RoPE) aos vetores Query e Key	37
Figura 14 - Funcionamento do BS-TransFormer.....	38
Figura 15 - Funcionamento do Mel-RoFormer	41
Figura 16 - Visão Geral do Whisper	44
Figura 17 - Fluxo de configuração e execução da inferência para geração das faixas vocal e instrumental utilizando a arquitetura Mel-Band RoFormer	52
Figura 18 - Fluxograma do script de geração de legendas com Whisper	55
Figura 19 - Comparação das formas de onda após a separação de fontes musicais	60
Figura 20 - Comparação espectral dos sinais obtidos após a separação de fontes..	60
Figura 21 - Comparação espectral dos sinais obtidos após a separação de fontes..	62
Figura 22 - Exemplo de legenda gerada automaticamente no formato SRT.....	64

Figura 23 - Exemplo da sincronização entre a legenda gerada automaticamente e a reprodução da faixa instrumental	65
Figura 24 - Taxas de erro das transcrições geradas pelo Whisper	67

SUMÁRIO

1	INTRODUÇÃO	14
1.1	Justificativa.....	15
1.2	Definição do Problema.....	15
1.3	Objetivo Geral	16
1.4	Objetivos Específicos	16
1.5	Estrutura do Trabalho	16
2	FUNDAMENTAÇÃO TEÓRICA	18
2.1	Separação de fontes musicais	19
2.1.1	Sinais de Tempo Contínuo vs. Tempo Discreto	19
2.1.2	Transformada de Fourier de Tempo Curto	19
2.1.2.1	<i>Princípio de Funcionamento da STFT</i>	<i>20</i>
2.1.2.2	<i>Processo de Segmentação do Sinal.....</i>	<i>21</i>
2.1.2.3	<i>Funções Janela</i>	<i>22</i>
2.1.2.4	<i>Resolução Tempo-Frequência</i>	<i>24</i>
2.1.3	Escala Mel	24
2.1.3.1	<i>Princípio de Funcionamento</i>	<i>24</i>
2.1.3.2	<i>Banco de filtros Mel</i>	<i>25</i>
2.1.3.3	<i>Aplicação nos modelos de Inteligência Artificial</i>	<i>27</i>
2.1.4	Redes Neurais Perceptrons de Múltiplas Camadas	28
2.1.4.1	<i>Modelo Matemático do Neurônio Artificial</i>	<i>29</i>
2.1.4.2	<i>Estrutura de uma Rede MLP</i>	<i>29</i>
2.1.4.3	<i>Funções de Ativação</i>	<i>30</i>
2.1.4.4	<i>Processo de Treinamento</i>	<i>31</i>
2.1.5	Transformers	32
2.1.5.1	<i>Estrutura Geral do Transformer.....</i>	<i>33</i>
2.1.5.2	<i>Mecanismo de Atenção</i>	<i>34</i>
2.1.5.3	<i>Codificação Posicional</i>	<i>35</i>
2.1.5.4	<i>Rotary Positional Embedding (RoPE).....</i>	<i>36</i>
2.1.6	Band-Split RoPE Transformer	38
2.1.6.1	<i>Arquitetura</i>	<i>38</i>
2.1.6.2	<i>Módulo de Divisão de Banda (Band-Split Module)</i>	<i>39</i>
2.1.6.3	<i>Blocos RoFormer.....</i>	<i>39</i>
2.1.6.4	<i>Estimativa de Máscara Multi-banda e Reconstrução.....</i>	<i>39</i>
2.1.7	Mel-Band RoFormer	40
2.1.7.1	<i>Arquitetura</i>	<i>40</i>
2.1.7.2	<i>Fluxo de processamento</i>	<i>41</i>
2.1.7.2	<i>Reconstrução do sinal</i>	<i>42</i>
2.1.7.3	<i>Desempenho</i>	<i>42</i>
2.1.8	<i>Signal-to-Distorion Ratio (SDR).....</i>	<i>42</i>
2.2	Geração Automática de Legenda.....	43
2.2.1	Whisper	43
2.2.2	Distância de Levenshtein	45
2.2.3	Word Error Rate (WER).....	45
2.2.4	Character Error Rate (CER)	46

3	METODOLOGIA	48
3.1	Separação de fontes musicais	48
3.1.1	Criação do ambiente virtual para a separação de fontes musicais	49
3.1.2	Script de configuração da inferência	51
3.2	Geração automática de legendas.....	52
3.2.1	Configuração do ambiente para o Whisper	53
3.2.2	Desenvolvimento do script para geração das legendas	54
4	APRESENTAÇÃO DOS RESULTADOS	58
4.1	Ambiente Experimental.....	58
4.2	Separação de Fontes Musicais	59
4.2.1	Resultados da separação	59
4.2.2	Desempenho computacional.....	61
4.2.2.1	<i>Resultados obtidos.....</i>	<i>61</i>
4.3	Geração Automática de Legendas.....	64
4.3.1	Resultados da geração automática de legendas.....	64
4.3.2	Avaliação dos resultados.....	65
5	CONSIDERAÇÕES FINAIS	68
5.1	Sugestões para trabalhos futuros	68
	REFERÊNCIAS.....	70
	APÊNDICE A - MELROFORMER_CONFIG.PY	73
	APÊNDICE B - GENERATESUBTITLES.PY.....	74

1 INTRODUÇÃO

A busca pelo prazer e bem-estar são os motores que impulsionam a existência do entretenimento. Apesar de que na aparência ser algo supérfluo, o entretenimento tem um papel importante para a saúde física e mental das pessoas (Peixoto, 2020). O entretenimento abrange diversas áreas de atuação do ser humano. A música, por exemplo, é parte muito importante porque é uma forma de expressar emoções, refletir identidades culturais e moldar comportamentos humanos, contribuindo de forma positiva para a saúde e a autoestima (Frahm, 2024).

Dentre as diversas formas de entretenimento musical está o Karaokê. Este trata-se de um fenômeno que permite a interatividade entre o sujeito cantor por cima de uma música gravada, contendo apenas seu acompanhamento instrumental. As letras são geralmente exibidas em uma tela de vídeo para guiar o cantor (Patel *et al.*, 2022).

O termo “karaokê” vem da língua japonesa. É um substantivo formado pela composição das palavras “*kara*”, que quer dizer vazio, e “*oke*” de *okesutora* (orquestra), significando sem orquestra. Por sua popularidade, o termo já consta dos dicionários de português e inglês (Sato, 2013).

O Karaokê foi criado pelo músico e inventor japonês Daisuke Inoue, em 1971 (Sato, 2013). Desde esta data o Karaokê evoluiu sua forma de apresentação. Existem hoje versões digitais para celulares, incluindo Android e iOS, além de vídeos no YouTube.

Também entende-se por Karaokê a faixa instrumental separada. A separação de fontes musicais é um problema que constitui uma área importante do processamento digital de sinais. A separação de fontes musicais oriundas de gravações completas pode ser realizada de forma automática com uso de algoritmos de inteligência artificial (IA), isto graças aos aprimoramentos das técnicas de IA em conjunto com métodos de processamento digital de sinais (Huang *et al.*, 2014).

Nesse sentido, este trabalho visa construir um Karaokê utilizando inteligência artificial, com dois modelos pré-treinados. Um modelo para a separação da faixa completa em vocal e instrumental e outro para fazer o reconhecimento automático de fala para a geração de legendas.

1.1 Justificativa

A separação de fontes musicais se constitui em um problema relevante no campo do processamento digital de sinais (*Digital Signal Processing* - DSP) e tem sido amplamente estudada nas últimas décadas (Vincent; Gribonval; Févotte, 2006). Tradicionalmente, a produção de faixas de karaokê pode ser realizada por meio de técnicas clássicas de DSP, empregando métodos como máscaras binárias, filtros digitais do tipo *Finite Impulse Response* (FIR) e *Infinite Impulse Response* (IIR), entre outras abordagens (Simpson; Roma; Plumbley, 2015).

A escolha pela utilização de técnicas baseadas em Inteligência Artificial fundamenta-se em dois aspectos principais. O primeiro consiste na oportunidade de aprofundar os conhecimentos em uma área de grande relevância tecnológica que, embora não seja abordada diretamente na matriz curricular do curso, está alicerçada em conceitos de processamento digital de sinais, matemática e programação desenvolvidos ao longo da graduação. O segundo aspecto refere-se ao desempenho obtido por modelos recentes de separação de fontes musicais, que vêm apresentando resultados superiores aos métodos clássicos de processamento digital de sinais, reduzindo a interferência entre as fontes separadas e produzindo faixas com maior qualidade para aplicações como a geração de versões karaokê (Wang; Lu; Won, 2023).

1.2 Definição do Problema

A geração automática de versões de Karaokê a partir de músicas inteiras apresenta desafios técnicos significativos devido à complexidade dos sinais de áudio musicais, nos quais voz e acompanhamento se encontram misturados em um único sinal (Vincent; Gribonval; Févotte, 2006). A separação de fontes musicais envolve a identificação e extração dos diferentes componentes sonoros presentes, sendo considerada uma tarefa difícil principalmente em gravações monofônicas (Huang *et al.*, 2012).

Outro desafio técnico está relacionado à geração automática de legenda sincronizada a partir do sinal vocal, uma vez que sistemas de reconhecimento automático de fala podem apresentar erros de transcrição, especialmente em sinais

musicais que possuem variações de timbre, ritmo e pronúncia (Demirel; Ahlbäck; Thakkar, 2020).

Define-se como problema deste trabalho o desenvolvimento de um sistema capaz de gerar faixas de karaokê automáticas a partir de músicas digitais inteiras utilizando modelos de IA pré-treinados e a geração de legendas sincronizadas, avaliando a qualidade dos resultados obtidos.

1.3 Objetivo Geral

Implementar um sistema Karaokê baseado em inteligência artificial.

1.4 Objetivos Específicos

- a) estudar métodos de separação de fontes musicais baseados em inteligência artificial;
- b) analisar modelos utilizados na separação de voz e instrumentos musicais;
- c) implementar um sistema capaz de separar músicas digitais em duas faixas: uma vocal e uma instrumental;
- d) implementar um sistema de geração de legendas a partir da faixa vocal;
- e) avaliar a qualidade das faixas geradas.

1.5 Estrutura do Trabalho

Este trabalho está organizado em cinco capítulos, além desta introdução. O Capítulo 2 apresenta a fundamentação teórica necessária para o entendimento do trabalho. Inicialmente, são abordados os principais conceitos relacionados à separação de fontes musicais, incluindo sinais em tempo contínuo e discreto, a Transformada de Fourier de Curto Tempo, a escala Mel, redes neurais do tipo Perceptron de Múltiplas Camadas, a arquitetura *Transformer*, os modelos *Band Split* RoFormer e Mel-Band RoFormer, além da métrica *Signal-to-Distortion Ratio*. Em seguida, são apresentados os conceitos relacionados à geração automática de

legendas, contemplando o modelo *Whisper*, a Distância de Levenshtein e as métricas *Word Error Rate* (WER) e *Character Error Rate* (CER).

O Capítulo 3 descreve a metodologia empregada para o desenvolvimento do sistema proposto. São apresentadas as etapas de configuração dos ambientes computacionais utilizados, bem como o desenvolvimento do processo de separação de fontes musicais e da geração automática de legendas, desde a preparação dos modelos até a execução das inferências.

O Capítulo 4 apresenta os resultados obtidos com a aplicação da metodologia proposta. Inicialmente, é descrito o ambiente experimental utilizado nos testes. Em seguida, são discutidos os resultados da separação de fontes musicais, incluindo uma análise do desempenho computacional, e os resultados da geração automática de legendas, avaliados por meio das métricas WER e CER.

Por fim, o Capítulo 5 apresenta as considerações finais do trabalho, destacando as principais contribuições alcançadas, as limitações identificadas e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O avanço das técnicas de Processamento Digital de Sinais (*Digital Signal Processing* - DSP) e da Inteligência Artificial possibilitam o desenvolvimento de sistemas capazes de analisar, interpretar e classificar sinais complexos em diversas áreas da engenharia. Aplicações envolvendo reconhecimento de padrões, classificação automática de sinais, análise espectral e previsão temporal passaram a utilizar métodos híbridos que combinam ferramentas clássicas de DSP com arquiteturas modernas de redes neurais artificiais (Demirel; Ahlbäck; Thakkar, 2020).

O desenvolvimento recente de modelos baseados em aprendizado profundo ampliou significativamente as capacidades dos sistemas de DSP, permitindo que tarefas tradicionalmente complexas, como separação de fontes musicais e reconhecimento automático de fala, alcançassem níveis de desempenho anteriormente inviáveis utilizando apenas técnicas clássicas. Entre essas abordagens destacam-se as arquiteturas baseadas em mecanismos de atenção (Transformers), capazes de modelar dependências de longo alcance em sinais temporais, bem como modelos especializados que integram representações espectrais do áudio com redes neurais profundas para a realização de tarefas específicas (Demirel; Ahlbäck; Thakkar, 2020).

A compreensão do funcionamento dessas arquiteturas requer o conhecimento de conceitos fundamentais de DSP e do aprendizado de máquina. Entre os conceitos abordados destacam-se a Transformada de Fourier de Tempo Curto (*Short-Time Fourier Transform* - STFT), a representação espectral na escala Mel, as redes neurais artificiais do tipo perceptrons de múltiplas camadas (*Multilayer Perceptron* - MLP) e a arquitetura Transformer, fundamentos que servem de base dos modelos Mel-Band RoFormer e *Whisper* empregados neste trabalho (Demirel; Ahlbäck; Thakkar, 2020).

Este capítulo apresenta os fundamentos teóricos necessários para a compreensão dessas técnicas, descrevendo seus princípios matemáticos e funcionamento.

2.1 Separação de fontes musicais

Nessa subseção serão abordados os conceitos necessários para a compreensão do funcionamento do modelo utilizado para a separação de fontes musicais.

2.1.1 Sinais de Tempo Contínuo vs. Tempo Discreto

Um sinal é definido como uma descrição formal de um fenômeno físico que evolui no tempo ou no espaço. Enquanto o mundo físico é modelado pelo paradigma contínuo, onde o tempo é um fluxo ininterrupto e infinitamente divisível, a computação moderna opera exclusivamente no domínio discreto. No modelo contínuo, a informação está disponível em qualquer instante, o que fundamenta a descrição analítica de como os sinais se propagam em meios físicos (Oppenheim; Schafer, 2012; Zhang *et al.*, 2020)

O modelo de tempo discreto representa o sinal como uma sequência de amostras. O índice da amostra (n) funciona como uma medida de tempo adimensional que, embora não possua unidade física intrínseca, impõe uma ordem sequencial rigorosa aos valores. Embora o processamento algorítmico se baseie nesses índices sequenciais, a interpretação física dos dados depende do intervalo de amostragem associado, que garante que a estrutura temporal do sinal seja preservada durante o aprendizado. Este modelo discreto possibilita a aplicação de técnicas complexas de processamento e aprendizado profundo, como redes neurais recorrentes e convolucionais, que processam informações de forma sistemática e eficiente (Oppenheim; Schafer, 2012).

2.1.2 Transformada de Fourier de Tempo Curto

A Transformada de Fourier é uma das ferramentas matemáticas mais importantes do processamento digital de sinais, sendo utilizada para representar um sinal no domínio da frequência. Entretanto, a Transformada de Fourier convencional apresenta uma limitação importante: ela fornece apenas a composição espectral

global do sinal, sem indicar em qual instante temporal determinada frequência ocorre (Oppenheim; Schafer, 2012).

Essa limitação torna-se crítica na análise de sinais não estacionários, isto é, sinais cujas características espectrais variam ao longo do tempo. Exemplos incluem sinais de voz, áudio musical, sinais biomédicos e vibrações mecânicas (Oppenheim; Schafer, 2012).

Para solucionar esse problema foi desenvolvida a Transformada de Fourier de Tempo Curto, técnica que fornece simultaneamente informações temporais e espectrais do sinal, através de uma representação bidimensional conhecida como espectrograma (Oppenheim; Schafer, 2012).

2.1.2.1 Princípio de Funcionamento da STFT

A STFT consiste na aplicação da Transformada de Fourier em pequenas janelas temporais do sinal. O sinal original é dividido em segmentos de curta duração por meio de uma função janela, assumindo-se que cada segmento pode ser considerado aproximadamente estacionário (Oppenheim; Schafer, 2012).

Matematicamente, a STFT de um sinal contínuo $x(t)$ é demonstrada pela Equação 1 (Oppenheim; Schafer, 2012).

$$STFT\{x(t)\}(\tau, f) = \int_{-\infty}^{\infty} x(t)w(t - \tau)e^{-j2\pi ft} dt \quad (1)$$

Onde $x(t)$ representa o sinal no domínio do tempo, $w(t - \tau)$ é a função janela centrada no instante τ , f representa a frequência e $e^{-j2\pi ft}$ corresponde ao núcleo complexo da Transformada de Fourier (Oppenheim; Schafer, 2012).

Para sinais discretos, a STFT pode ser expressa conforme mostra a Equação 2:

$$X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]w[n - m]e^{-j\omega n} \quad (2)$$

Nesse contexto $x[n]$ é o sinal discreto, $w[n - m]$ é a janela deslocada e ω representa a frequência angular discreta (Oppenheim; Schafer, 2012). Esta

representação permite que redes neurais, como o MLP e o Transformer, analisem a evolução das componentes de frequência do áudio para tarefas de separação de fontes e transcrição (Haykin, 2007; Vaswani *et al.*, 2017).

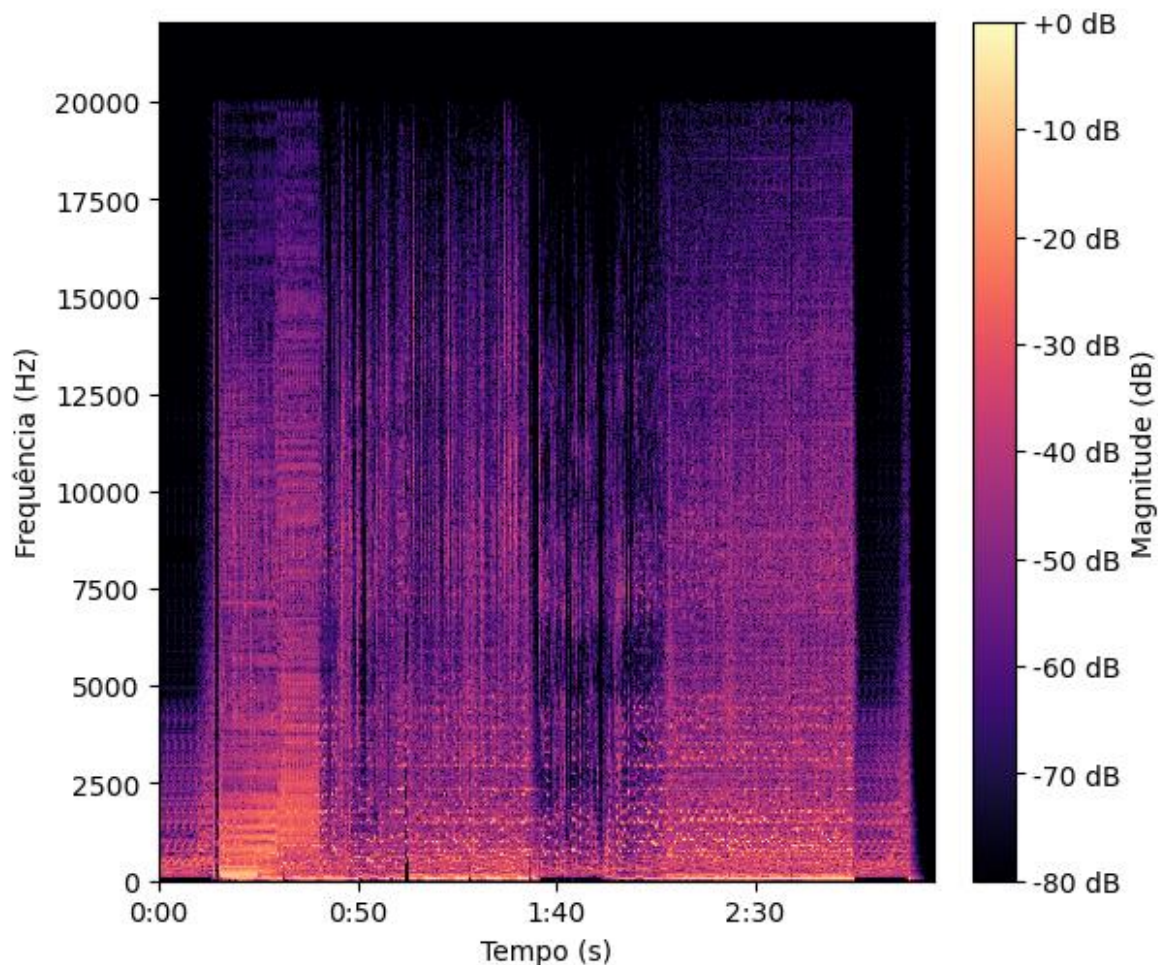
2.1.2.2 Processo de Segmentação do Sinal

O funcionamento da STFT pode ser dividido em etapas. Primeiramente o sinal é segmentado em pequenas janelas temporais de curta duração (Oppenheim; Schafer, 2012). Em seguida, cada segmento é multiplicado por uma função de ponderação, também conhecida como sequência de janelamento ($w[n]$) (Oppenheim; Schafer, 2012).

Após o janelamento, a Transformada de Fourier é aplicada individualmente em cada segmento. No domínio discreto, isto corresponde ao cálculo de uma sequência de Transformadas de Fourier Discretas (*Discrete Fourier Transform* - DFT) de N pontos sobre os blocos de amostras janeladas. Por fim, os espectros obtidos são organizados cronologicamente, à medida que a janela "desliza" sobre o sinal original em saltos de um intervalo de tempo específico, capturando a evolução das componentes de frequência ao longo do tempo.

Esse processo resulta em uma representação bidimensional denominada espectrograma, em que o eixo horizontal representa o tempo (n), o vertical representa a frequência e a intensidade de cor representa a magnitude espectral em escala logarítmica, a Figura 1 apresenta um exemplo de espectrograma a partir de uma música. Esta visualização é essencial para identificar assinaturas acústicas, como os formantes na voz humana ou padrões rítmicos em sinais musicais (Jurafsky; Martin, 2008; Oppenheim; Schafer, 2012; Prandoni; Vetterli, 2008).

Figura 1 - Exemplo de Espectrograma



Fonte: Elaborado pelo autor (2026).

O espectrograma é uma das formas mais utilizadas para visualização de sinais no domínio tempo-frequência, pois permite identificar a evolução espectral do áudio ao longo do tempo (Jurafsky; Martin, 2008; Prandoni; Vetterli, 2008).

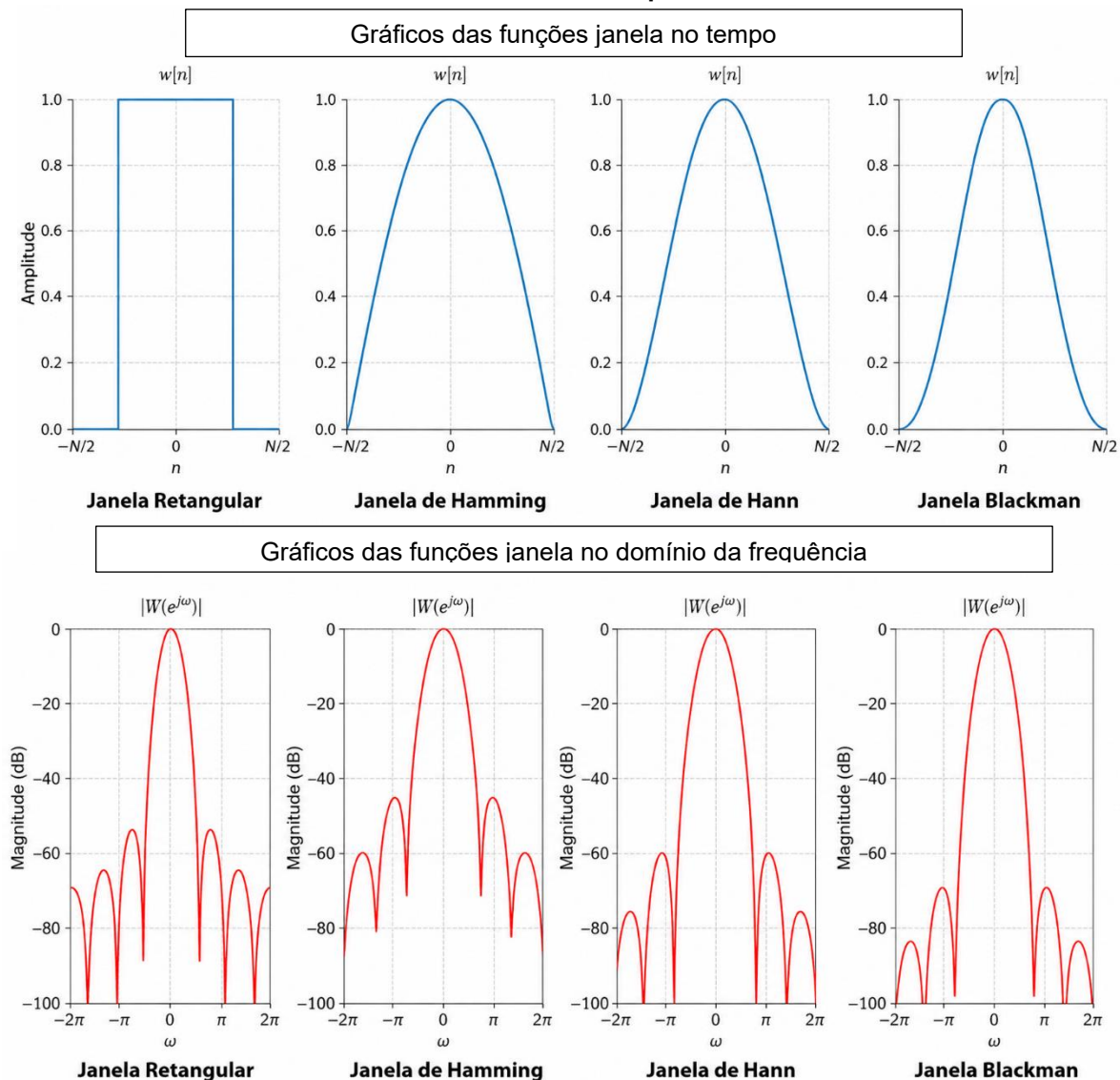
2.1.2.3 Funções Janela

A escolha da função janela influencia diretamente a resolução temporal e espectral da STFT, além de afetar o vazamento espectral (spectral leakage) observado na análise do sinal. Entre as funções janela mais utilizadas destacam-se a janela retangular, a janela de Hamming, a janela de Hann e a janela Blackman (Oppenheim; Schaffer, 2012).

A janela retangular caracteriza-se pela simplicidade computacional, porém apresenta elevado vazamento espectral. A janela de Hamming reduz

significativamente os lóbulos laterais, sendo amplamente empregada em aplicações práticas de processamento digital de sinais. A janela de Hann possui comportamento semelhante ao da janela de Hamming, proporcionando boa suavização espectral. Já a janela Blackman apresenta maior atenuação dos lóbulos laterais, embora resulte em menor resolução espectral, os gráficos dessas janelas podem ser visualizados na Figura 2, elaborado por meio de uma software matemático (Oppenheim; Schaffer, 2012).

Figura 2 - Comparação das funções janela no domínio do tempo (azul) e suas respectivas respostas em frequência (vermelho), ilustrando o impacto da geometria da janela na resolução e no vazamento espectral



Fonte: Elaborado pelo autor (2026).

2.1.2.4 Resolução Tempo-Frequência

Existe um compromisso fundamental entre resolução temporal e resolução espectral na STFT, janelas curtas tem uma melhor resolução temporal enquanto janelas longas tem a resolução espectral melhor (Oppenheim; Schafer, 2012).

Esse comportamento decorre diretamente do princípio da incerteza de Fourier, segundo o qual não é possível obter simultaneamente alta precisão temporal e espectral.

2.1.3 Escala Mel

A representação espectral obtida pela Transformada de Fourier de Tempo Curto utiliza uma escala linear de frequências, na qual intervalos iguais correspondem a diferenças constantes em hertz (Simpson; Roma; Plumbley, 2015; Zhang *et al.*, 2020). Essa representação não corresponde à forma como o sistema auditivo humano percebe as variações de frequência. Em baixas frequências, pequenas diferenças são facilmente distinguidas pelo ouvido humano, enquanto, em frequências elevadas, essa capacidade de discriminação diminui progressivamente (Li; Xu; Zhang, 2021; Zhang *et al.*, 2020).

Com o objetivo de aproximar a representação espectral da percepção auditiva humana, foi desenvolvida a escala Mel (*Mel Scale*). O termo Mel designa uma unidade psicoacústica utilizada para representar a percepção subjetiva da altura tonal (*pitch*) pelo sistema auditivo humano. Diferentemente da frequência física, medida em hertz, a escala Mel procura representar a forma como variações de frequência são percebidas por um ouvinte, sendo baseada em experimentos de percepção auditiva realizados por Stevens, Volkman e Newman (Li; Xu; Zhang, 2021).

2.1.3.1 Princípio de Funcionamento

A escala Mel estabelece uma relação não linear entre a frequência física, medida em hertz (Hz), e a frequência percebida pelo sistema auditivo humano. Essa relação é dada pela Equação 3, onde m representa a frequência da escala Mel e f a

frequência em hertz (Hz). A relação inversa pode ser vista na Equação 4 (Oppenheim; Schafer, 2012):

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (3)$$

$$f = 700 \left(10^{\frac{m}{2595}} - 1 \right) \quad (4)$$

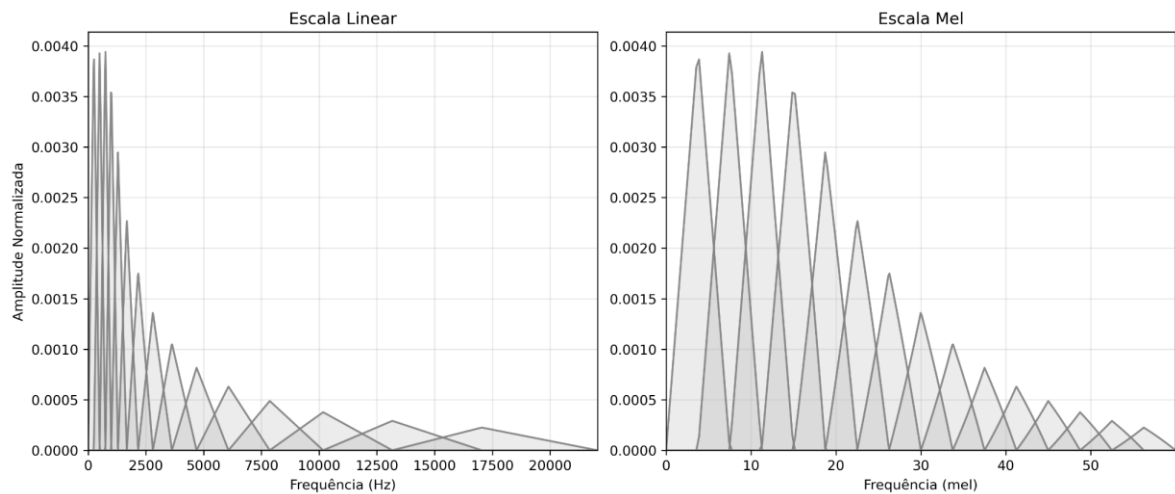
Observa-se que a escala apresenta comportamento aproximadamente linear para frequências inferiores a cerca de 1 kHz e comportamento progressivamente logarítmico para frequências. Como consequência, baixas frequências são representadas com maior resolução, enquanto altas frequências são comprimidas, produzindo uma representação mais compatível com a percepção auditiva humana (Jurafsky; Martin, 2008; Wang; Lu; Won, 2023).

2.1.3.2 Banco de filtros Mel

Embora a conversão entre hertz e Mel possa ser descrita matematicamente pelas equações anteriores, a transformação de um espectrograma linear para um espectrograma Mel não é realizada por uma simples conversão das frequências do espectrograma linear. Na prática, essa conversão é realizada por meio de um banco de filtros triangulares, distribuídos uniformemente na escala Mel (Wang; Lu; Won, 2023).

Cada filtro acumula a energia presente em uma faixa específica de frequências e apresenta sobreposição parcial com os filtros adjacentes. Essa característica permite representar de forma contínua a distribuição espectral de energia, reduzindo simultaneamente a dimensionalidade da representação e enfatizando componentes de frequência mais relevantes para a percepção auditiva. Conforme ilustrado na Figura 3, embora os filtros sejam distribuídos uniformemente na escala Mel, sua representação em frequência linear resulta em uma maior concentração de filtros nas baixas frequências e um espaçamento progressivamente maior nas regiões de altas frequências (Oppenheim; Schafer, 2012; Wang; Lu; Won, 2023).

Figura 3 - Comparação da distribuição dos filtros do banco Mel nas escalas linear e Mel

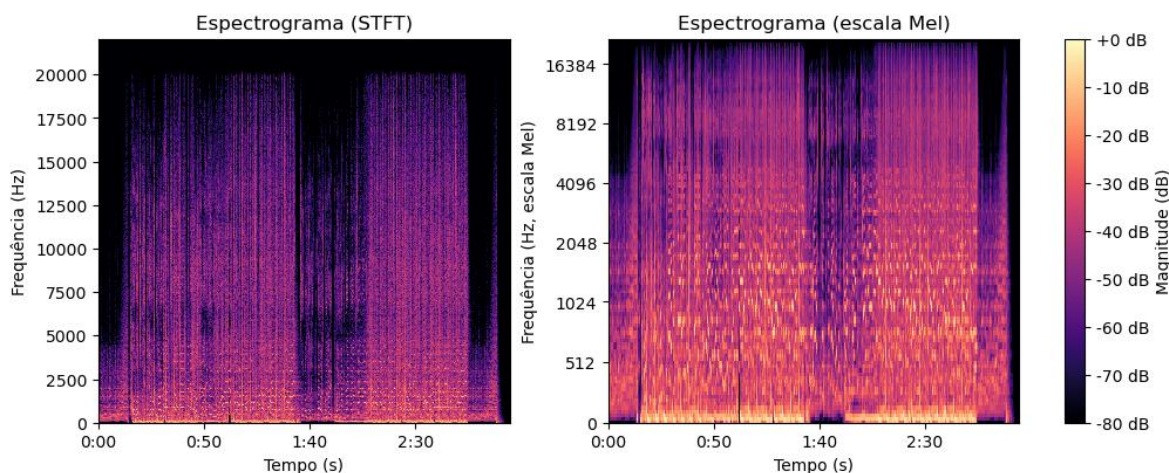


Fonte: Elaborado pelo autor (2026).

A energia calculada por cada filtro passa a compor uma nova representação espectral denominado espectrograma Mel, cuja distribuição das frequências deixa de ser linear e passa a refletir de maneira mais adequada às características perceptuais do sistema auditivo humano (Jurafsky; Martin, 2008; Wang; Lu; Won, 2023).

A Figura 4 apresenta uma comparação entre um espectrograma obtido diretamente pela STFT e o correspondente espectrograma na escala Mel, evidenciando a redistribuição da resolução espectral ao longo do eixo de frequências (Wang; Lu; Won, 2023).

Figura 4 - Comparação entre os espectrogramas STFT e Mel



Fonte: Elaborado pelo autor (2026).

2.1.3.3 Aplicação nos modelos de Inteligência Artificial

A utilização da escala Mel tornou-se uma das principais formas de representação de sinais de áudio em aplicações modernas de aprendizado profundo, pois fornece uma representação mais compacta e perceptualmente significativa do conteúdo espectral do sinal (Demirel; Ahlbäck; Thakkar, 2020).

No contexto deste trabalho, essa representação é utilizada tanto na arquitetura Mel-Band RoFormer, empregada na etapa de separação de fontes musicais, quanto no modelo *Whisper*, utilizado para a geração automática de legendas. Em ambos os casos, o espectrograma Mel constitui a entrada das redes neurais, permitindo que os modelos aprendam padrões acústicos a partir de uma representação mais compatível com a forma como o sistema auditivo humano percebe os sons (Demirel; Ahlbäck; Thakkar, 2020; Liu; Kong; Liu, 2021).

Dessa forma, embora originalmente desenvolvida a partir de estudos em psicoacústica, a escala Mel tornou-se uma das principais formas de representação de sinais de áudio em aplicações de aprendizado profundo. Sua utilização como etapa de pré-processamento permite reduzir a dimensionalidade dos dados de entrada ao mesmo tempo em que preserva características espectrais relevantes para a percepção auditiva (Demirel; Ahlbäck; Thakkar, 2020).

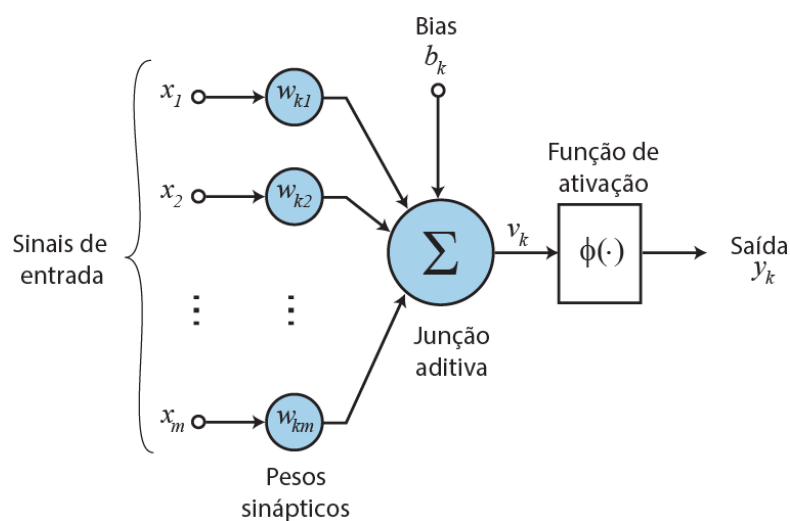
2.1.4 Redes Neurais Perceptrons de Múltiplas Camadas

As Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no funcionamento do cérebro humano. Entre as arquiteturas mais tradicionais destaca-se o MLP, amplamente utilizado em tarefas de classificação, regressão e reconhecimento de padrões (Haykin, 2007).

O MLP é formado por múltiplas camadas de neurônios artificiais conectados entre si, permitindo a modelagem de relações não lineares complexas.

Os neurônios artificiais constituem a unidade básica de processamento das RNAs. Eles se inspiram no funcionamento de neurônios reais, recebem um estímulo, processam essas informações e transmitem sinais de acordo com o estímulo recebido. Analogicamente os neurônios artificiais recebem sinais de entrada, realizam operações matemáticas sobre esses sinais e produzem uma saída (Haykin, 2007).

Figura 5 - Modelo não-linear de neurônio



Fonte: Haykin (2007)

Como visto na Figura 5, o neurônio artificial processa os sinais de entrada atribuindo pesos a eles, definindo sua importância. Esses pesos são definidos pelo modelo durante o treinamento. O bias demonstrado na figura atua como um deslocamento da função de ativação, permitindo maior flexibilidade ao modelo (Haykin, 2007).

2.1.4.1 Modelo Matemático do Neurônio Artificial

Um neurônio artificial realiza três operações principais: recepção das entradas, uma combinação linear ponderada, e a aplicação de uma função de ativação (Haykin, 2007; Zhang *et al.*, 2020).

O funcionamento básico de um neurônio pode ser representado pela Equação 5 abaixo.

$$y = f \left(\sum_{i=1}^n w_i x_i + b \right) \quad (5)$$

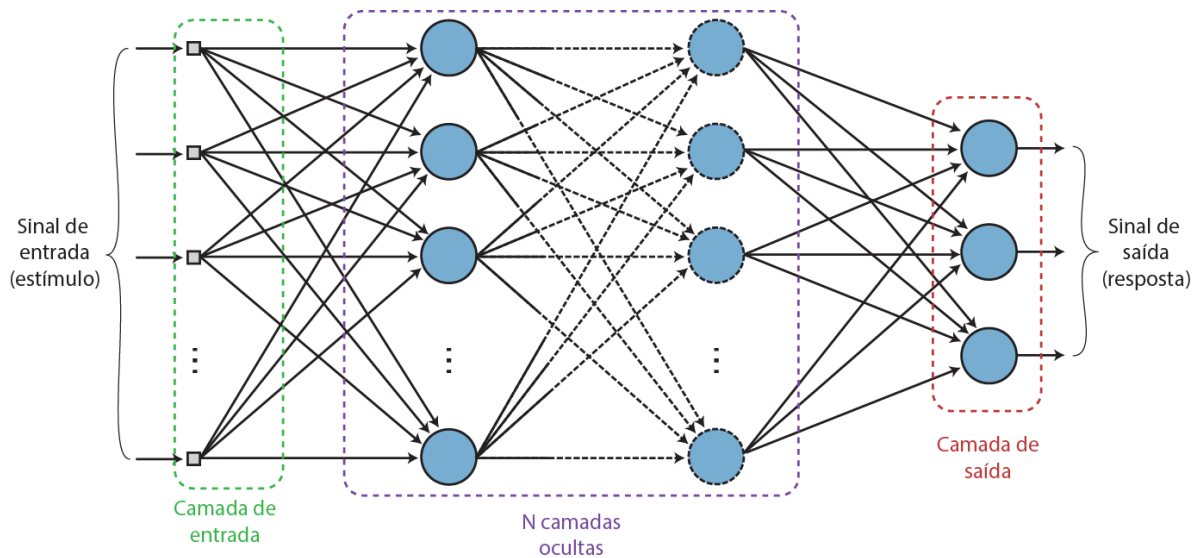
Sendo x_i as entradas, w_i representando os pesos sinápticos, b é o termo de bias, $f(\cdot)$ é a função de ativação e y representa a saída do neurônio (Haykin, 2007).

2.1.4.2 Estrutura de uma Rede MLP

Uma rede MLP é composta por três tipos principais de camadas. A camada de entrada é responsável por receber os dados que serão processados pela rede. As camadas ocultas realizam o processamento intermediário das informações, enquanto a camada de saída produz a resposta final do modelo. Em geral, o aumento da quantidade de neurônios e de camadas ocultas amplia a capacidade da rede de aprender representações mais complexas dos dados, permitindo modelar problemas de maior complexidade (Haykin, 2007; Zhang *et al.*, 2020).

Cada neurônio realiza operações matemáticas envolvendo pesos sinápticos e funções de ativação, como mostrado na Figura 6 (Haykin, 2007).

Figura 6 - Grafo arquitetural de um perceptron de múltiplas camadas com N camadas ocultas

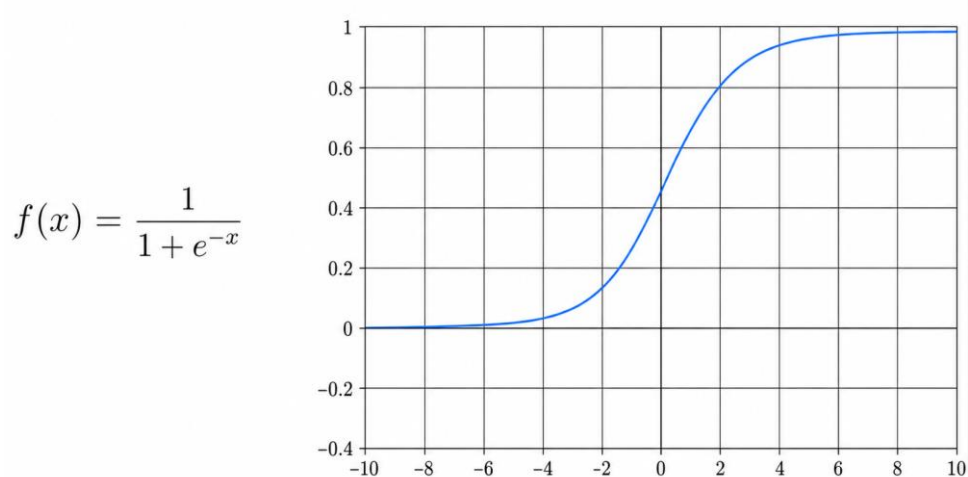


Fonte: adaptado de Haykin (2007)

2.1.4.3 Funções de Ativação

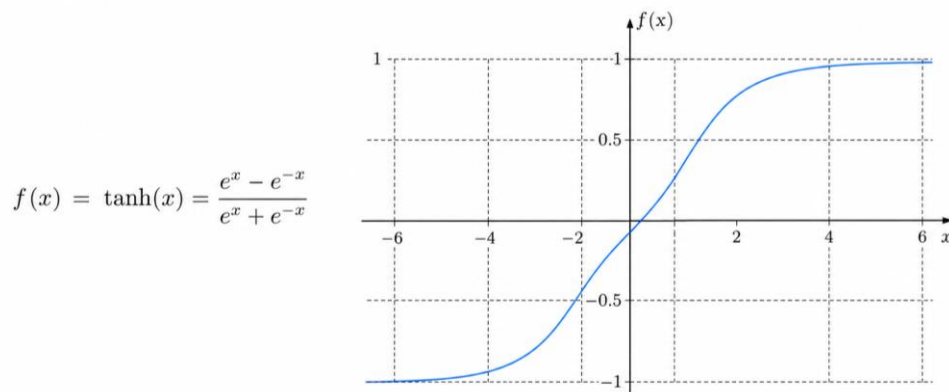
As funções de ativação introduzem não linearidade ao modelo. Entre as mais utilizadas destacam-se as funções Sigmoid, que produz saídas entre 0 e 1 (Figura 7), Tangente Hiperbólica que tem como resposta saídas entre -1 e 1 (Figura 8) e ReLU, amplamente utilizada devido a simplicidade computacional e eficiência de treinamento da rede neural exemplificado na Figura 9 (Haykin, 2007; Li; Xu; Zhang, 2021; Zhang *et al.*, 2020).

Figura 7 - Função Sigmoid



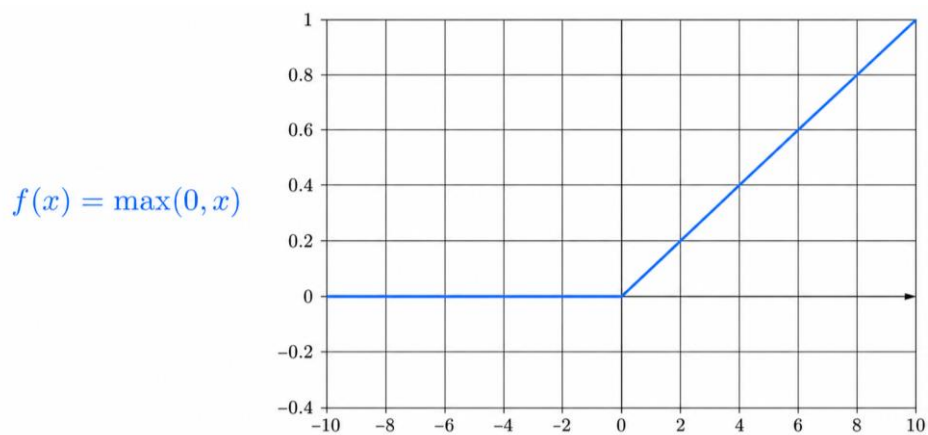
Fonte: Elaborado pelo autor (2026).

Figura 8 - Função da Tangente Hiperbólica



Fonte: Elaborado pelo autor (2026).

Figura 9 - Função ReLU



Fonte: Elaborado pelo autor (2026).

2.1.4.4 Processo de Treinamento

O treinamento de uma rede MLP consiste no ajuste iterativo dos pesos sinápticos com o objetivo de minimizar o erro entre a saída produzida pela rede e a saída desejada (Jurafsky; Martin, 2008; Zhang *et al.*, 2020). Esse processo é realizado em duas etapas principais. Na primeira, denominada *Forward Propagation*, os dados são propagados da camada de entrada até a camada de saída, produzindo uma predição. Em seguida, na etapa de *Backpropagation*, o erro é propagado no sentido inverso da rede, permitindo calcular o gradiente da função de erro em relação a cada peso e, conseqüentemente, realizar sua atualização. Para isso, o algoritmo utiliza a

regra da cadeia do cálculo diferencial, tornando possível otimizar iterativamente os parâmetros da rede (Haykin, 2007; Zhang *et al.*, 2020).

O algoritmo de atualização geralmente utiliza gradiente descendente representado na Equação 6 abaixo (Haykin, 2007; Zhang *et al.*, 2020).

$$w_{novo} = w_{antigo} - \eta \frac{\partial E}{\partial w} \quad (6)$$

Onde η é a taxa de aprendizado e E representa a função erro (Haykin, 2007; Zhang *et al.*, 2020).

Durante o treinamento, a rede pode memorizar excessivamente os dados de treinamento, fenômeno denominado *overfitting*. São utilizadas algumas técnicas para mitigar esse fenômeno. Entre as mais comuns estão a *dropout*, regularização L1 e L2, *early stopping* e o aumento de dados (*data augmentation*) (Demirel; Ahlbäck; Thakkar, 2020; Haykin, 2007; Zhang *et al.*, 2020).

A capacidade de generalização representa a habilidade do modelo em responder corretamente a dados não vistos durante o treinamento (Haykin, 2007; Zhang *et al.*, 2020).

2.1.5 Transformers

Os Transformers representam uma arquitetura de redes neurais introduzida por Vaswani e colaboradores (2017) e explorada por Liu, Kong e Liu (2021). Essa arquitetura revolucionou aplicações envolvendo dados sequenciais ao substituir mecanismos recorrentes por mecanismos de atenção (Demirel; Ahlbäck; Thakkar, 2020; Li; Xu; Zhang, 2021; Liu; Kong; Liu, 2021).

Embora inicialmente propostos para tarefas de processamento de linguagem natural, os Transformers passaram a ser empregados em diversas áreas, incluindo visão computacional, análise de séries temporais, bioinformática, processamento de sinais e processamento de áudio (Demirel; Ahlbäck; Thakkar, 2020; Li; Xu; Zhang, 2021; Liu; Kong; Liu, 2021).

Antes dos Transformers, modelos como *Recurrent Neural Networks* (RNNs) e *Long Short-Term Memories* (LSTMs) dominavam o processamento

sequencial. Entretanto, esses modelos apresentam limitações importantes como dificuldade em capturar dependências de longo prazo, treinamento sequencial lento, problemas de gradiente e baixa paralelização computacional (Demirel; Ahlbäck; Thakkar, 2020; Li; Xu; Zhang, 2021; Liu; Kong; Liu, 2021).

Os Transformers solucionam essas limitações utilizando mecanismos de atenção paralela (Liu; Kong; Liu, 2021).

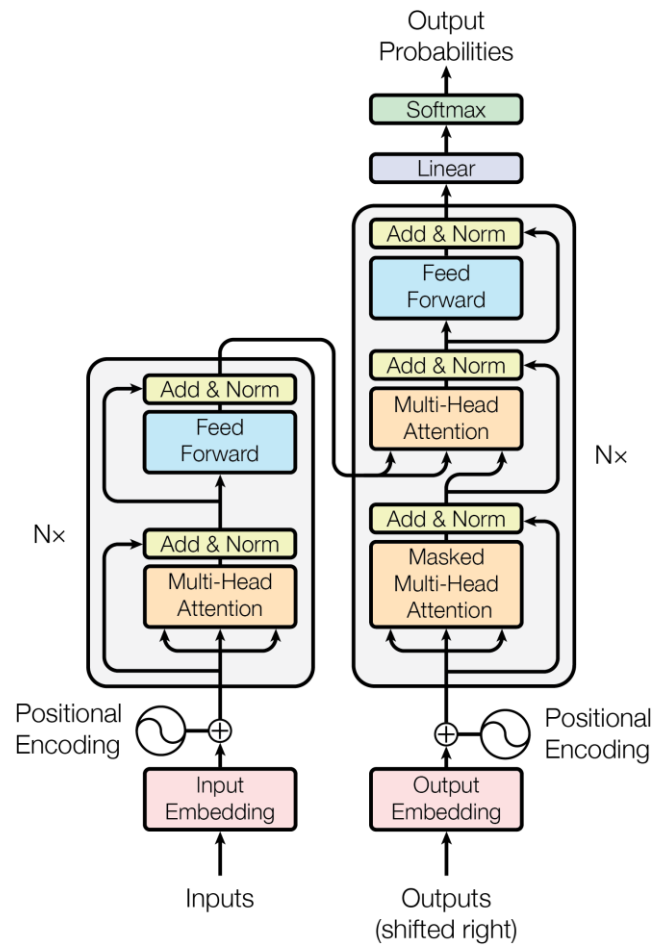
2.1.5.1 Estrutura Geral do Transformer

A arquitetura Transformer clássica é composta por dois blocos principais: um *Encoder*, responsável por transformar a sequência de entrada em uma representação vetorial de alto nível, e um *Decoder*, responsável por utilizar essa representação para gerar a sequência de saída correspondente, a estrutura geral pode ser vista na Figura 10 (Radford *et al.*, 2022; Vaswani *et al.*, 2017; Zhang *et al.*, 2020).

Cada *Encoder* é constituído por uma camada de atenção do tipo *Multi-Head Self-Attention*, seguida por uma rede neural totalmente conectada (*Feed Forward Network* – FFN). Entre essas camadas são empregadas conexões residuais (*Residual Connections*) e mecanismos de normalização (*Layer Normalization*), cuja finalidade é facilitar o treinamento da rede e melhorar a estabilidade da propagação do gradiente (Radford *et al.*, 2022; Vaswani *et al.*, 2017; Zhang *et al.*, 2020).

De forma semelhante, cada *Decoder* também possui mecanismos de atenção e redes *feed forward*, acrescentando uma camada de atenção responsável por relacionar as informações produzidas pelo *Encoder* com a sequência parcialmente gerada pelo próprio *Decoder* (Vaswani *et al.*, 2017; Zhang *et al.*, 2020). Essa estrutura permite ao modelo aprender representações altamente contextualizadas e capturar dependências de longo alcance entre os elementos da sequência (Li; Xu; Zhang, 2021; Lu *et al.*, 2023; Vaswani *et al.*, 2017).

Figura 10 - Estrutura geral da arquitetura Transformer



Fonte: Vaswani et al. (2017).

A arquitetura Transformer é composta por blocos repetitivos de *Encoder* e *Decoder*, permitindo a construção de modelos profundos capazes de aprender representações cada vez mais abstratas da sequência de entrada (Zhang *et al.*, 2020).

2.1.5.2 Mecanismo de Atenção

O conceito central dos Transformers é o mecanismo de atenção (*attention*), responsável por determinar quais partes da sequência possuem maior relevância para determinada tarefa (Patel *et al.*, 2022; Vaswani *et al.*, 2017).

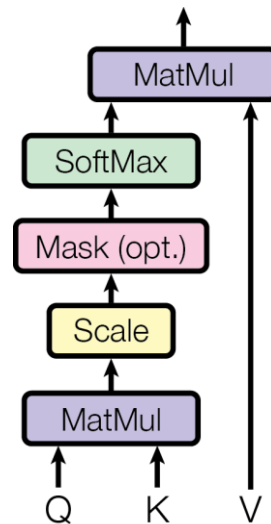
O mecanismo utiliza três vetores principais, *Query* (Q), *Key* (K) e *Value* (V). A atenção escalada por produto interno é definida pela Equação 7 e representado pela Figura 11 (Lu *et al.*, 2023; Vaswani *et al.*, 2017; Li; Xu; Zhang, 2021; Zhang *et al.*, 2020).

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad (7)$$

QK^T mede similaridade, d_k representa a dimensão dos vetores e $softmax$ normaliza os pesos de atenção.

Os Transformers utilizam múltiplos mecanismos de atenção em paralelo, permitindo que diferentes relações sejam aprendidas simultaneamente. Esse mecanismo é denominado *Multi-Head Attention*. Cada cabeça de atenção aprende diferentes características da sequência, aumentando a capacidade representacional do modelo (Demirel; Ahlbäck; Thakkar, 2020; Vaswani *et al.*, 2017; Zhang *et al.*, 2020).

Figura 11 - Funcionamento do mecanismo de atenção escalada por produto interno (*Scaled Dot-Product Attention*)



Fonte: Vaswani *et al.* (2017).

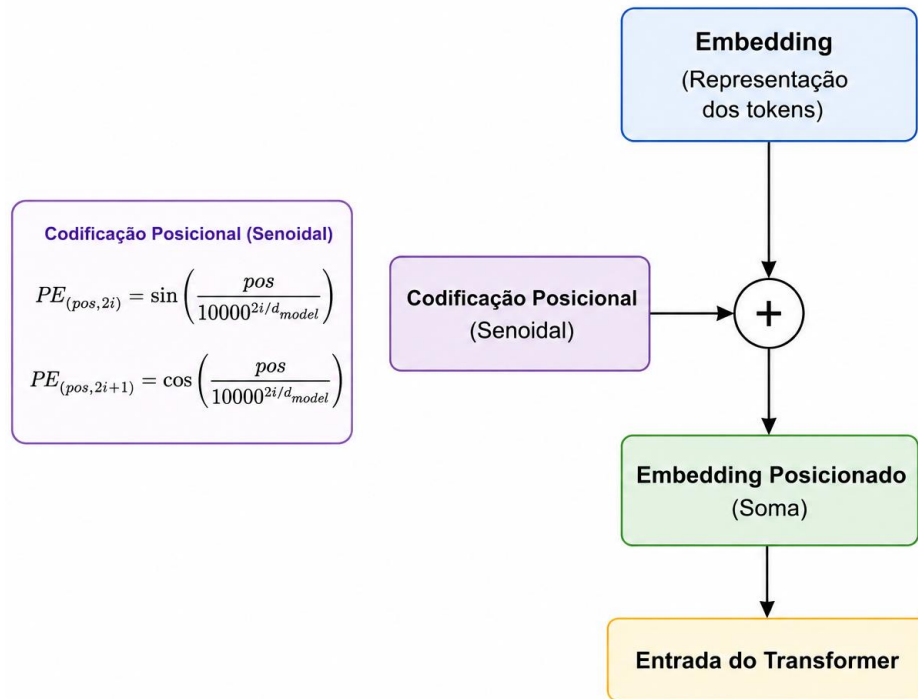
2.1.5.3 Codificação Posicional

Como os Transformers não utilizam recorrência, é necessário inserir informações sobre a posição temporal dos dados (Figura 12), é realizada por meio da codificação posicional, representada nas Equações 8 e 9 abaixo (Demirel; Ahlbäck; Thakkar, 2020; Li; Xu; Zhang, 2021; Su *et al.*, 2024; Vaswani *et al.*, 2017).

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right) \quad (8)$$

$$PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d}}\right) \quad (9)$$

Figura 12 - Incorporação da codificação posicional à representação de entrada do Transformer



Fonte: Elaborado pelo autor (2026).

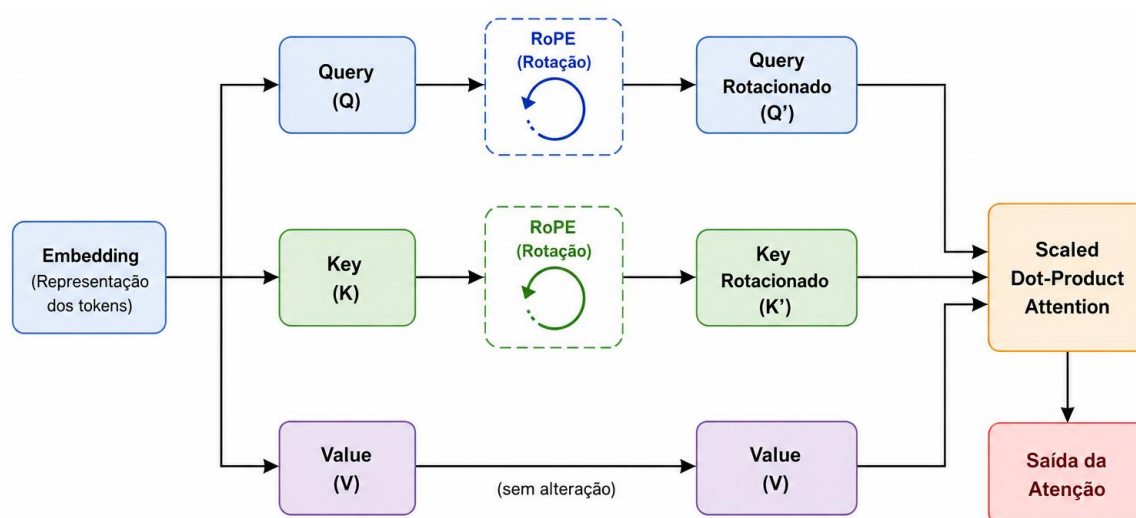
A utilização dessas funções senoidais permite que o modelo identifique relações de ordem entre os elementos da sequência, preservando simultaneamente informações relativas às distâncias entre diferentes posições (Oppenheim; Schafer, 2012).

2.1.5.4 Rotary Positional Embedding (RoPE)

Embora a codificação posicional senoidal introduzida na arquitetura Transformer original seja eficiente para diversas aplicações, ela apresenta limitações quando empregada em tarefas que envolvem sequências longas. Como as informações posicionais são adicionadas diretamente aos vetores de entrada, a capacidade do modelo em representar relações relativas entre elementos muito distantes pode ser reduzida (Demirel; Ahlbäck; Thakkar, 2020; Liu; Kong; Liu, 2021).

Com o objetivo de superar essa limitação, foi proposto o *Rotary Positional Embedding* (RoPE), introduzido por Su et al. (2021). Diferentemente da codificação posicional clássica, o RoPE incorpora as informações de posição diretamente durante o cálculo do mecanismo de atenção, por meio da aplicação de uma transformação rotacional aos vetores *Query* e *Key*, incorporando a informação de posição diretamente ao mecanismo de atenção (Demirel; Ahlbäck; Thakkar, 2020; Liu; Kong; Liu, 2021).

Figura 13 - Aplicação do Rotary Positional Embedding (RoPE) aos vetores Query e Key



Fonte: Elaborado pelo autor (2026).

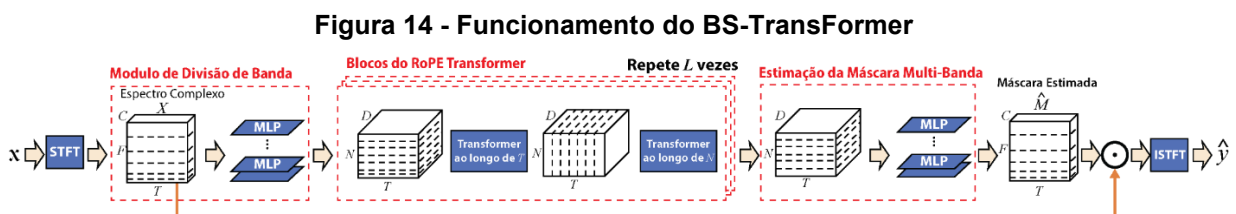
Conforme ilustrado na Figura 13, apenas os vetores *Query* e *Key* são submetidos à rotação posicional, enquanto o vetor *Value* permanece inalterado. Após essa transformação, os vetores rotacionados são utilizados no cálculo da atenção escalada por produto interno (*Scaled Dot-Product Attention*), permitindo que a informação posicional seja considerada durante o próprio mecanismo de atenção (Li; Xu; Zhang, 2021).

Como consequência, o modelo passa a preservar de forma mais eficiente as relações entre as diferentes posições da sequência, considerando simultaneamente a distância e a ordem entre seus elementos. Essa característica melhora a capacidade do Transformer em modelar dependências de longo alcance, aspecto particularmente importante em aplicações envolvendo sinais de áudio e outras sequências extensas (Li; Xu; Zhang, 2021; Liu; Kong; Liu, 2021).

A utilização do RoPE tornou-se especialmente relevante em arquiteturas modernas baseadas em Transformers por preservar de forma mais consistente as relações posicionais entre os elementos da sequência. Em decorrência dessas características, essa técnica foi incorporada em diversos modelos recentes, incluindo o *RoFormer*, o *BS-RoFormer* e o *Mel-Band RoFormer*, utilizados como base para a separação de fontes musicais abordada neste trabalho (Li; Xu; Zhang, 2021; Liu; Kong; Liu, 2021).

2.1.6 Band-Split RoPE Transformer

O Band-Split RoPE Transformer (BS-RoFormer) é uma arquitetura de estado da arte para separação de fontes musicais que opera integralmente no domínio da frequência. O modelo foi proposto para superar a rigidez de redes convolucionais (*Convolutional Neural Networks* - CNNs) tradicionais, que aplicam um mesmo conjunto de filtros a todo o espectro de frequência, ignorando que diferentes instrumentos musicais possuem padrões rítmicos e harmônicos distintos em faixas espectrais específicas (Lu *et al.*, 2023).



Fonte: adaptado de Lu, *et al.* (2023).

2.1.6.1 Arquitetura

A Figura 14 apresenta uma visão geral da arquitetura do BS-RoFormer. Inicialmente, o sinal de áudio é convertido para o domínio tempo-frequência por meio da STFT. O espectrograma complexo obtido é dividido em sub-bandas independentes, que são projetadas para um espaço de características latentes antes de serem processadas pelos blocos RoFormer. Ao final do processamento, o modelo estima máscaras espectrais para cada sub-banda, que são utilizadas para reconstruir o espectrograma da fonte desejada e, posteriormente, o sinal de áudio por meio da

Transformada de Fourier de Tempo Curto Inversa (*Inverse Short-Time Fourier Transform* - iSTFT) (Lu *et al.*, 2023).

2.1.6.2 Módulo de Divisão de Banda (*Band-Split Module*)

O módulo de divisão de bandas é responsável por segmentar o espectrograma complexo em um conjunto de sub-bandas não sobrepostas ao longo do eixo das frequências. Cada sub-banda é processada individualmente por uma pequena rede composta por uma camada RMSNorm (*Root Mean Square Layer Normalization*) seguida de uma camada linear (MLP), responsável por projetar as informações espectrais para um espaço de características latentes. Dessa forma, o modelo produz representações específicas para cada faixa do espectro, permitindo que padrões característicos de diferentes instrumentos sejam aprendidos de maneira independente (Lu *et al.*, 2023).

2.1.6.3 Blocos RoFormer

Diferente dos modelos Transformer tradicionais que processam sequências de dados de maneira unidimensional, a arquitetura do BS-RoFormer alterna o processamento entre os eixos temporal e espectral. Inicialmente, o Transformer Temporal, ou Time-Transformer, modela as dependências ao longo do tempo dentro de cada sub-banda. Em seguida, o Transformer de Sub-banda modela as relações entre diferentes regiões do espectro para um mesmo instante temporal (Lu *et al.*, 2023).

Essa estratégia permite que o modelo capture simultaneamente informações rítmicas, temporais e harmônicas, favorecendo a separação de fontes musicais com características espectrais distintas (Lu *et al.*, 2023).

2.1.6.4 Estimativa de Máscara Multi-banda e Reconstrução

A etapa final consiste em retornar as representações latentes ao domínio do espectrograma para o isolamento da fonte alvo. Através de um módulo de estimativa de máscara multi-banda, MLPs individuais operam por sub-banda. O

modelo estima Máscaras de Razão Ideal Complexas (*Complex Ideal Ratio Masks* - cIRMs), que contemplam tanto os componentes reais quanto imaginários do sinal. A estimativa final da fonte isolada é obtida pela multiplicação elemento a elemento entre a máscara estimada e o espectrograma original, seguida da aplicação da Transformada de Fourier de Tempo Curto Inversa, produzindo o sinal de áudio separado (Lu *et al.*, 2023).

2.1.7 Mel-Band RoFormer

O Mel-Band RoFormer (Mel-RoFormer) é uma evolução da arquitetura BS-RoFormer desenvolvida para aprimorar a separação de fontes musicais por meio da incorporação de conceitos baseados na percepção auditiva humana (Wang; Lu; Won, 2023). Enquanto o BS-RoFormer realiza a divisão do espectrograma em sub-bandas definidas por heurísticas e sem sobreposição, o Mel-RoFormer introduz um módulo de projeção em bandas Mel, permitindo que a representação espectral seja organizada de forma mais compatível com a sensibilidade do sistema auditivo humano (Wang; Lu; Won, 2023).

Essa abordagem fundamenta-se na observação de que a audição humana apresenta maior capacidade de discriminar pequenas variações em baixas frequências do que em frequências elevadas. Dessa forma, a escala Mel reorganiza os bins de frequência para que intervalos percebidos como igualmente espaçados em termos de altura sonora apresentem larguras equivalentes ao longo do espectro, proporcionando uma representação mais adequada para tarefas de análise e separação de sinais de áudio (Wang; Lu; Won, 2023).

2.1.7.1 Arquitetura

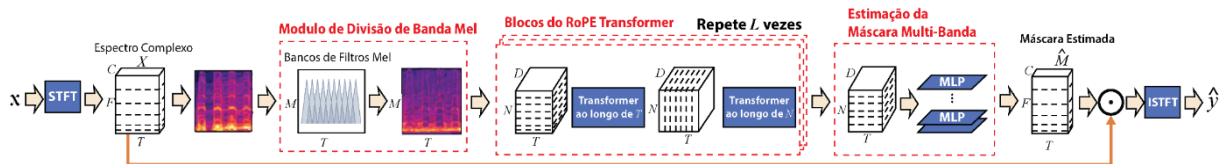
Uma das principais diferenças em relação ao BS-RoFormer está na utilização de sub-bandas sobrepostas. Nesse modelo, cada banda possui largura equivalente ao dobro da distância entre seu centro e o centro da banda anterior, fazendo com que metade de uma banda se sobreponha à metade da banda seguinte. Essa estratégia reduz descontinuidades entre bandas adjacentes e favorece uma representação espectral mais contínua (Wang; Lu; Won, 2023).

Além disso, o módulo de projeção em bandas Mel atua de maneira semelhante a um banco de filtros Mel, porém seus parâmetros são ajustados automaticamente durante o treinamento da rede por meio das camadas MLP (Wang; Lu; Won, 2023). Dessa forma, o próprio processo de aprendizagem otimiza a representação das diferentes regiões do espectro de frequência para a tarefa de separação de fontes musicais.

2.1.7.2 Fluxo de processamento

Na Figura 15 é apresentada uma visão geral do fluxo de processamento realizado pelo Mel-Band RoFormer. Inicialmente, o sinal de áudio é convertido para o domínio tempo-frequência por meio da STFT, produzindo um espectrograma complexo. Em seguida, essa representação é reorganizada pelo módulo de projeção em bandas Mel, responsável por agrupar as frequências de acordo com a escala Mel (Wang; Lu; Won, 2023).

Figura 15 - Funcionamento do Mel-RoFormer



Fonte: elaborado pelo autor (2026)

As representações obtidas são então processadas pelos blocos Transformer da arquitetura RoFormer, que empregam mecanismos de atenção para modelar simultaneamente dependências temporais e espectrais entre as diferentes regiões do espectrograma. Durante esse processo, as camadas MLP refinam as representações intermediárias e contribuem para o aprendizado dos filtros utilizados na projeção em bandas Mel (Wang; Lu; Won, 2023).

Ao final do processamento, as representações estimadas para cada banda são projetadas novamente para o domínio de frequência original. A partir dessas estimativas, o modelo reconstrói os espectrogramas correspondentes às diferentes fontes musicais. Por fim, cada espectrograma é convertido novamente para o domínio do tempo por meio da iSTFT, produzindo os sinais de áudio separados (Wang; Lu; Won, 2023).

2.1.7.2 Reconstrução do sinal

Após a estimativa das representações correspondentes às diferentes fontes musicais, cada banda Mel é projetada novamente para o domínio de frequência original por meio de uma projeção inversa. Como uma mesma frequência pode pertencer simultaneamente a mais de uma banda devido à sobreposição, o modelo calcula a média dos valores estimados para produzir a máscara complexa final. Essa estratégia proporciona uma reconstrução mais suave do espectrograma, reduzindo descontinuidades entre bandas e contribuindo para a diminuição de artefatos perceptíveis no áudio reconstruído (Wang; Lu; Won, 2023).

2.1.7.3 Desempenho

Os resultados experimentais apresentados pelos autores Wang, Lu e Won (2023) demonstram que a utilização da projeção em bandas Mel melhora o desempenho da separação de vocais, bateria e diversos outros instrumentos quando comparada ao BS-RoFormer. Entretanto, o modelo apresenta limitações na separação do instrumento baixo (*bass*), cujo processo de treinamento evolui de forma mais lenta e resulta em valores de *Signal-to-Distortion Ratio* (SDR) inferiores aos obtidos pela arquitetura baseada em bandas disjuntas. Segundo os autores, esse comportamento sugere que a escala Mel pode não representar de forma ideal as características espectrais de instrumentos concentrados predominantemente nas regiões de frequências mais baixas (Wang; Lu; Won, 2023).

2.1.8 *Signal-to-Distortion Ratio* (SDR)

O *Signal-to-Distortion Ratio* é uma métrica amplamente utilizada para avaliar a qualidade da separação de fontes musicais. Seu objetivo é quantificar o grau de semelhança entre o sinal estimado por um algoritmo e o sinal de referência (*ground truth*), considerando a quantidade de distorção introduzida durante o processo de separação (Lu *et al.*, 2023; Patel *et al.*, 2022).

De forma geral, o SDR representa a relação entre a energia do sinal desejado e a energia correspondente aos erros presentes na estimativa, incluindo

componentes provenientes de outras fontes, ruídos e demais distorções. Dessa forma, quanto maior o valor de SDR, maior é a fidelidade do sinal separado em relação ao sinal original de referência (Patel *et al.*, 2022; Vincent; Gribonval; Févotte, 2006).

Uma forma simplificada de representar o SDR é dada pela Equação 10 em que s representa o sinal de referência e $\hat{s}$ o sinal estimado pelo algoritmo de separação (Patel *et al.*, 2022).

$$SDR = 10 \log_{10} \left(\frac{\sum_n \|s(n)\|^2 + \epsilon}{\sum_n \|s(n) - \hat{s}(n)\|^2 + \epsilon} \right) \quad (10)$$

Na avaliação de modelos de separação de fontes, o SDR é frequentemente utilizado como uma das principais métricas de desempenho, permitindo comparar objetivamente diferentes abordagens e arquiteturas (Liu; Kong; Liu, 2021; Simpson; Roma; Plumbley, 2015). Em geral, modelos que apresentam maiores valores de SDR produzem separações com menor quantidade de interferências e artefatos, indicando melhor qualidade do áudio obtido (Défossez, 2022; Simpson; Roma; Plumbley, 2015).

2.2 Geração Automática de Legenda

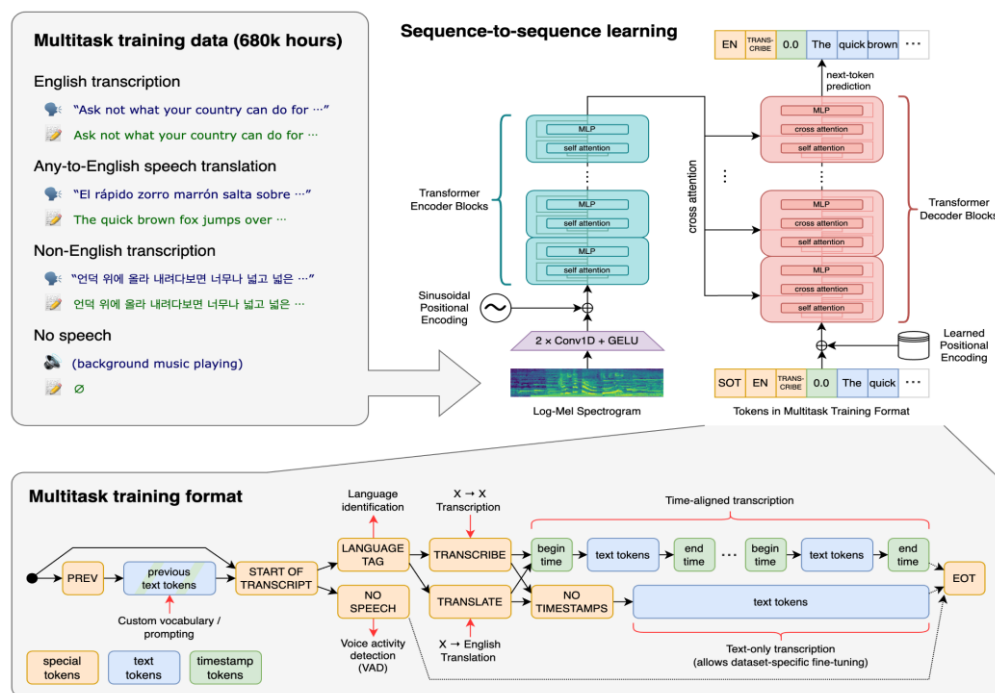
Esta seção apresenta os fundamentos teóricos necessários para a compreensão do processo de geração automática de legendas.

2.2.1 Whisper

O *Whisper* é um modelo de reconhecimento automático de fala chamado de *Automatic Speech Recognition* (ASR) desenvolvido pela OpenAI com o objetivo de realizar a transcrição de fala para texto de forma robusta em diferentes idiomas. O modelo foi treinado utilizando aproximadamente 680 mil horas de áudio coletadas da internet, abrangendo uma grande diversidade de idiomas, sotaques, condições acústicas e estilos de fala. Essa grande variedade de dados permitiu ao modelo apresentar elevada capacidade de generalização, tornando-o adequado para diferentes tarefas de reconhecimento de fala (Radford *et al.*, 2022; Ruiz, 2024).

O *Whisper* utiliza uma arquitetura Transformer do tipo encoder-decoder, conforme apresentado na seção 2.1.5. Inicialmente, o sinal de áudio é convertido em um espectrograma na escala Mel, que é utilizado como entrada do codificador (*encoder*). O codificador extrai representações do conteúdo acústico do sinal, enquanto o decodificador (*decoder*) utiliza essas representações para gerar sequencialmente os tokens correspondentes ao texto reconhecido, esse fluxo pode ser visualizado na Figura 16 (Li; Xu; Zhang, 2021).

Figura 16 - Visão Geral do *Whisper*



Fonte: Robust Speech Recognition via Large-Scale Weak Supervision.

Além da transcrição convencional, o *Whisper* também oferece suporte a diferentes tarefas, como identificação automática do idioma, tradução para o inglês e geração de marcações temporais (*timestamps*), permitindo associar cada segmento transcrito ao instante correspondente no áudio. Essas características tornam o modelo adequado para aplicações que exigem sincronização entre áudio e texto, como sistemas de legendagem automática e karaokê (Demirel; Ahlback; Thakkar, 2020; Radford *et al.*, 2022; Ruiz, 2024).

2.2.2 Distância de Levenshtein

A Distância de Levenshtein, também conhecida como distância de edição (*Edit Distance*), é uma métrica utilizada para quantificar o grau de diferença entre duas sequências de caracteres ou palavras. Seu princípio consiste em determinar o número mínimo de operações necessárias para transformar uma sequência em outra (Jurafsky; Martin, 2008; Radford *et al.*, 2022).

As operações consideradas pela métrica são substituição, quando um elemento é trocado por outro; inserção, quando um novo elemento é adicionado à sequência; e remoção, quando um elemento existente é excluído. Cada uma dessas operações possui custo unitário, e a distância corresponde à menor soma de custos necessária para converter a sequência de referência na sequência obtida (Jurafsky; Martin, 2008; Ruiz, 2024).

Como exemplo, considere as palavras "canto" e "canta". Para transformar a primeira na segunda é necessário apenas uma operação de substituição da letra "o" pela letra "a", resultando em uma distância de Levenshtein igual a 1 (Liu; Kong; Liu, 2021).

Por sua simplicidade e eficiência, a Distância de Levenshtein é amplamente empregada na avaliação de sistemas de reconhecimento automático de fala, correção ortográfica e processamento de linguagem natural (Demirel; Ahlbäck; Thakkar, 2020; Liu; Kong; Liu, 2021).

2.2.3 Word Error Rate (WER)

O *Word Error Rate* é uma das métricas mais utilizadas para avaliar o desempenho de sistemas de reconhecimento automático de fala. Seu objetivo é quantificar a diferença entre o texto gerado pelo modelo e a transcrição de referência, considerando os erros cometidos durante o processo de reconhecimento (Li; Xu; Zhang, 2021; Liu; Kong; Liu, 2021).

O cálculo do WER baseia-se na Distância de Levenshtein, contabilizando os três tipos de erros: substituições (Substitutions – S), inserções (Insertions – I) e remoções (Deletions – D). Esses erros são normalizados pelo número total de

palavras da transcrição de referência (N), conforme apresentado na Equação 11 (Demirel; Ahlbäck; Thakkar, 2020; Li; Xu; Zhang, 2021).

$$WER = \frac{S + D + I}{N} \quad (11)$$

O WER é geralmente expresso na forma decimal ou em porcentagem (Zhang *et al.*, 2020). Um valor igual a 0 (ou 0%) indica uma transcrição perfeita, enquanto valores maiores representam um número crescente de erros de reconhecimento. Como inserções também são contabilizadas, o WER pode assumir valores superiores a 1 (ou 100%) (Zhang *et al.*, 2020).

Apesar de sua ampla utilização, o WER apresenta limitações quando aplicado a idiomas que não utilizam separação explícita entre palavras. Nesses casos, a segmentação do texto pode influenciar significativamente o resultado da métrica, tornando mais adequada a utilização de métricas baseadas em caracteres, como o *Character Error Rate* (Demirel; Ahlbäck; Thakkar, 2020; Zhang *et al.*, 2020)

2.2.4 Character Error Rate (CER)

O *Character Error Rate* é uma métrica utilizada para avaliar a qualidade de sistemas de reconhecimento automático de fala a partir da comparação entre a transcrição gerada e o texto de referência em nível de caracteres (Radford *et al.*, 2022). Assim como o WER, seu cálculo é baseado na Distância de Levenshtein, considerando o número mínimo de inserções, remoções e substituições necessárias para transformar uma sequência de caracteres na outra (Jurafsky; Martin, 2008; Radford *et al.*, 2022).

O CER é obtido pela razão entre o número total de operações de edição e a quantidade de caracteres presentes na transcrição de referência, conforme apresentado na Equação 12.

$$CER = \frac{S + D + I}{N} \quad (12)$$

Embora as equações 11 e 12 apresentem a mesma formulação matemática, as métricas diferem quanto à unidade empregada na comparação das transcrições. No WER, as operações de inserção, remoção e substituição são contabilizadas sobre palavras, ao passo que no CER essas mesmas operações são realizadas sobre caracteres. Consequentemente, o termo N representa o número total de palavras na Equação 11 e o número total de caracteres na Equação 12 (Zhang *et al.*, 2020; Zhu; Gao, 2005).

Assim como no WER, um valor igual a 0 (zero) indica que a transcrição gerada corresponde exatamente ao texto de referência, enquanto valores maiores representam uma quantidade crescente de erros de reconhecimento (Jurafsky; Martin, 2008; Li; Xu; Zhang, 2021).

Uma das principais vantagens do CER é sua independência da segmentação em palavras, tornando essa métrica especialmente adequada para idiomas que não utilizam separação explícita entre palavras, como o japonês e o chinês. Por esse motivo, o CER é frequentemente empregado na avaliação de sistemas multilíngues (Jurafsky; Martin, 2008; Radford *et al.*, 2022; Li; Xu; Zhang, 2021; Zhang *et al.*, 2020).

3 METODOLOGIA

Nesta seção será abordado a prática implementada para a utilização das ferramentas visando a criação de faixas karaokê e suas legendas. Os códigos desenvolvidos para a realização dos experimentos e implementação do sistema estão disponíveis no repositório do projeto no GitHub: <https://github.com/PlantDragon/tcc-karaoke-ia>.

3.1 Separação de fontes musicais

Para a implementação da etapa de separação de fontes musicais foi utilizado o repositório Music-Source-Separation-Training (Solovyev; Stempkovskiy; Habruseva, 2026), desenvolvido por Roman Solovyev (ZFTurbo). A escolha dessa implementação deve-se ao fato de o projeto disponibilizar uma implementação consolidada de diferentes arquiteturas modernas para separação de fontes musicais, incluindo o modelo Mel-Band RoFormer, utilizado neste trabalho. Além da implementação do algoritmo, o repositório disponibiliza modelos previamente treinados, permitindo a utilização direta das arquiteturas propostas na literatura sem a necessidade de reproduzir integralmente o processo de treinamento.

Foi empregado os pesos do checkpoint KimberleyJensen edition. Esse checkpoint foi escolhido por apresentar desempenho superior na separação de vocais em relação a outras variantes disponibilizadas no mesmo repositório, alcançando um SDR de 10,98 dB para a separação de vocais, conforme informado pelos mantenedores do projeto (Solovyev; Stempkovskiy; Habruseva, 2026).

A utilização de um modelo pré-treinado foi motivada principalmente pelas elevadas exigências computacionais envolvidas no treinamento de modelos de grande porte para separação de fontes musicais. Conforme descrito no trabalho original que propôs a arquitetura BS-RoFormer, o treinamento do modelo completo foi realizado durante aproximadamente quatro semanas utilizando 16 GPUs NVIDIA A100 de 80 GB, com um tamanho total de lote (*batch size*) igual a 128, além da utilização de técnicas de treinamento em precisão mista (FP16/FP32), *checkpointing* para redução do consumo de memória, otimização AdamW e *Exponential Moving Average* (EMA).

O modelo resultante possui aproximadamente 93,4 milhões de parâmetros treináveis (LU *et al.*, 2023).

Assim, os esforços deste trabalho concentraram-se na integração do modelo ao fluxo de processamento proposto e na avaliação de sua aplicação à tarefa de separação de fontes musicais.

3.1.1 Criação do ambiente virtual para a separação de fontes musicais

Para a execução do modelo de separação de fontes musicais Mel-Band RoFormer, foi necessária a criação de um ambiente virtual dedicado, visando maior reprodutibilidade ao projeto, uma vez que todas as dependências permanecem isoladas do sistema operacional e podem ser instaladas em outras máquinas utilizando as mesmas versões das bibliotecas, evitando conflitos com outras instalações do sistema operacional, bem como facilitando a replicação dos experimentos desenvolvidos neste trabalho.

A utilização de um ambiente virtual dedicado proporciona maior reprodutibilidade ao projeto, uma vez que todas as dependências permanecem isoladas do sistema operacional e podem ser instaladas em outras máquinas utilizando as mesmas versões das bibliotecas.

Inicialmente, foi instalada a distribuição Anaconda, escolhida por facilitar o gerenciamento de ambientes virtuais e dependências em projetos de aprendizado de máquina. Em seguida, foi criado um novo ambiente utilizando o Python na versão 3.11, uma vez que versões mais recentes, como a 3.13, ainda não possuem compatibilidade completa com diversas bibliotecas empregadas pelo modelo.

Com o ambiente configurado, procedeu-se à instalação da versão do PyTorch compatível com a placa gráfica utilizada e com a versão do CUDA Toolkit instalada no computador. Na operação foi empregada a versão CUDA 12.1, permitindo que o processamento fosse realizado diretamente pela GPU.

A GPU utilizada é baseada na arquitetura Pascal, pertencente a uma geração anterior às arquiteturas atualmente empregadas em aplicações de aprendizado profundo. Embora plenamente compatível com a execução do modelo, essa arquitetura não dispõe de recursos presentes em GPUs mais recentes, como Tensor Cores e suporte à implementação otimizada do mecanismo de atenção

conhecido como *FlashAttention*. Dessa forma, determinadas otimizações computacionais utilizadas por modelos Transformer modernos não puderam ser exploradas.

Posteriormente, foram instaladas as bibliotecas necessárias ao funcionamento do projeto. Embora o repositório disponibilize um arquivo “requirements.txt” contendo as dependências do sistema, sua utilização não foi suficiente para obter um ambiente funcional, em razão de incompatibilidades entre versões de alguns pacotes e da necessidade de utilizar versões específicas do PyTorch compatíveis com a GPU empregada neste trabalho. Dessa forma, as bibliotecas foram instaladas individualmente, permitindo o controle das versões de cada dependência e garantindo a correta execução do modelo de separação de fontes musicais.

Entre as principais bibliotecas instaladas destacam-se a NumPy, utilizada para operações matriciais, Librosa e SoundFile para processamento de sinais de áudio, além de outras dependências requeridas pelo modelo.

Além das bibliotecas Python, também foi necessária a instalação do FFmpeg, ferramenta responsável pela leitura e conversão de diferentes formatos de áudio. Sua presença é indispensável tanto para a separação das fontes musicais quanto para etapas posteriores do processamento, como a geração automática de legendas utilizando modelos de reconhecimento de fala.

Após a instalação das dependências, foi realizado o download do modelo pré-treinado Mel-Band RoFormer, disponibilizado pelos autores do projeto (Solovyev; Stempkovskiy; Habruseva, 2026). O arquivo de pesos (.ckpt) foi armazenado em um diretório específico dentro do projeto, enquanto o arquivo de configuração (.yaml) foi mantido na pasta configs do repositório original.

A execução do modelo é realizada informando três elementos principais: o arquivo de configuração do modelo, o checkpoint contendo os pesos treinados e o diretório contendo os arquivos de áudio de entrada. Durante a inicialização, o programa identifica automaticamente a disponibilidade de uma GPU compatível com CUDA, carregando o modelo na memória da placa gráfica para acelerar o processamento.

Como etapa de validação da instalação, foi verificada a disponibilidade do CUDA através da biblioteca PyTorch, para confirmar que o ambiente estava corretamente configurado para utilizar a aceleração por hardware.

Com o ambiente configurado, o sistema estava pronto para a realização da tarefa de separação das fontes musicais.

3.1.2 Script de configuração da inferência

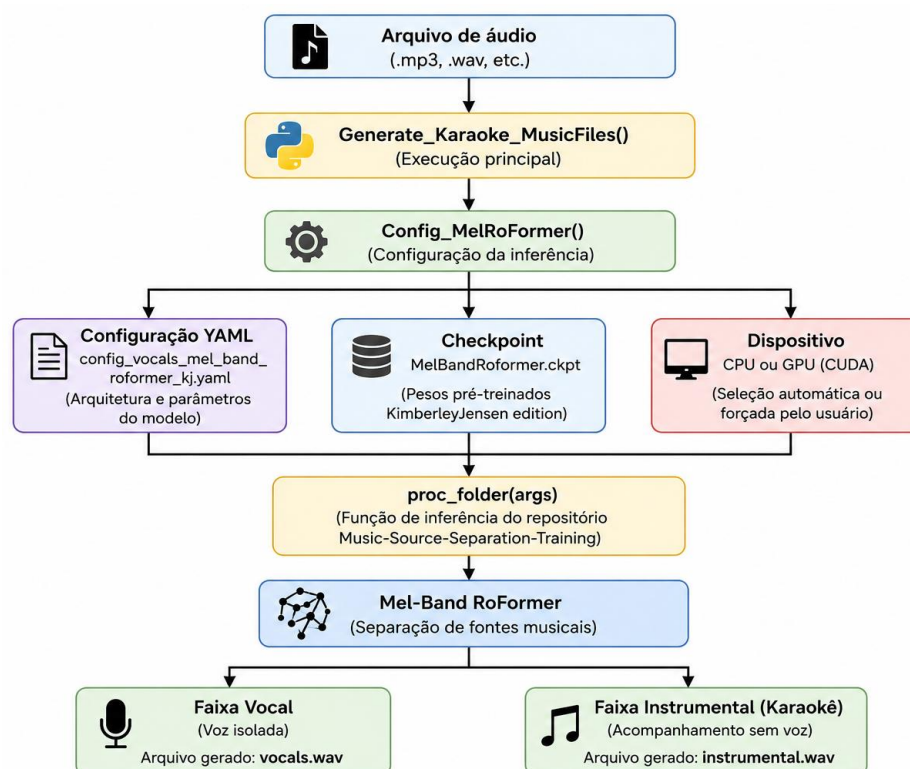
Após a configuração do ambiente de execução, foi desenvolvido um script em Python denominado `MelRoFormer_config.py`, qual pode ser visualizado no Apêndice A, responsável por automatizar a configuração e a inicialização do processo de inferência do modelo Mel-Band RoFormer.

O desenvolvimento desse script teve como objetivo concentrar, em um único ponto de configuração, todos os parâmetros necessários para a execução do modelo, simplificando sua utilização e reduzindo a necessidade de alterações diretamente nos arquivos da implementação original disponibilizada pelo repositório Music-Source-Separation-Training.

Inicialmente, o script adiciona dinamicamente o diretório do repositório ao caminho de busca de módulos do Python, permitindo o acesso às funções responsáveis pela execução da inferência. Em seguida, são definidos os principais parâmetros utilizados pelo modelo, incluindo o arquivo de configuração da arquitetura, o checkpoint contendo os pesos pré-treinados, os diretórios destinados aos arquivos de entrada e saída e a seleção do dispositivo de processamento (CPU ou GPU).

Esses parâmetros são organizados em uma estrutura de configuração e posteriormente encaminhados à função de inferência disponibilizada pelo próprio repositório, responsável pelo carregamento do modelo, processamento dos arquivos de áudio e geração das fontes musicais separadas, o fluxo é visualmente demonstrado na Figura 17.

Figura 17 - Fluxo de configuração e execução da inferência para geração das faixas vocal e instrumental utilizando a arquitetura Mel-Band RoFormer



Fonte: Elaborado pelo autor (2026).

A utilização deste script permitiu centralizar todas as configurações necessárias para a execução da ferramenta, facilitando sua reutilização durante os experimentos realizados, neste trabalho, e reduzindo a possibilidade de inconsistências decorrentes da configuração manual dos parâmetros.

3.2 Geração automática de legendas

Após a obtenção da faixa vocal por meio da etapa de separação de fontes musicais, tornou-se necessário converter o conteúdo falado em texto para possibilitar a geração automática das legendas sincronizadas utilizadas. Para essa finalidade, foi empregado o modelo *Whisper*, desenvolvido pela OpenAI, por se tratar de uma solução de reconhecimento automático de fala (*Automatic Speech Recognition – ASR*) baseada em arquiteturas Transformer e treinada sobre uma grande diversidade de idiomas, sotaques e condições acústicas.

A escolha do *Whisper* deve-se à elevada qualidade das transcrições produzidas pelo modelo, além da disponibilidade de uma implementação oficial em código aberto, que oferece modelos pré-treinados de diferentes capacidades e permite sua integração direta ao fluxo de processamento desenvolvido. Essas características possibilitam a realização da transcrição automática das faixas vocais sem a necessidade de treinamento adicional do modelo.

Embora exista uma implementação derivada denominada *WhisperX*, que incorpora melhorias como alinhamento forçado e refinamento das marcações temporais em nível de palavra, essa etapa depende da disponibilidade de modelos de alinhamento específicos para cada idioma (Ruiz, 2024).

Como o objetivo deste trabalho é permitir a geração automática de legendas para músicas em diferentes idiomas, buscou-se evitar a adoção de uma solução cuja principal vantagem estivesse restrita a um conjunto limitado de línguas (Bain, 2022; Bain *et al.*, 2023). Dessa forma, optou-se pela utilização da implementação oficial do *Whisper*, que oferece suporte multilíngue mais abrangente e permite a geração das legendas sincronizadas necessárias ao fluxo de processamento proposto, ainda que suas marcações temporais em nível de palavra possam apresentar menor precisão quando comparadas às obtidas pelo alinhamento forçado empregado no *WhisperX*.

3.2.1 Configuração do ambiente para o *Whisper*

Para a etapa de transcrição automática das vozes separadas, foi criado um ambiente virtual independente utilizando o gerenciador de ambientes Conda. Diferentemente do ambiente empregado na separação de fontes musicais, que foi configurado especificamente para permitir a utilização da aceleração por GPU, o ambiente destinado ao *Whisper* foi desenvolvido de forma independente. O objetivo é evitar conflitos entre dependências e possibilitar a utilização da versão mais recente da implementação oficial do *Whisper* e de suas dependências. Essa separação também contribui para a reprodutibilidade do sistema, permitindo que cada etapa do processamento seja configurada e executada de maneira independente.

Inicialmente, foi criado um ambiente virtual utilizando o interpretador Python na versão 3.11, escolhida por apresentar ampla compatibilidade com a implementação

oficial do *Whisper* e suas bibliotecas. Em seguida, foram instalados o framework PyTorch em sua versão destinada à execução em CPU e as demais dependências necessárias para a utilização do *Whisper*.

Optou-se pela execução do *Whisper* utilizando apenas o processador (CPU) pois, a utilização da placa gráfica disponível exige versões específicas do framework PyTorch, que são incompatíveis com as versões mais recentes do *Whisper*.

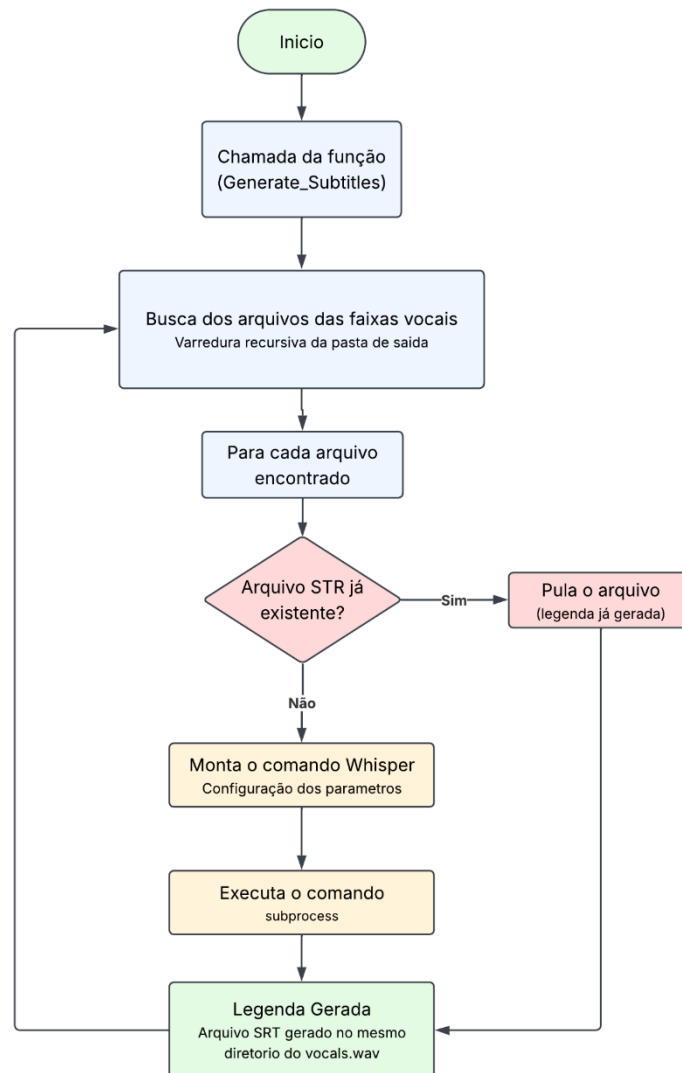
Posteriormente, foi instalada a implementação oficial do *Whisper*, disponibilizada pela OpenAI, juntamente com suas dependências. Ao longo da execução da transcrição, o FFmpeg, previamente instalado no sistema, foi utilizado para realizar a leitura e o pré-processamento dos arquivos de áudio, convertendo-os para o formato esperado pela implementação do *Whisper*. Na sequência, foi realizado o download do modelo pré-treinado Large-v3, selecionado por apresentar a maior capacidade de reconhecimento entre as variantes disponibilizadas pela OpenAI, priorizando a qualidade da transcrição em detrimento do tempo de processamento. O modelo foi empregado na geração automática das legendas a partir dos arquivos de voz obtidos na etapa de separação de fontes musicais.

3.2.2 Desenvolvimento do script para geração das legendas

Após a configuração do ambiente de execução, foi desenvolvido um script em Python (Apêndice B) responsável por automatizar a geração das legendas a partir das faixas vocais produzidas na etapa de separação de fontes musicais. O script foi projetado para operar diretamente sobre os arquivos de voz gerados pelo processo anterior, estabelecendo a integração entre as duas etapas do fluxo de processamento.

Inicialmente, o script realiza uma busca recursiva no diretório de saída da etapa de separação de fontes, localizando todos os arquivos de voz gerados pelo modelo. Para cada arquivo encontrado, é verificada a existência de uma legenda previamente criada. Caso o arquivo de legenda já exista, seu processamento é descartado, evitando a execução desnecessária da inferência, como ilustrado na Figura 18.

Figura 18 - Fluxograma do script de geração de legendas com *Whisper*



Fonte: Elaborado pelo autor (2026).

Quando uma nova transcrição é necessária, o script reúne automaticamente todos os parâmetros de configuração utilizados na execução da implementação oficial do *Whisper*, incluindo o dispositivo de processamento, o formato do arquivo de saída e os parâmetros relacionados à organização visual das legendas, a Tabela 1 mostra os principais parâmetros utilizados. Essas configurações são utilizadas para construir o comando responsável pela execução do processo de transcrição.

Tabela 1 - Principais parâmetros utilizados na configuração do *Whisper*

Parâmetro	Valor utilizado	Finalidade
Model	large-v3	Seleciona a variante pré-treinada do <i>Whisper</i> utilizada na transcrição.
Device	Cpu	Define o dispositivo utilizado durante a inferência.
output_format	Srt	Especifica o formato do arquivo de saída das legendas (SubRip Subtitle).
output_dir	Diretório do arquivo de entrada	Define o local onde o arquivo de legenda será armazenado.
word_timestamps	True	Habilita a estimativa das marcações temporais em nível de palavra, permitindo maior precisão na sincronização das legendas.
highlight_words	True	Destaca individualmente as palavras no arquivo SRT conforme suas marcações temporais.
max_line_width	45	Limita cada linha da legenda a aproximadamente 45 caracteres, melhorando sua legibilidade.
max_line_count	2	Limita cada legenda a, no máximo, duas linhas, conforme recomendações de legendagem.

hallucination_silence_threshold	2.0 (caso utilize)	Define um intervalo mínimo de silêncio para que possíveis alucinações do modelo sejam descartadas, reduzindo a geração de legendas em trechos sem conteúdo vocal.
---------------------------------	--------------------	---

Fonte: Elaborado pelo autor (2026).

A comunicação com o *Whisper* é realizada por meio da interface de linha de comando disponibilizada pela própria implementação, sendo sua execução iniciada pelo módulo *subprocess* da linguagem Python. A utilização do interpretador obtido por meio de *sys.executable* garante que o modelo seja executado utilizando o mesmo ambiente virtual no qual o *script* foi iniciado. Evita, assim, incompatibilidades entre diferentes instalações do Python e de suas dependências.

Após a conclusão da inferência, o arquivo de legenda é armazenado no mesmo diretório da respectiva faixa vocal, preservando a organização dos arquivos gerados durante todo o fluxo de processamento. Dessa forma, cada música permanece organizada em um único diretório contendo os arquivos correspondentes à faixa vocal, à faixa instrumental e à legenda sincronizada.

O desenvolvimento desse *script* permitiu automatizar completamente a etapa de transcrição, reduzindo a necessidade de intervenção do usuário e garantindo que todos os arquivos fossem processados de forma padronizada, utilizando sempre os mesmos parâmetros de configuração.

4 APRESENTAÇÃO DOS RESULTADOS

Neste capítulo são apresentados os resultados obtidos a partir da implementação do sistema proposto para geração automática de faixas de karaokê, são descritas as condições em que os experimentos foram realizados e, em seguida, aparecem os resultados referentes à separação de fontes musicais, à geração automática de legendas e ao desempenho computacional do sistema. Por fim, é realizada uma análise dos resultados obtidos, destacando as principais características observadas durante a execução da ferramenta.

4.1 Ambiente Experimental

Os experimentos foram realizados em um computador pessoal executando o sistema operacional Windows. A configuração utilizada durante o desenvolvimento e os testes da ferramenta é apresentada na Tabela 2.

Tabela 2 - Configuração do ambiente experimental

Componente		Especificação
Sistema Operacional		Windows 10 Pro x64
Processador	Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz	
Memória RAM	16 GB	
GPU	NVIDIA GeForce GTX 1050 4 GB	
CUDA Version	13.0	
CUDA Toolkit	12.1	
Python	3.11.15	
Pytorch	2.3.1+cu121	

Fonte: Elaborado pelo autor (2026).

Conforme discutido na metodologia, a utilização da GPU foi empregada apenas durante a etapa de separação de fontes musicais. A etapa de geração automática de legendas foi executada em CPU devido às limitações de compatibilidade entre as versões do framework PyTorch requeridas pelo modelo *Whisper* e a arquitetura da placa gráfica utilizada neste trabalho.

4.2 Separação de Fontes Musicais

A etapa de separação de fontes musicais foi executada utilizando o modelo pré-treinado Mel-Band RoFormer, conforme descrito no Capítulo 3. Para cada arquivo de entrada foram geradas duas saídas, uma contendo apenas os vocais e outra contendo o acompanhamento instrumental, utilizadas posteriormente pela etapa de geração automática de legendas e pela composição da faixa de karaokê.

4.2.1 Resultados da separação

Apesar da elevada qualidade da separação obtida na maior parte das músicas analisadas, foram observadas algumas situações em que o modelo apresentou limitações. As principais ocorreram em trechos nos quais a voz era utilizada predominantemente como um elemento instrumental, sem desempenhar a função principal de melodia vocal. Nesses casos, mesmo quando a vocalização fazia parte da textura instrumental da música, o modelo tendia a classificá-la como componente vocal, comportamento coerente com o objetivo para o qual foi treinado, que consiste em separar sinais produzidos pela voz humana dos demais instrumentos musicais.

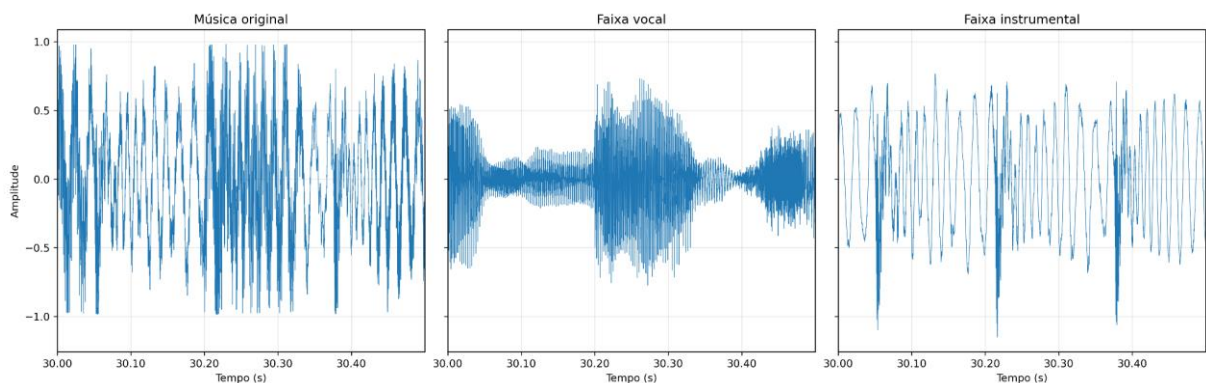
Em algumas músicas, trechos executados por instrumentos de sopro com características tímbricas semelhantes às da voz humana, como a flauta, foram parcialmente incorporados à faixa vocal. Da mesma forma, músicas que utilizam *backing vocals* de forma intensa, atuando quase como elementos instrumentais da composição, apresentaram maior quantidade de artefatos durante a separação, evidenciando a dificuldade do modelo em distinguir, nesses casos, a função musical desempenhada por cada componente sonoro.

Esse comportamento é esperado em modelos de separação de fontes baseados em aprendizado profundo, uma vez que a separação completa entre sinais simultâneos constitui um problema inerentemente complexo e dependente das características espectrais de cada gravação.

De maneira geral, as faixas instrumentais apresentaram qualidade suficiente para a composição da versão karaokê proposta.

Para ilustrar os resultados obtidos, a Figura 19 apresenta um trecho de aproximadamente 0,5 s das formas de onda correspondentes ao áudio original, à faixa vocal e à faixa instrumental geradas pelo modelo.

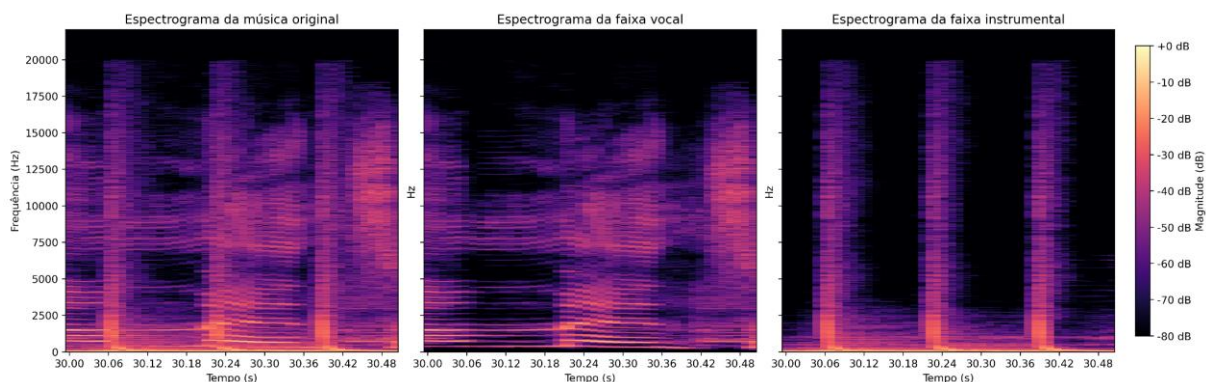
Figura 19 - Comparação das formas de onda após a separação de fontes musicais



Fonte: Elaborado pelo autor (2026).

Em seguida, a Figura 20 apresenta os espectrogramas referentes ao mesmo trecho temporal, permitindo comparar visualmente a distribuição espectral de energia entre os sinais original e separados.

Figura 20 - Comparação espectral dos sinais obtidos após a separação de fontes



Fonte: Elaborado pelo autor (2026).

A análise visual dos espectrogramas evidencia que o modelo foi capaz de isolar de maneira satisfatória os principais componentes vocais da música, preservando simultaneamente a maior parte dos elementos instrumentais na faixa complementar. Observa-se que as regiões de maior concentração de energia associadas à voz foram predominantemente transferidas para a faixa vocal, enquanto os instrumentos permaneceram concentrados na faixa instrumental, indicando uma separação coerente entre as duas componentes.

4.2.2 Desempenho computacional

Com o objetivo de avaliar o desempenho computacional da etapa de separação de fontes musicais, foram realizados experimentos comparando os tempos de processamento utilizando CPU e GPU. Para isso, foram selecionadas três músicas de diferentes características, processadas individualmente e, posteriormente, em um único lote (*batch*) contendo os três arquivos simultaneamente.

Cada configuração foi executada três vezes em ambos os dispositivos de processamento, sendo considerado o tempo médio obtido em cada condição de teste. Os experimentos foram realizados utilizando o mesmo modelo pré-treinado, mantendo inalterados todos os demais parâmetros de configuração, de forma que a única variável analisada fosse o dispositivo de processamento empregado durante a inferência.

4.2.2.1 Resultados obtidos

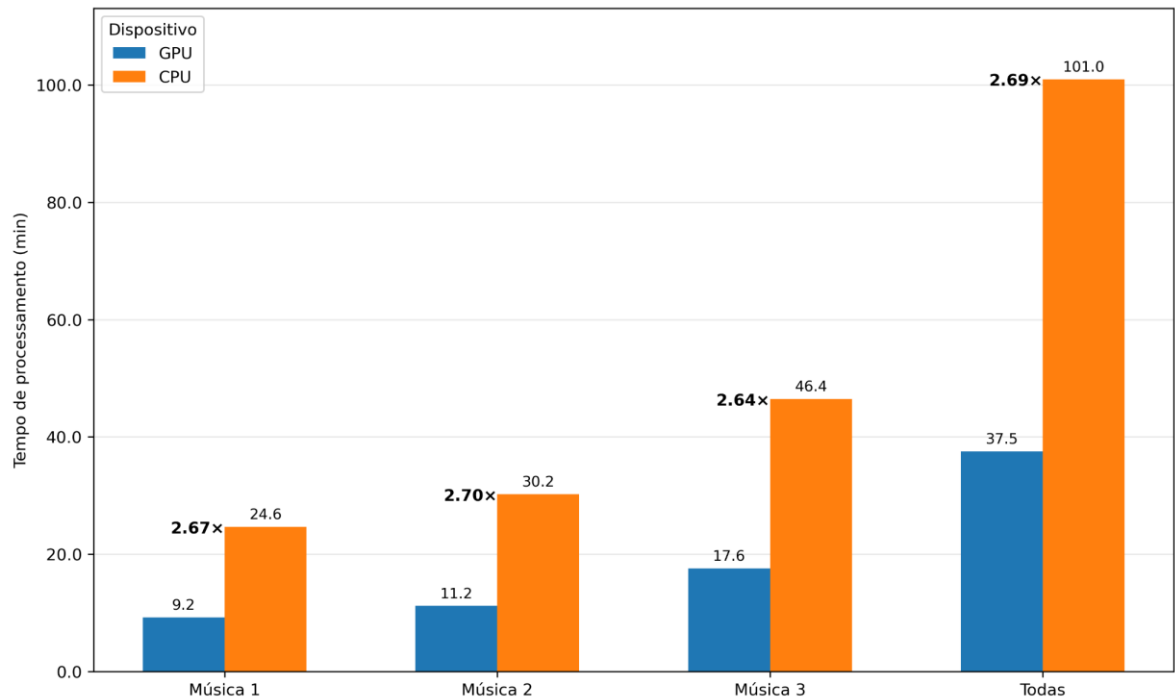
A Tabela 3 apresenta os tempos médios de processamento obtidos durante os experimentos, enquanto a Figura 21 apresenta a comparação gráfica entre os tempos de execução em CPU e GPU.

Tabela 3 - Configuração do ambiente experimental

	Duração (min)	CPU		GPU		Speedup
		média (min)	desvio padrão(min)	média (min)	desvio padrão(min)	CPU/GPU
Música 1	3.23	24.62	0.73	9.22	1.92	2.67
Música 2	4.03	30.19	0.58	11.17	2.48	2.70
Música 3	6.38	46.42	0.40	17.55	3.97	2.64
Todas	13.64	100.95	2.14	37.50	8.38	2.69

Fonte: Elaborado pelo autor (2026).

Figura 21 - Comparação espectral dos sinais obtidos após a separação de fontes



Fonte: Elaborado pelo autor (2026).

Os resultados demonstram uma redução significativa no tempo de processamento quando utilizada a aceleração por GPU em todas as condições avaliadas. O ganho de desempenho foi observado tanto no processamento individual

das músicas quanto na execução em lote, evidenciando a eficiência da utilização da GPU durante a inferência do modelo Mel-Band RoFormer.

Observa-se que o *speedup* permaneceu praticamente constante entre todos os experimentos, variando entre 2,64 vezes a 2,70 vezes, com média aproximada de 2,68 vezes. Esse comportamento indica que o ganho proporcionado pela GPU se mantém independentemente da música processada ou da quantidade de arquivos executados em sequência.

No processamento individual, os tempos médios variaram de 24,62 minutos a 46,42 minutos em CPU e de 9,22 minutos a 17,55 minutos em GPU. Quando as três músicas foram processadas em um único lote, o tempo total passou de 100,95 minutos em CPU para 37,50 minutos em GPU, mantendo praticamente o mesmo fator de aceleração observado nos demais experimentos.

O tempo necessário para o processamento em lote foi praticamente equivalente à soma dos tempos obtidos para o processamento individual das músicas, indicando que a execução em lote constitui uma forma de automatizar o processamento de múltiplos arquivos, sem alterar significativamente o tempo total de inferência do modelo.

Ressalta-se que os experimentos foram realizados utilizando uma placa gráfica NVIDIA GeForce GTX 1050 baseada na arquitetura Pascal. Por se tratar de uma arquitetura anterior às gerações mais recentes de GPUs NVIDIA, o equipamento não possui suporte às otimizações atualmente empregadas em modelos baseados em Transformers, como o *Flash Attention*. Durante a execução do modelo foram utilizadas implementações alternativas do mecanismo de atenção compatíveis com a arquitetura disponível, de modo que os tempos obtidos representam um cenário conservador de desempenho.

Apesar dessas limitações, os resultados evidenciam que a utilização da GPU reduz significativamente o tempo necessário para a separação de fontes musicais, tornando a ferramenta mais adequada para aplicações práticas e confirmando a importância da aceleração por *hardware* para a execução de modelos de grande porte baseados em Transformers.

4.3 Geração Automática de Legendas

A etapa de geração automática de legendas foi realizada a partir das faixas vocais produzidas durante a separação de fontes musicais. Para cada faixa foi gerado automaticamente um arquivo de legendas no formato SRT, contendo as marcações temporais e o texto reconhecido pelo modelo *Whisper*.

4.3.1 Resultados da geração automática de legendas

A Figura 22 apresenta um exemplo de uma das legendas produzidas automaticamente para uma das músicas processadas, evidenciando a correspondência entre o conteúdo textual reconhecido e os trechos da faixa vocal.

Figura 22 - Exemplo de legenda gerada automaticamente no formato SRT

```

1 1
2 00:00:22,980 --> 00:00:24,220
3 <u>In</u> all my years of living It isn't very often
4 that I get pissed off I try to chill with the
5
6 2
7 00:00:24,220 --> 00:00:25,460
8 In <u>all</u> my years of living It isn't very often
9 that I get pissed off I try to chill with the
10
11 3
12 00:00:25,460 --> 00:00:25,760
13 In all <u>my</u> years of living It isn't very often
14 that I get pissed off I try to chill with the
15
16 4
17 00:00:25,760 --> 00:00:26,420
18 In all my <u>years</u> of living It isn't very often
19 that I get pissed off I try to chill with the
20
21 5
22 00:00:26,420 --> 00:00:26,920
23 In all my years <u>of</u> living It isn't very often
24 that I get pissed off I try to chill with the
25
26 6
27 00:00:26,920 --> 00:00:27,500
28 In all my years of <u>living</u> It isn't very often
29 that I get pissed off I try to chill with the

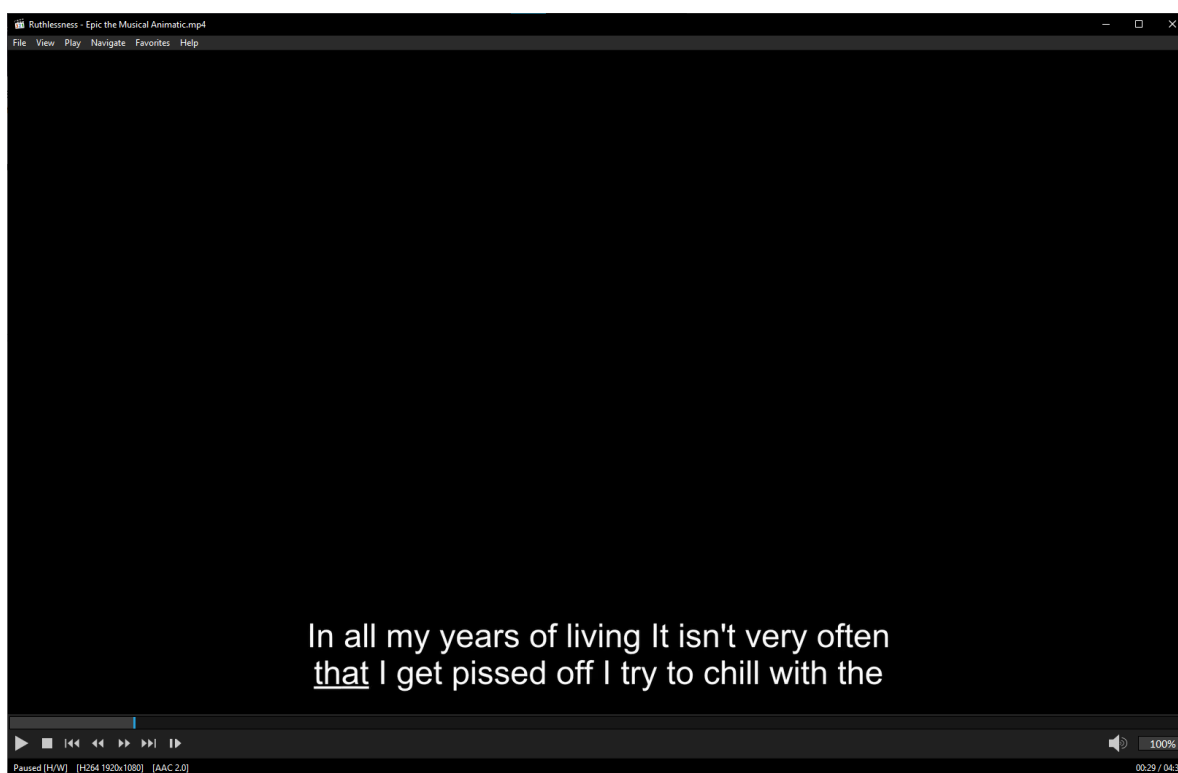
```

Fonte: Elaborado pelo autor (2026).

A legenda gerada preserva a divisão em trechos sincronizados com a música, permitindo sua utilização durante a reprodução da faixa instrumental sem a necessidade de sincronização manual.

A Figura 23 apresenta um exemplo da utilização da legenda gerada automaticamente durante a reprodução da faixa instrumental, ilustrando o resultado obtido pela integração entre as etapas de separação de fontes musicais e transcrição automática.

Figura 23 - Exemplo da sincronização entre a legenda gerada automaticamente e a reprodução da faixa instrumental



Fonte: Elaborado pelo autor (2026).

4.3.2 Avaliação dos resultados

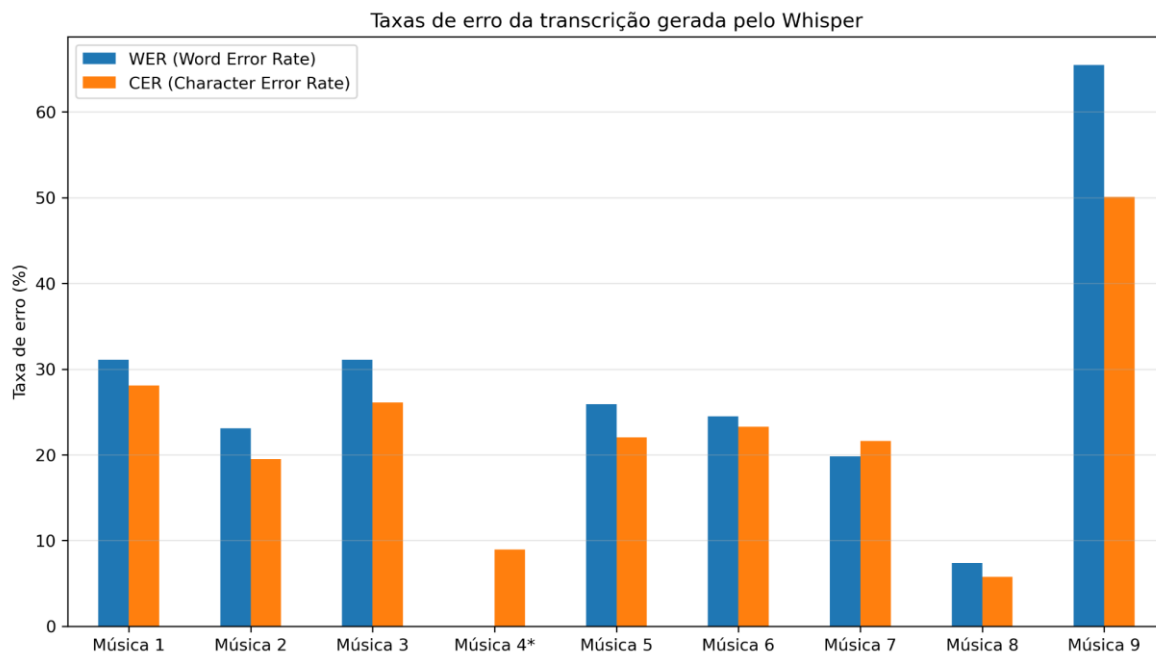
Para avaliar quantitativamente a qualidade das transcrições produzidas pelo *Whisper*, foram calculadas as métricas *Word Error Rate* (WER) e *Character Error Rate* (CER), amplamente utilizadas na avaliação de sistemas de reconhecimento automático de fala. Para idiomas que não apresentam separação explícita entre palavras, como o japonês, foi utilizada apenas a métrica CER, uma vez que o cálculo do WER depende da segmentação do texto em palavras.

A Tabela 4 apresenta os resultados obtidos para as músicas analisadas, enquanto a Figura 24 apresenta graficamente as taxas de erro observadas.

4Tabela 4 - Resultados das métricas WER e CER

Música	Idioma	CER (%)	WER (%)	Word hit rate (%)
Música 1	Inglês	28,11	31,08	88,18
Música 2	Português	19,48	23,12	87,44
Música 3	Inglês	26,09	31,11	90,56
Música 4	Japonês	8,94	—	—
Música 5	Inglês	22,04	25,92	76,62
Música 6	Português	23,28	24,48	93,71
Música 7	Inglês	21,64	19,83	86,12
Música 8	Inglês	5,79	7,39	95,61
Música 9	Português	50,09	65,47	58,99

Fonte: Elaborado pelo autor (2026).

Figura 24 - Taxas de erro das transcrições geradas pelo *Whisper*

* WER não calculado para idiomas sem separação explícita entre palavras.

Fonte: Elaborado pelo autor (2026).

Os resultados demonstram que o *Whisper* foi capaz de produzir transcrições com baixas taxas de erro na maioria das músicas. Entretanto, a análise qualitativa evidenciou que essas métricas, isoladamente, não representam todos os aspectos relevantes para a utilização das legendas em aplicações de karaokê.

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver um sistema capaz de automatizar a criação de versões karaokê a partir de músicas comerciais, integrando técnicas de separação de fontes musicais e reconhecimento automático de fala.

Os resultados obtidos demonstram que o sistema foi capaz de gerar automaticamente as faixas necessárias para a composição de uma versão karaokê, produzindo resultados satisfatórios quanto à qualidade da separação de fontes e da transcrição automática. Entretanto, o tempo necessário para a etapa de separação musical, mesmo utilizando aceleração por GPU, ainda constitui um fator limitante para aplicações que exijam processamento em tempo real ou próximo do tempo real.

Além do desempenho obtido na separação das fontes musicais, verificou-se que a utilização do *Whisper* permitiu automatizar a geração de legendas sincronizadas com qualidade suficiente para a aplicação proposta. Embora tenham sido observadas limitações relacionadas à presença de backing vocals, vocalizações e outras situações que reduziram a legibilidade das legendas em alguns trechos.

Como principal contribuição deste trabalho, destaca-se a integração de modelos de inteligência artificial pré-treinados em uma solução única capaz de automatizar etapas que tradicionalmente demandam elevado esforço manual. A combinação entre a separação de fontes musicais e a transcrição automática demonstrou ser uma alternativa viável para reduzir significativamente o tempo necessário para a produção dos principais elementos que compõem uma versão karaokê.

De modo geral, os resultados obtidos evidenciam o potencial da aplicação de técnicas modernas de inteligência artificial na automação de processos relacionados ao processamento digital de áudio, contribuindo para o desenvolvimento de ferramentas capazes de simplificar a produção de conteúdo musical e servir como base para futuras pesquisas na área.

5.1 Sugestões para trabalhos futuros

Como continuidade deste trabalho, sugere-se a realização de um estudo comparativo entre técnicas clássicas de Processamento Digital de Sinais e modelos

baseados em Inteligência Artificial aplicados à separação de fontes musicais, avaliando aspectos como qualidade da separação, desempenho computacional e adequação à geração de versões karaokê.

Uma outra possibilidade consiste na investigação de modelos capazes de separar a voz principal (*lead vocal*) dos *backing vocals*, reduzindo a sobreposição de diferentes linhas vocais durante a etapa de transcrição automática e, conseqüentemente, melhorando a legibilidade das legendas geradas.

Também podem ser exploradas técnicas de pós-processamento das legendas, visando à correção de erros de transcrição, ao tratamento de vocalizações e à remoção de trechos indesejados, contribuindo para aumentar a qualidade das legendas produzidas automaticamente.

Além disso, recomenda-se investigar estratégias para reduzir o tempo de processamento da etapa de separação de fontes, por meio da utilização de modelos mais eficientes, otimizações de inferência ou hardware com maior capacidade computacional, ampliando a viabilidade do sistema em aplicações que demandem menor tempo de resposta.

Por fim, sugere-se a integração das etapas desenvolvidas neste trabalho em uma aplicação com interface gráfica destinada ao usuário final, permitindo automatizar todo o processo de geração de versões karaokê em um único ambiente.

REFERÊNCIAS

- BAIN, M. **WhisperX**: Automatic Speech Recognition with Word-level Timestamps (& Diarization). GitHub, 2022. Disponível em: <https://github.com/m-bain/whisperX>. Acesso em: 02 jul. 2026.
- BAIN, M. et al. WhisperX: Time-Accurate Speech Transcription of Long-Form Audio. In: INTERSPEECH, 2023. **Anais...** [S. l.]: ISCA, 2023. p. 1-5. DOI: <https://doi.org/10.48550/arXiv.2303.0074>. Disponível em: <https://arxiv.org/abs/2303.00747>. Acesso em: 17 de ago. 2024.
- DÉFOSSEZ, Alexandre. Hybrid spectrogram and waveform source separation. In: MDX WORKSHOP, 2021. **Anais [...]**. [S. l.], 30 ago. 2022. DOI: [10.48550/arXiv.2111.03600](https://arxiv.org/abs/2111.03600). Disponível em: <https://arxiv.org/abs/2111.03600>. Acesso em: 14 out. 2024.
- DEMIREL, E.; AHLBÄCK, S.; THAKKAR, S. Automatic lyrics transcription using dilated convolutional neural networks with self-attention. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2020, Glasgow, Reino Unido. **Anais [...]**. Glasgow, Reino Unido: IEEE, 2020. DOI: [10.1109/IJCNN48605.2020.9207052](https://ieeexplore.ieee.org/document/9207052). Disponível em: <https://ieeexplore.ieee.org/document/9207052>. Acesso em: 13 ago. 2024.
- FRAHM. **Descubra qual é a influência da música no comportamento humano**, 2024. Disponível em: <https://frahm.com.br/a-influencia-da-musica-no-comportamento-humano/>. Acesso em: 03 mar. 2026.
- HAYKIN, Simon. **Redes Neurais: Princípios e Prática**. 2. ed. Porto Alegre: Bookman, 2007. E-book. ISBN 9788577800865. Disponível em: <https://app.minhabiblioteca.com.br/reader/books/9788577800865/>. Acesso em: 20 ago. 2025.
- HUANG, P. S. et al. Singing-voice separation from monaural recordings using deep recurrent neural networks. In: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE, 15., 2014, Taipei. **Anais [...]**. Taipei: ISMIR, 2014. p. 477-482. Disponível em: <https://archives.ismir.net/ismir2014/paper/000154.pdf>. Acesso em: 2 ago. 2026.
- JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. 2. ed. Upper Saddle River (Nova Jersey – EUA): Prentice Hall, 2008.
- LI, S.; XU, M.; ZHANG, X. L. Conformer-based End-to-end Speech Recognition with Rotary Position Embedding. **arXiv**, Ithaca, 2021. Disponível em: <https://arxiv.org/abs/2107.05907>. Acesso em: 4 ago. 2024.
- LIU, H.; KONG, Q.; LIU, J. CWS-PResUNet: Music Source Separation with Channel-wise Subband Phase-aware ResUNet. **Proceedings of the MDX Workshop**, 2021,

[s. l.], 9 dez. 2021. DOI 10.48550/arXiv.2112.04685. Disponível em: <https://arxiv.org/abs/2112.04685>. Acesso em: 13 ago. 2024.

LU, W. T. et al. **Music source separation with band-split RoPE transformer**. [S. l.]: arXiv, 5 set. 2023. DOI: 10.48550/arXiv.2309.02612. Disponível em: <https://arxiv.org/abs/2309.02612>. Acesso em: 14 out. 2024.

OPPENHEIM, A. V.; SCHAFER, R. W. **Processamento em tempo discreto de sinais**. 3. ed. São Paulo: Pearson Education do Brasil, 2012.
PEIXOTO, S. (ed.). Correspondente Médico: Qual a importância do entretenimento para a saúde mental?. **CNN Brasil**, [s. l.], 28 jul. 2020. Disponível em: <https://www.cnnbrasil.com.br/saude/correspondente-medico-qual-a-importancia-do-entretenimento-para-a-saude-mental/>. Acesso em: 13 out. 2024.

PATEL, Preet et al. Karaoke generation from songs: recent trends and opportunities. In: APSIPA ANNUAL SUMMIT AND CONFERENCE (APSIPA ASC), 2022, Chiang Mai, Tailândia. **Anais** [...]. Chiang Mai, Tailândia: APSIPA ASC, 2022. DOI: 10.23919/APSIPAASC55919.2022.9980133. Disponível em: <https://ieeexplore.ieee.org/document/9980133>. Acesso em: 13 ago. 2024.

PRANDONI, P.; VETTERLI, M. **Signal Processing for Communications**. Lausanne: EPFL Press, 2008. Distribuído por CRC Press.

RADFORD, A. *et al.* Robust Speech Recognition via Large-Scale Weak Supervision. **arXiv**, Ithaca, 2022. Disponível em: <https://arxiv.org/abs/2212.04356>. Acesso em: 14 ago. 2024.

RUIZ, R. E. **Servicio web para la conversión de canciones a formato karaoke usando inteligencia artificial**. 24 jul. 2024. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Telecomunicações) – Escola Politécnica Superior de Linares, Universidade de Jaén, Jaén (Espanha), 2024. Disponível em: <https://hdl.handle.net/10953.1/24128>. Acesso em: 14 ago. 2024.

SATO, Francisco Noriyuki. 2013. **Karaokê**. Disponível em: <https://www.culturajaponesa.com.br/index.php/diversos/musica/karaoke/>. Acesso em: 04 mar. 2026.

SIMPSON, A. J. R.; ROMA, G.; PLUMBLEY, M. D. Deep Karaoke: Extracting Vocals from Musical Mixtures Using a Convolutional Deep Neural Network. **Centre for Vision, Speech and Signal Processing, University of Surrey**, Guildford, Reino Unido, 17 abr. 2015. DOI 10.48550/arXiv.1504.04658. Disponível em: <https://arxiv.org/abs/1504.04658>. Acesso em: 13 ago. 2024.

SOLOVYEV, R.; STEMPKOVSKIY, A.; HABRUSEVA, T. **Music-Source-Separation-Training**: Repository for training models for music source separation. GitHub, 2023. Disponível em: <https://github.com/ZFTurbo/Music-Source-Separation-Training>. Acesso em: 3 ago. 2026.

SU, J. *et al.* RoFormer: Enhanced transformer with Rotary Position Embedding. **Neurocomputing**, Amsterdam (Países Baixos), v. 568, art. 127063, 2024. DOI:

<https://doi.org/10.1016/j.neucom.2023.127063>. Disponível em:
<https://www.sciencedirect.com/science/article/pii/S092523122301593X>. Acesso em:
 4 ago. 2024.

VASWANI, A. *et al.* Attention Is All You Need. In: CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (NIPS), 31., 2017, Long Beach, CA. **Anais** [...] Long Beach, Califórnia (EUA): Neural Information Processing Systems Foundation, 2017. p. 1–11. Disponível em:
<https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>. Acesso em: 4 ago. 2024.

VINCENT, E.; GRIBONVAL, R.; FÉVOTTE, C. Performance measurement in blind audio source separation. **IEEE Transactions on Audio, Speech, and Language Processing**, v. 14, n. 4, p. 1462-1469, 31 jul. 2006. DOI: 10.1109/TSA.2005.858005. Acesso em: 15 ago. 2024.

WANG, J. C.; LU, W. T.; WON, M. Mel-band RoFormer for music source separation. In: INTERNATIONAL SOCIETY FOR MUSIC INFORMATION RETRIEVAL CONFERENCE (ISMIR), 24., 2023, Milan, Itália. Extended abstracts for the Late-Breaking Demo Session. **Anais** [...]. Milan, Itália, 2023. DOI: 10.48550/arXiv.2310.01809. Disponível em: <https://arxiv.org/abs/2310.01809>. Acesso em: 14 out. 2024.

ZHANG, A. *et al.* **Dive into Deep Learning**: release 0.14.3. Cambridge (Inglaterra): Cambridge University Press, 2020. Disponível em: <https://d2l.ai>. Acesso em: 18 ago. 2024.

ZHU, Y.; GAO, S. Extracting vocal melody from karaoke music audio. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), 2005, Amsterdam, Países Baixos. **Anais** [...]. Amsterdam, Países Baixos: IEEE, 2005. DOI: 10.1109/ICME.2005.1521620. Disponível em:
<https://ieeexplore.ieee.org/document/1521620>. Acesso em: 14 out. 2024.

APÊNDICE A - MELROFORMER_CONFIG.PY

```

1  import sys
2  from pathlib import Path
3
4  repo_path = Path(__file__).parent / "Music-Source-Separation-Training"
5  sys.path.insert(0, str(repo_path))
6
7  from inference import proc_folder
8
9  CONFIG_PATH =
10     r"Music-Source-Separation-Training\configs\model\config_vocals_mel_band_roformer_kj.yaml"
11  CONFIG_PATH_CPU =
12     r"Music-Source-Separation-Training\configs\model\config_vocals_mel_band_roformer_kj_cpu.yaml"
13  CHECKPOINT = r"checkpoint\MelBandRoformer.ckpt"
14  GET_INSTRUMENTAL = True
15
16  def Generate_Karaoke_MusicFiles(music_folder = r"input", output_folder = r"output",
17     force_cpu = False):
18     #config = CONFIG_PATH_CPU if force_cpu else CONFIG_PATH
19
20     args = Config_MelRoFormer(CONFIG_PATH, CHECKPOINT, music_folder, output_folder,
21     force_cpu, GET_INSTRUMENTAL)
22     proc_folder(args)
23
24  def Config_MelRoFormer (config_path, start_checkpoint, music_folder, output_folder,
25     force_cpu = False, get_instrumental = True):
26     args = {
27         "model_type": "mel_band_roformer",
28         "config_path": config_path,
29         "start_check_point": start_checkpoint,
30         "input_folder": music_folder,
31         "store_dir": output_folder,
32         "extract_instrumental": get_instrumental,
33         "force_cpu": force_cpu}
34
35     return args
36
37  if __name__ == "__main__":
38     Generate_Karaoke_MusicFiles(music_folder="benchmark_input\Golden", output_folder=
39     "test_output")

```

APÊNDICE B – GENERATESUBTITLES.PY

```

1  from pathlib import Path
2  import subprocess
3  import sys
4
5  OUTPUT_FOLDER = Path("output")
6  MODEL = "large-v3"
7  INIT_PROMPT = "This file contains the isolated vocal track of a song."
8
9  def ConfigWhisper(input_folder, output_folder = OUTPUT_FOLDER, model= MODEL,
10 initial_prompt=INIT_PROMPT, output_format="srt", device="cpu"):
11     return [
12         sys.executable,
13         "-m",
14         "whisper",
15         input_folder,
16         "--model", model,
17         "--device", device,
18         "--output_dir", output_folder,
19         "--output_format", output_format,
20         "--word_timestamps", "True",
21         "--highlight_words", "True",
22         "--max_line_width", "45",
23         "--max_line_count", "2",
24         "--hallucination_silence_threshold", "2.0",
25         "--initial_prompt", initial_prompt
26     ]
27
28 def GenerateSubtitle(input_file, model= MODEL, initial_prompt=INIT_PROMPT):
29     vocal_file = Path(input_file)
30
31     command = ConfigWhisper(str(vocal_file), str(vocal_file.parent), model,
32 initial_prompt)
33
34     subprocess.run(command, check=True)
35
36 def GenerateSubtitles(output_folder = OUTPUT_FOLDER, model= MODEL, initial_prompt=
37 INIT_PROMPT):
38     for vocal_file in output_folder.rglob("vocals.wav"):
39         srt_file = vocal_file.with_suffix(".srt")
40
41         if srt_file.exists():
42             print(f"Pulando {vocal_file} (SRT já existe)")
43             continue
44
45         print(f"Processando {vocal_file}")
46
47         command = ConfigWhisper(str(vocal_file), str(vocal_file.parent), model,
48 initial_prompt)
49
50         subprocess.run(command, check=True)
51
52 if __name__ == "__main__":
53     GenerateSubtitles()

```