

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Trabalho de Graduação 2 - ES952

Relatório Final

Transcrição musical de violão em tempo real

AUTOR: VÍNICIUS DE OLIVEIRA CIVALI.

ORIENTADOR: PROF. DR. BRUNO SANCHES MASIEIRO.

CAMPINAS, SP

Dezembro de 2023

Sumário

LISTA DE FIGURAS.....	4
NOMENCLATURA.....	6
ABREVIACÕES E SIGLAS.....	9
RESUMO	11
1 INTRODUÇÃO.....	12
2 OBJETIVO.....	13
3 REVISÃO BIBLIOGRÁFICA.....	14
3.1 REAL-TIME AUTOMATIC PIANO MUSIC TRANSCRIPTION SYSTEM.....	14
3.1.1 <i>Introdução</i>	14
3.1.2 <i>Modelo de autorregressão</i>	14
3.1.3 <i>Implementação para tempo real</i>	15
3.1.4 <i>Visualização e exportação Midi</i>	15
3.1.5 <i>Discussão</i>	16
3.2 PRACTICAL INTRODUCTION TO AUTOMATION MUSIC TRANSCRIPTION.....	17
3.2.1 <i>Introdução</i>	17
3.2.2 <i>Tratamento de dados</i>	17
3.2.3 <i>Rede neural recorrente básica</i>	18
3.2.4 <i>LSTM Networks</i>	19
3.2.5 <i>BiLSTM Networks</i>	20
3.2.6 <i>Método de seleção de pico e avaliação do modelo</i>	20
3.2.7 <i>Discussão</i>	21
3.3 UNSUPERVISED MUSIC SOURCE SEPARATION USING DIFFERENTIABLE PARAMETRIC SOURCE MODELS	22
3.3.1 <i>Introdução</i>	22
3.3.2 <i>Método</i>	23
3.3.3 <i>Experimentos</i>	31
3.3.4 <i>Resultados</i>	36
3.3.5 <i>Discussão</i>	43
4 BASE DE DADOS.....	44
5 METODOLOGIA	46

5.1	ESCOLHA DA BASE DE DADOS	46
5.2	RESTRIÇÕES IMPOSTAS A PRIORI	46
5.3	SEPARADOR DE FONTES	47
5.4	UTILIZAÇÃO DE BiLSTM	48
5.5	TRANSFORMAR MIDI EM PARTITURA	48
6	DESENVOLVIMENTO	49
6.1	CRIAÇÃO DA BASE DE DADOS.....	49
6.2	PRÉ-PROCESSAMENTO DOS DADOS.....	49
6.3	EXTRAÇÃO DE CARACTERÍSTICAS	51
6.3.1	<i>Criação da matriz X de treinamento</i>	<i>52</i>
6.3.2	<i>Criação da matriz Y de treinamento</i>	<i>54</i>
6.3.3	<i>Discutindo X e Y.....</i>	<i>54</i>
6.4	ESCOLHA E DESENVOLVIMENTO DO MODELO DE APRENDIZADO	55
6.5	TREINAMENTO	56
7	REFLEXÃO SOBRE O DESENVOLVIMENTO	57
7.1	PERSPECTIVAS FUTURAS.....	57
8	FERRAMENTAS	58
9	CONCLUSÃO	59
10	CRONOGRAMA	60
11	REFERÊNCIAS BIBLIOGRÁFICAS.....	61
12	ANEXO	63
12.1	MATRIZ X.....	63
12.2	MATRIZ Y	64

Lista de figuras

1. [Entrada do modelo, espectrograma \(\[3\], Imagem do autor\).](#)
2. [Avaliação do modelo \(dados da fonte: R.S., J.H. e C.S., "Automatic Drum Transcription using Bi-directional Recurrent Neural Networks." dblp, 2016. Imagem do autor\).](#)
3. [A visão geral do procedimento de treinamento não supervisionado proposto para uma Rede Neural Profunda \(DNN\) para a separação de fontes de áudio supera limitações encontradas em métodos supervisionados, que dependem de grandes conjuntos de dados rotulados. Esse procedimento incorpora modelos paramétricos diferenciáveis de fontes de áudio no DNN, permitindo que o sistema seja orientado por dados e adaptado para representar características específicas de cada fonte sonora. Além disso, utiliza informações de frequência fundamental \(F0\) para auxiliar na separação das fontes. Os resultados experimentais mostram que a abordagem proposta supera métodos não supervisionados e supervisionados, mesmo com um conjunto de dados de treinamento relativamente pequeno. Isso torna o método aplicável em cenários com limitações na obtenção de dados de treinamento.](#)
4. [É apresentado uma visão geral da decomposição do modelo fonte-filtro. Os parâmetros do modelo são indicados em fonte azul. O símbolo 'o' representa multiplicação elemento a elemento. Embora a maioria dos componentes seja visualizada por meio de espectrogramas de magnitude, o processamento não é necessariamente realizado no domínio tempo-frequência.](#)
5. [Visão geral das etapas de processamento para a estimativa de parâmetros. Transformações com parâmetros aprendíveis são mostradas em verde, etapas de processamento predefinidas em cinza e as saídas \(intermediárias\) em caixas brancas. A forma de saída de uma transformação é mostrada na parte direita da caixa.](#)
6. [Calculo dos coeficientes do filtro \$a_k\$ a partir de \$\omega_k\$.](#)
7. [Os gráficos de violino e box plot apresentam os valores SI-SDR em dB para todos os quadros de avaliação. As caixas representam os intervalos entre o primeiro e o terceiro quartil, sendo que as medianas são destacadas por](#)

uma linha horizontal preta. As barras do box plot (azul escuro) se estendem desde o primeiro até o percentil 99. Os gráficos de violino abrangem todo o intervalo de dados. No gráfico (b), o NMF2 apresenta cinco valores atípicos entre -60 e -80 dB, que não estão representados no gráfico.

8. É apresentado os valores de p obtidos a partir de testes t de comparação em pares entre as distribuições dos valores SI-SDR para todos os experimentos realizados.
9. Efeitos de desativação de voz.
10. Mapeamento das notas do braço de um violão clássico de 19 trastes em afinação padrão, confeccionada pelo autor do trabalho.

Nomenclatura

- Keras: É um pacote de algoritmos de deep learning, originalmente desenvolvido na linguagem Python;
- LibROSA: É uma biblioteca para análise de áudios e sinais musicais;
- Função de ativação sigmóide: Uma função matemática utilizada em redes neurais artificiais para transformar a entrada em uma saída que pode ser utilizada como sinal de ativação para a próxima camada da rede;
- PyAudio: PyAudio é uma biblioteca de áudio para Python que permite aos desenvolvedores acessar facilmente os recursos de áudio do sistema operacional em que estão trabalhando. Com o PyAudio, é possível gravar áudio a partir de microfones e outras fontes de entrada, reproduzir áudio em alto-falantes e outras fontes de saída, além de processar áudio em tempo real;
- PyTorch: Uma estrutura de aprendizado de máquina baseada na biblioteca Torch, usada para aplicativos como visão computacional e processamento de linguagem natural, originalmente desenvolvida pela Meta AI e agora parte do guarda-chuva da Linux Foundation;
- Logic Pro X: Uma coleção completa de ferramentas de criação sofisticadas para composição, criação de batidas, edição e mixagem profissionais, integradas a uma interface moderna, concebida para obter resultados rapidamente e oferecer mais poder sempre que necessário;
- RtMidi: uma biblioteca de código aberto para C++, C, Python e Java que permite que os desenvolvedores criem aplicativos que interajam com dispositivos MIDI (Musical Instrument Digital Interface);
- Piano Roll 2D: Uma representação gráfica do sequenciador MIDI que permite aos músicos visualizarem e editarem sequências MIDI em forma de grade. A representação gráfica é geralmente exibida em duas dimensões, com o eixo horizontal representando o tempo e o eixo vertical representando a nota musical;
- Omnizart: O texto apresenta o Omnizart, uma biblioteca Python que oferece uma solução simplificada para transcrição automática de música. A biblioteca reúne resultados de pesquisa do Laboratório de Tecnologia Musical e Cultural e é capaz de analisar músicas polifônicas e transcrever

notas musicais de instrumentos, progressões de acordes, eventos de bateria, melodia vocal em nível de quadro e em nível de nota, além de ritmo.

- Latent source representation: A representação de fonte latente refere-se a uma representação oculta ou subjacente de informações ou características em um conjunto de dados. É uma forma de capturar aspectos não diretamente observáveis dos dados, permitindo extrair informações mais abstratas e significativas.
- Janela de Hann: A janela de Hann é uma função matemática utilizada no processamento de sinais, especialmente em análise espectral, para reduzir artefatos causados por bordas bruscas e melhorar a precisão das medidas. Ela é uma janela com formato de uma curva suave que é multiplicada pelo sinal de entrada para atenuar as oscilações nas extremidades do sinal.
- Filtragem de Wiener: A filtragem de Wiener é um método de processamento de sinais usado para melhorar a qualidade de um sinal ao remover ruídos indesejados. Ela utiliza um modelo estatístico para estimar a relação entre o sinal original e o ruído, permitindo a criação de um filtro adaptativo que minimiza a distorção e maximiza a relação sinal-ruído. Isso resulta em uma melhoria na clareza e na qualidade do sinal de interesse.
- Filtro all-pole: Um filtro all-pole é um tipo de filtro digital que é composto exclusivamente por polos e não possui zeros. É chamado de "all-pole" porque sua função de transferência é descrita apenas pelos polos, que são pontos no plano complexo que afetam a resposta em frequência do filtro. Os filtros all-pole são amplamente utilizados em várias aplicações de processamento de sinais, como remoção de ruído, equalização de áudio e modelagem de sistemas lineares. Eles são eficazes na moldagem da resposta em frequência do sinal, permitindo o controle do espectro de frequência e a atenuação de componentes indesejados.
- Mel-espectrograma: O mel-espectrograma é uma representação visual do espectro de frequência de um sinal de áudio ao longo do tempo. Ele combina características do espectrograma (que mostra as intensidades de diferentes frequências ao longo do tempo) com a escala mel (que reflete melhor a percepção auditiva humana). Simplificadamente, o mel-espectrograma é uma maneira de visualizar como as diferentes frequências

de som estão presentes em um sinal de áudio em relação ao tempo. Ele é útil em várias aplicações, incluindo processamento de fala, música e análise de áudio, pois fornece uma representação mais alinhada com a forma como os seres humanos percebem o som.

- **Frame:** Em processamento de sinais de áudio, um "frame" refere-se a uma pequena janela de tempo extraída do sinal. Esse conceito é fundamental em técnicas como a Short-Time Fourier Transform (STFT) e a criação de Mel-espectrogramas. Cada frame representa uma análise localizada do sinal em intervalos sucessivos, permitindo a observação das propriedades temporais e de frequência ao longo do tempo. Essa abordagem é comumente usada em diversas aplicações, como processamento de fala e reconhecimento de padrões musicais.
- **Evento:** Lista que contém todas as notas tocadas em um determinado tempo da música.

Abreviações e siglas

- RNN: Recurrent neural network;
- LSTM: Long Short-Term Memory;
- BLSTM: Bidirectional Long Short-Term Memory;
- MIDI: Musical Instrument Digital Interface;
- STFT: Transformada rápida de Fourier;
- AMT: Automatic Music Transcription;
- AR: Autorregressivo;
- CNN: Redes neurais convolucionais;
- MSS: Musical Source Separation;
- DNN: Deep Neural Networks;
- GRUs: Unidades de Recorrência de Portas (Gated Recurrent Units) são uma variante de redes neurais recorrentes (RNNs) que possuem mecanismos de gating para melhorar o fluxo de informações em sequências temporais.
- MLP: Perceptron Multicamadas (Multilayer Perceptron) é uma arquitetura clássica de rede neural artificial composta por várias camadas de neurônios, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída.
- FIR: Resposta ao Impulso Finita (Finite Impulse Response) refere-se a um tipo de filtro digital cuja resposta ao impulso tem uma duração finita no tempo. Os coeficientes desse tipo de filtro são estáticos e não dependem dos valores de saída anteriores.
- IIR: Resposta ao Impulso Infinita (Infinite Impulse Response) refere-se a um tipo de filtro digital cuja resposta ao impulso tem uma duração infinita no tempo. Os coeficientes desse tipo de filtro podem depender dos valores de saída anteriores, o que resulta em uma resposta com realimentação.
- LSFs: Coeficientes Lineares de Predição (Linear Prediction Coefficients) são parâmetros usados em técnicas de análise de sinais para modelar a fonte vocal de um sinal de fala.
- FFT: Transformada Rápida de Fourier (Fast Fourier Transform) é um algoritmo eficiente para calcular a Transformada de Fourier Discreta (DFT)

de um sinal, permitindo a análise e decomposição do sinal em componentes de frequência.

- NMF: Decomposição em Fatores Não-negativos (Non-negative Matrix Factorization) é uma técnica de análise de dados que visa decompor uma matriz não-negativa em dois fatores que possuem propriedades úteis para a representação e análise dos dados.
- F0: Frequência Fundamental refere-se à frequência mais baixa presente em um sinal de áudio, que corresponde à nota musical fundamental percebida.
- MIDI: Interface Digital de Instrumentos Musicais (Musical Instrument Digital Interface) é um protocolo de comunicação padrão usado para conectar e controlar instrumentos musicais eletrônicos, permitindo a transmissão de informações sobre notas, dinâmica, duração e outros aspectos musicais. Também é usado como formato de arquivo para representar dados musicais digitais.
- WAV: Waveform Audio File Format (Formato de Arquivo de Áudio em Forma de Onda). Trata-se de um formato de arquivo de áudio digital amplamente utilizado para armazenar gravações de som em computadores e dispositivos de áudio. O formato WAV é conhecido por sua alta qualidade de áudio, pois não realiza compressão de dados, o que significa que preserva a qualidade original do som.

Resumo

Este projeto visa a construção de um sistema de transcrição musical em tempo real, ou seja, criar a partitura/tablatatura de acordo com que se toca o instrumento, mais especificamente o violão de seis cordas com a afinação Standard (E, A, D, G, B, E). Este escopo é mais restrito pelo fato de a área ser desafiadora levando em consideração a escassez de tecnologias como estás voltado o violão.

O processo de criação musical enfrenta diversas barreiras, são necessários diferentes conhecimentos para a música sair da imaginação do compositor até chegar em sua forma escrita, neste sentido, esta empreitada visa facilitar o processo criativo, criando uma ferramenta para auxiliar tanto músicos experientes quanto iniciantes na criação de suas obras.

Com o advento da inteligência artificial, surgiram ferramentas que executam esta tarefa, porém não é uma área muito explorada, abrindo margem para novas empreitadas. Visto isso, é conhecido que já existem transcritores musicais que trabalham em pós processamento, tendo como input uma gravação, como é bem salientado neste vídeo, [\[1\]](#), em que o autor explora a biblioteca, Omnizart, já existente e outros recursos do Python que realizam a transcrição musical. Há também transcritores em tempo real funcionais que utilizam RNNs (Recurrent neural network), como já apontado tanto em [\[2\]](#), quanto em [\[3\]](#), em que no cenário musical os modelos que melhor se adequam são LSTM (Long Short-Term Memory) e BLSTM (Bidirectional Long Short-Term Memory) respectivamente. Desta forma, o atual trabalho visa realizar um levantamento de desenvolvimentos já realizados anteriormente na área a fim de mapear as tecnologias existentes e entender suas implementações.

1 Introdução

A transcrição de música em tempo real é o processo de conversão de um sinal de áudio em uma representação simbólica da música, como partituras, tablaturas e/ou arquivos MIDI (Musical Instrument Digital Interface), em tempo real. Com a crescente demanda por análise e processamento de música, a transcrição em tempo real ganhou interesse significativo de pesquisadores acadêmicos e da indústria. Nesta revisão bibliográfica, pretendemos explorar os recentes avanços em algoritmos e técnicas de transcrição de música em tempo real. Analisaremos as abordagens de alguns trabalhos chave e seus pontos fortes e limitações.

A transcrição musical em tempo real é um desafio importante no campo da música e processamento de sinais. O trabalho de Jeong [\[2\]](#) apresenta um sistema de transcrição automática de músicas de piano em tempo real, utilizando técnicas de aprendizado de máquina para extrair informações das notas tocadas. Já o artigo de Hu [\[3\]](#) apresenta uma introdução prática sobre a transcrição automática de música, abordando conceitos como processamento de sinais e técnicas de aprendizado de máquina aplicadas à transcrição de música em formato de áudio.

Essas referências demonstram que a transcrição musical em tempo real é uma área em desenvolvimento, com muitos desafios a serem enfrentados. O uso de técnicas de processamento de sinais e aprendizado de máquina é uma abordagem promissora para a transcrição musical em tempo real, mas ainda existem muitos problemas a serem resolvidos, como a identificação de diferentes instrumentos e a precisão da transcrição. Através dessa revisão bibliográfica, é possível perceber que a transcrição musical em tempo real é um campo de estudo em constante evolução e com muitas possibilidades de aplicação.

2 Objetivo

Este trabalho tem como objetivo criar um modelo de transcrição musical para violão acústico baseado em técnicas avançadas de aprendizado de máquina, como redes neurais e processamento de sinais, a fim de realizar uma transcrição precisa das notas musicais tocadas no instrumento. O modelo será treinado em um conjunto de dados abrangente e diversificado, contendo gravações de violão acústico em diferentes estilos musicais e técnicas de execução. O objetivo é fornecer uma ferramenta eficiente e confiável para músicos, compositores e pesquisadores musicais que desejam transcrever e analisar peças musicais executadas no violão acústico, facilitando a criação de partituras e o estudo de performances musicais.

Desta forma, como este relatório trata-se da primeira parte do projeto, será trabalhado mais revisões bibliográficas na área de transcrição musical em tempo real, a fim de levantar quais as bases de dados utilizadas, as respectivas redes neurais utilizadas, assim como suas estruturas e demais informações relevantes para o início da confecção de um novo produto na área.

Além do mais, o mapeamento da proposta de implementação será exemplificado, valendo salientar que o escopo não está totalmente fechado, ou seja, pode e haverá alterações ao longo do desenvolvimento.

3 Revisão bibliográfica

3.1 Real-time Automatic Piano Music Transcription System

3.1.1 Introdução

Este trabalho [2] tem como objetivo a construção de um sistema AMT (Automatic Music Transcription) que funcione em um ambiente online em tempo real, ao contrário dos demais trabalhos que ocorrem no cenário off-line. Desta forma, neste artigo é demonstrado a implementação de um sistema AMT em tempo real voltado para o piano, trabalho o qual foi apresentado na SKT Tech Gallery em Pangyo, Coréia.

3.1.2 Modelo de autorregressão

Este trabalho se apoia na pesquisa [4], a qual propusera um modelo AMT baseado em aprendizado profundo com um RNN autorregressivo (AR). Desta forma, a conexão AR alimenta o resultado transcrito do período de tempo anterior na ativação do período de tempo seguinte. Diferentemente da pesquisa [5] que faz a previsão de tudo apenas na etapa final, o modelo com AR sintetiza a saída da transcrição frame a frame, este ponto torna-se bastante positivo, além de realizar as estimativas para os estados das notas, como off, onset, sustain, offset e re-onset por cada pitch para cada frame.

Faz-se de suma importância a discussão da base de dados, assim, este trabalho utilizou para os treinamentos a base de dados MAESTRO V2.0.0 [6], composta por 200 horas de áudio e arquivos MIDI correspondentes de apresentações de piano.

Durante o desenvolvimento foram realizadas 162 combinações de hiperparâmetros para o treinamento, como diferentes taxas de aprendizado, decaimento de peso, número de canais CNN (Redes neurais convolucionais) e número de nós ocultos no LSTM e selecionou o modelo com maior pontuação F1 de notas com deslocamentos.

Não apenas, mas também, a utilização de um modelo AR consegue ser adaptado facilmente sem perda de desempenho para um cenário online, ponto o qual não é o foco deste trabalho necessariamente, mas em posse de um trabalho de sucesso que enfrentou este desafio a perspectiva de uma possível mudança para este campo pode vir a utilizar de lastro tal trabalho.

3.1.3 Implementação para tempo real

O desafio deste trabalho se deu devido o fato de ser uma transcrição em tempo real e online, assim a solução deveria abarcar computação em tempo real com baixa latência e sem diminuir a qualidade e precisão do modelo. Além disso, o caminho seguido foi a adaptação do modelo [\[4\]](#) para um sistema online, o que levou a algumas escolhas no projeto para assegurar as qualidades necessárias supracitadas.

Algumas escolhas empíricas foram tomadas e outras mantidas dos trabalhos referência, como o tamanho do salto do espectrograma de mel, idêntico ao comprimento de cada intervalo de tempo no módulo de linguagem (consiste em duas camadas de LSTM unidirecional auto-regressiva) e ser equivalente à resolução de tempo da transcrição. Foi notado que o tamanho do salto reduzia o custo de computação, mas a necessidade da preservação da resolução do tempo manteve os saltos escolhidos em [4 5] de 512 amostras em taxa de amostragem de 16 [KHz], que é 32 [ms] com uma janela de salto de 2048;

O trabalho referência [\[4\]](#) usaram três camadas de redes neurais convolucionais (CNN) com tamanho de kernel de três quadros no eixo do tempo, desta forma, o campo de visão da camada final é de sete quadros no eixo de tempo dos melspectrogramas, ou seja, 5120 amostras, 320 [ms] de áudio em taxa de amostragem de 16 [kHz]. Em busca da otimização requerida pela empreitada online há um buffer de áudio de 5120 amostras atualizado a cada 512, buffer este convertido em melspectrograma com 221 bins e 7 quadros. Quando as 512 novas amostras atualizam apenas o último quadro do melspectrograma é atualizado enquanto os demais são deslocados de um quadro. É válido frisar que a atualização da camada CNN é feita apenas para o último quadro do melspectrograma para resguardar o esforço computacional.

3.1.4 Visualização e exportação Midi

A ferramentas utilizadas para nosso interesse foram o PyAudio para a entrada do microfone, PyTorch para as redes neurais e Librosa para converter amostras de áudio em espectrograma de mel, além do mais, para amostragem do resultado na forma transcrita simbólica foi utilizado o Logic Pro X, o qual processa o sinal MIDI de entrada como uma partitura musical, embora as informações métricas sejam convertidas com base na batida do metrônomo. Em meios práticos o Python envia sinais MIDI para portas internas MIDI

do RtMidi e o Logic Pro X, que processam os sinais como partitura, o que por si só se mostra uma solução elegante para a construção da forma simbólica da música, o que expande os horizontes.

Como visualização de resultados, foram utilizados programas para construção de um piano roll 2D, Python e interface JavaScript, mas como é algo específico do projeto o ponto de interesse fica à ideia para possíveis representações visuais para este trabalho, o que não havia sido considerado;

3.1.5 Discussão

Os pontos positivos associados a este trabalho foram saber sobre a base de dados, a qual infelizmente não apresenta dados para violão, os modelos utilizados e principalmente a maneira da construção simbólica da música, através de softwares já existentes, o que propiciou novas possíveis ideias, além da ideia de trabalharmos com algoritmos com autorregressão para um melhor resultado.

3.2 *Practical Introduction to Automation Music Transcription*

3.2.1 Introdução

Este trabalho, [3], visa realizar uma introdução prática ao campo de transcrição musical, passa pelas diferentes etapas do processo, desde o tratamento da entrada, explicando sucintamente transformada de Fourier, espectrograma, conceitos básicos de redes neurais recorrentes e aborda, nossa parcela de interesse, um modelo de transcrição musical baseado nas redes LSTM e BiLSTM.

Visto que se trata de uma introdução prática, ensina como criar modelos com o pacote Keras, disponibilizando os códigos em Python, finalizando discutindo métodos de avaliação de desempenho.

3.2.2 Tratamento de dados

Utilizando a biblioteca Librosa aplica-se uma STFT (Transformada rápida de Fourier) à entrada, o que gera um espectrograma.

Por sua vez, após esta primeira etapa, tem-se que concatenar o espectrograma para vetores, em que cada linha no vetor representa o nível de frequência e, cada coluna, um time frame. Os valores no vetor representam as amplitudes. Um cuidado interessante realizado é a adição de zeros ao fim da música, baseado no tamanho do áudio, a fim de evitar a concatenação de duas músicas.

Outra premissa cuidadosa é que se assume que toda música supostamente teria o mesmo número de níveis de frequência, mas não a mesma duração, logo a matriz é transposta a fim de que cada música tenha o mesmo número de colunas.

Com esta formatação os vetores podem ser facilmente concatenados e usados para treinamento.

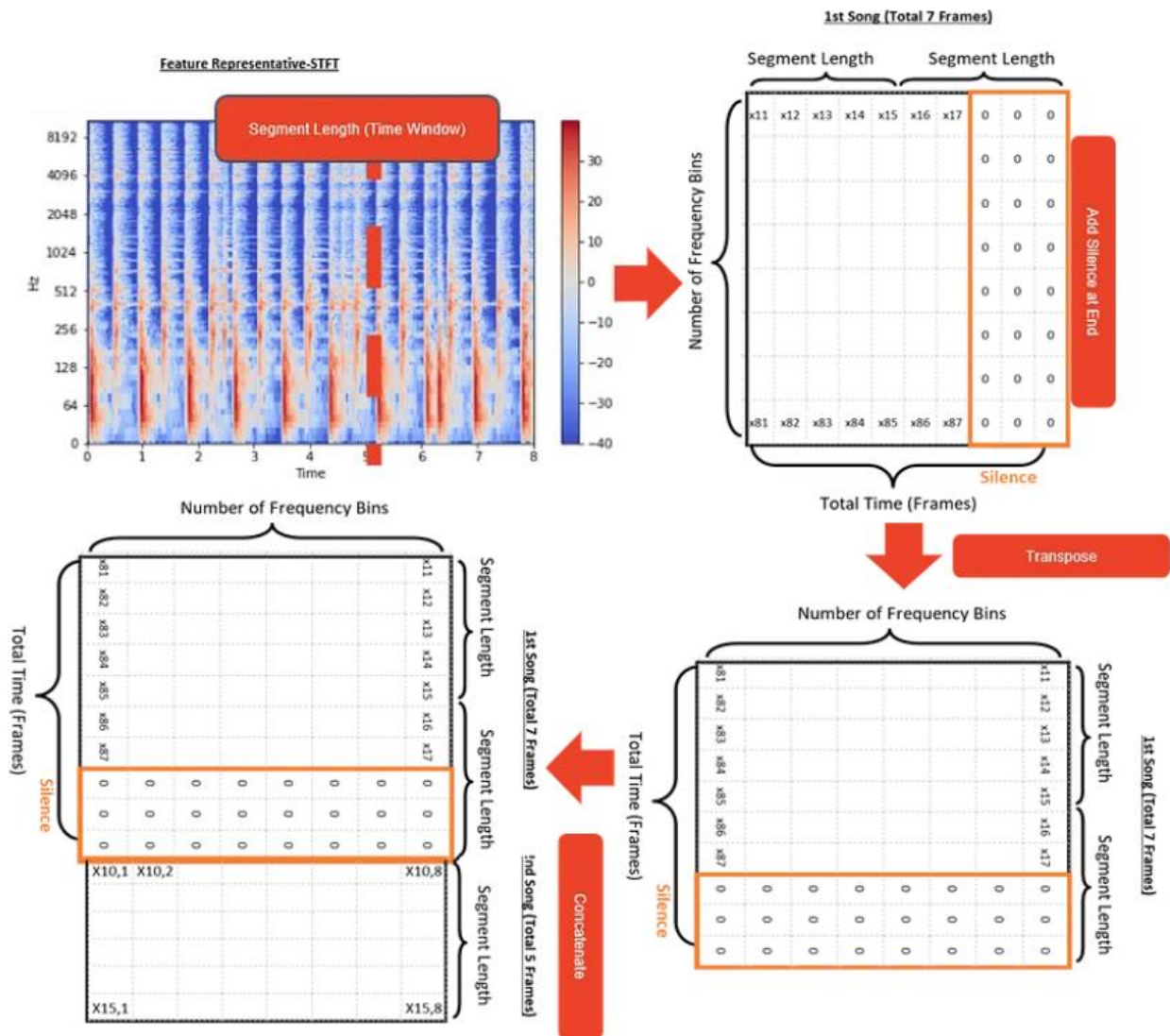


Fig 1: Entrada do modelo, espectrograma. ([3], Imagem feita pelo autor)

3.2.3 Rede neural recorrente básica

É explicado como o modelo funciona utilizando uma rede neural simples, desta forma são executadas 4 etapas neste exemplo:

1. Concatenação dos espectrogramas, explicado anteriormente e representado na [Figura 1](#);
2. A entrada é dividida em pequenos segmentos. Janelas, com base no tamanho de seguimento definido, no caso do trabalho em questão foi utilizado nesta primeira

implementação cinco segundos. Assim, os quadros dos espectrogramas são divididos em segmentos de 5 segundos e utilizados sequencialmente como a entrada que alimenta a camada oculta da rede.

É salientado que este exemplo utiliza unfolded RNN, a qual fornece todas as saídas, não apenas a final. Entretanto, este tipo básico de rede apresenta um problema conhecido como fuga de gradiente.

Primeiramente o Gradiente Sigmóide, também conhecido como função de ativação sigmóide, é uma função matemática utilizada em redes neurais artificiais para transformar a entrada em uma saída que pode ser utilizada como sinal de ativação para a próxima camada da rede.

O fenômeno de fuga de gradiente sigmoide em redes neurais ocorre quando a função de ativação sigmoide é utilizada em redes neurais com muitas camadas o que pode causar problemas de treinamento e convergência do modelo.

A fuga de gradiente ocorre quando o gradiente da função de erro se torna muito pequeno à medida que é propagado para trás através das várias camadas da rede neural. Quando isso acontece, os pesos das conexões entre as camadas não são atualizados corretamente, resultando em um modelo que não consegue aprender com os dados de treinamento.

O problema nesta abordagem simplista é que vamos apenas adicionando mais camadas a fim de melhorar os resultados, isto faz com que a acurácia da função sigmóide caia, logo este modelo mostra-se limitado.

3.2.4 LSTM Networks

Em virtude do problema da utilização de redes neurais simples, a fuga do gradiente, é sugerido a utilização do modelo LSTM (Long Short Term Memory).

Este modelo contém um estado de memória interna a qual é filtrada pelo Forget Gate e adicionada, ao invés de multiplicar, a entrada processada, o que de ante mão é bem útil quanto a pequenos gradientes. Por sua vez o Forget Gate determina qual estado anterior deve ser utilizado (Forget Gate com saída 1) e qual deve ser esquecido (Forget Gate com saída 0). Este comportamento permite a LSTM aprender comportamento baseado em um contexto útil.

Com este comportamento de memória o LSTM se mostra bem flexível, tendo controle a entrada, o que é lembrado pelo estado de memória interna e o que é enviado para célula seguinte, devido essa flexibilidade é um modelo extensamente utilizado em previsões dependentes do tempo, o que enquadra a transcrição musical.

3.2.5 BiLSTM Networks

O modelo LSTM tem uma variação bem famosa, a BiLSTM (Bidirectional Long Short Term Memory). Diferente de seu modelo primal, as BiLSTM possuem duas camadas de memória, uma para cada direção da sequência de entrada. Isso significa que as informações são processadas em ambas as direções da sequência, o que permite que a rede neural capture informações contextuais de ambos os lados da sequência.

Segundo a autora, este modelo impulsionou os resultados devido o comportamento de memória bidirecional.

3.2.6 Método de seleção de pico e avaliação do modelo

O método de seleção de pico é utilizado aqui para identificar o início de cada instrumento, desta forma, é aplicado à saída da função sigmóide (1 se saída > limite, caso contrário 0). Entretanto, o problema deste método é que os resultados variam muito de acordo com o kit de bateria ou instrumento, além do pico de início ter uma relação com os diferentes gêneros de música. Utilizando este método convencional ter-se-á valores de 1 ao redor do pico de alta probabilidade apenas, ocorrendo uma perda do pico real de baixa probabilidade levando a falsos positivos e falsos negativos altos. Por mais que haja este problema o método de seleção de pico foi usado, o que neste trabalho corrobora visto que o instrumento principal é a bateria, ou seja, instrumento formado por várias partes menores que exigem ser distinguidas a fim do algoritmo identificar os respectivos comportamentos, sons e etc.

3.2.7 Discussão

Por fim este trabalho apresentou ótimos resultados voltados para instrumentos percussivos, ultrapassando apenas a bateria, se baseando na arquitetura de BiLSTM, o que pode ser visto nesta imagem criada pelo autor:

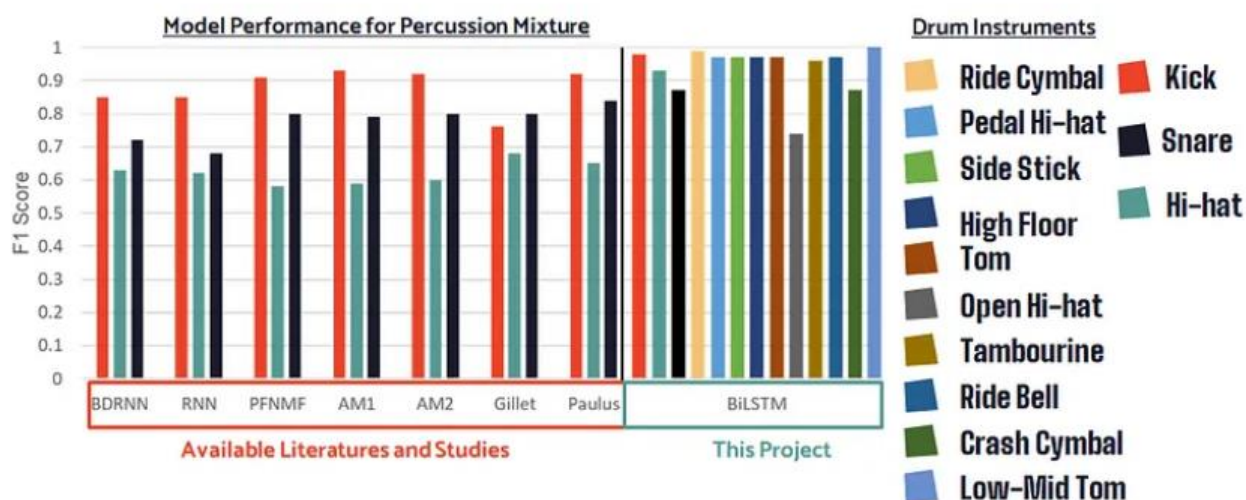


Fig 2: Avaliação do modelo (dados da fonte: R.S., J.H. e C.S., "Automatic Drum Transcription using Bi-directional Recurrent Neural Networks." dblp, 2016. ([2], Imagem do autor)

Visto que este trabalho é uma introdução mais simplista e prática há vários pontos em aberto, qual a base de dados utilizada, qual o algoritmo de erro entre outros pontos, porém o valor maior é mostrado no sucesso do uso da BiLSTM para os meios de transcrição musical e a disponibilidade de todos os códigos desenvolvidos, o que pode ser de grande ajuda.

3.3 Unsupervised Music Source Separation Using Differentiable Parametric Source Models

3.3.1 Introdução

Este trabalho tem como desafio a separação de fontes sonoras em áudios complexos, áudios com diferentes fontes sonoras, utilizando métodos de aprendizagem não supervisionada. No âmbito musical recebe o nome de separação de fonte musical (MSS).

O ponto positivo do uso de métodos não supervisionados é que estes conseguem realizar a separação das fontes apenas com treinamento adequado e sem a necessidade da gravação das fontes isoladamente, podendo alcançar resultados similares e superiores aos métodos supervisionados, também sendo mais flexíveis em relação aos dados de treinamento.

Cada fonte é modelada com um modelo de fonte paramétrica diferenciável. Durante o treinamento, a tarefa da DNN (Rede neural profunda) é resintetizar a mistura observada como uma soma das fontes, estimando os parâmetros da fonte. A separação é alcançada porque os F0s (frequências fundamentais) para todas as fontes são estimados a partir do áudio original, onde as fontes estão misturadas, e atribuídos às fontes de antemão.

Além de ser não supervisionado e capaz de separar fontes homogêneas, a abordagem tem outras vantagens: alta eficiência de dados, bem como estimativas de fonte paramétricas, portanto interpretáveis e modificáveis. Resumidamente, as contribuições deste trabalho, são:

- Uma nova abordagem de aprendizado profundo não supervisionado para separação de fontes de áudio;
- A integração de modelos de fonte paramétrica na separação de fonte de áudio baseada em aprendizagem profunda;
- Um novo procedimento diferenciável para estimar filtros all-pole variantes no tempo estáveis de uma DNN usando linha espectral;
- Parametrização de frequência, uma extensa avaliação experimental do método proposto em uma tarefa de separação de fontes musicais e comparação com linhas de base supervisionadas e livres de aprendizado;

- O código-fonte aberto para o método e experimentos propostos.

3.3.2 Método

É observado o som mono original fazendo $m(t) = \sum_{j=1}^J s_j(t)$ de J sinais de fonte monofônica $s_j(t)$, em que $t \in \{1, \dots, T\}$ indexa amostras de tempo discreto. O objetivo do trabalho é estimar todas as fontes de sinal s_j , propondo uma nova metodologia para treinar uma DNN para a tarefa sem acessar nenhuma fonte isoladamente.

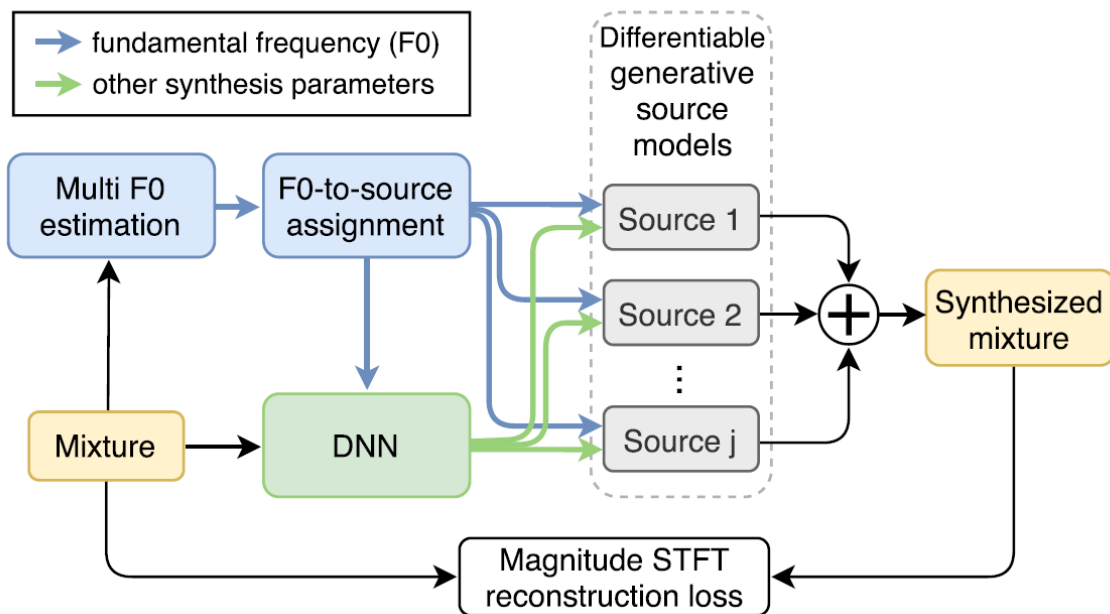


Fig 3: Visão geral do procedimento de treinamento não supervisionado proposto para uma Rede Neural Profunda (Deep Neural Network - DNN) destinada à separação de fontes de áudio. ([27], Imagem do autor).

3.3.2.1 Modelo de fonte

As fontes por sua vez são modeladas por modelos de fontes paramétricas diferenciáveis. O método de modelagem não é específico para nenhum modelo de fonte com tanto que seja passível de ser formulada de maneira diferenciável.

Assim, esta modelagem descreve o sinal como uma excitação de sinal de uma fonte sonora (por exemplo, a glote) a qual é modificada por um filtro variante no tempo (por exemplo, o trato vocal). Desta maneira pode-se usar essa modelagem para diferentes fontes, desde voz humana até instrumentos musicais.

Pode-se ver uma representação visual do modelo na Figura 4, onde foi assumido que a verdadeira fonte de sinal $s_j(t)$ é dividida em N quadros de tamanho T' amostras. O n -ésimo quadro é dado por:

$$s_j(n, t) = s_j(t + n \cdot B), \quad t \in \{1, \dots, T'\} \quad (1)$$

- B é o tamanho do salto entre os quadros nas amostras e $n \in \{1, \dots, N\}$.

Considerando a notação $\tilde{s}_j(n, t)$ como a estimativa do quadro do sinal da fonte gerado pelo modelo de fonte, o modelo de fonte pode ser formulado no domínio de Z como:

$$\tilde{s}_j(n, z) = E_j(n, z) \cdot \frac{1}{A_j(n, z)} \quad (2)$$

- $E_j(n, z)$ é a transformada z do sinal de excitação $e_j(n, z)$
- $\frac{1}{A_j(n, z)}$ é a função de transferência de um filtro multipolar variante no tempo de ordem K .

O processo de filtragem (1) pode ser melhor descrito como:

$$\tilde{s}_j(n, z) = e(n, t) - \sum_{k=1}^K a_k(n) X \tilde{s}(n, t - k) \quad (3)$$

- $a_k(n)$ são o coeficiente de filtros por cada quadro n ;
- " X " representa multiplicação escalar.

Detalhes de implementação serão discutidos na seção adequada.

É usado uma onda senoidal mais ruído para gerar o sinal de excitação $e(n, t)$, os autores discutem que é um ótimo sistema de sintetização para música e sinais de fala, que sintetiza o som como uma soma de senoides e ruído branco. Os autores comentando também de uma nova versão diferenciável, a qual apresentou ótimos resultados quando usado para modelos baseados em aprendizado profundo. Como é modelado uma fonte mono, as frequências da onda são restringidas para múltiplos inteiros da frequência fundamental. Por fim, o modelo se reduz ao modelo de harmônicos mais ruído, assim:

$$e(n, t) = [\alpha(n, t) \cdot h(n, t)] * r(t) + [w(t) * d(t)] \cdot g(n) \quad (4)$$

- " $*$ " representa convolução.
- $\alpha(n, t)$ Amplitude variável no tempo do sinal harmônico $h(n, t)$.
- $r(t)$ e $d(t)$ são o ganho de ruído constante para o quadro n .

O sinal harmônico $h(n, t)$ é definido como:

$$h(n, t) = \sum_{i=1}^I \sin(\Phi_i(n, t)) \quad (5)$$

$$\Phi_i(n, t) = 2 \cdot \pi \cdot \sum_{v=1}^t \frac{i \cdot f_0(n, v)}{f_s} \quad (6)$$

- Φ_i fase instantânea do i -ésimo harmônica (fase inicial é assumida como zero);
- f_0 frequência fundamental instantânea;
- f_s frequência de amostragem.

Observar que o sinal $h(n, t)$ é totalmente parametrizado pela frequência fundamental variável no tempo f_0 .

O filtro $r(t)$ impõem um formato espectral fixo em $h(n, t)$. Sem $r(t)$ todas as ondas têm a mesma amplitude.

O trabalho salienta que para alguns tipos de som pode-se assumir uma forma espectral específica invariante no tempo.

Também pode-se usar um parâmetro de amplitude específico para cada senoide em $h(n, t)$;

O trabalho por fim escolheu fazer o ganho dependente da frequência e não do número harmônico.

Similarmente, $d(t)$ determina o formato espectral da componente do ruído.

Ambos os filtros são invariantes no tempo, assim eles representam apenas a forma espectral global.

Também salientam que variações de curto prazo, por exemplo às articulações das palavras, são modelados pelo filtro multipolar $\frac{1}{A_j(n, z)}$.

Em resumo, tem-se que os parâmetros do modelo de fonte são $\{a_k(n), \alpha(t), f_0(t), r(t), g(n), d(t)\}$. Será explicitado na próxima seção como obtê-los. α e f_0 precisam variar lentamente o suficiente ao longo do tempo para o modelo seja matematicamente identificável. O autor comenta que a maneira que esses parâmetros são estimados levam a uma trajetória suave.

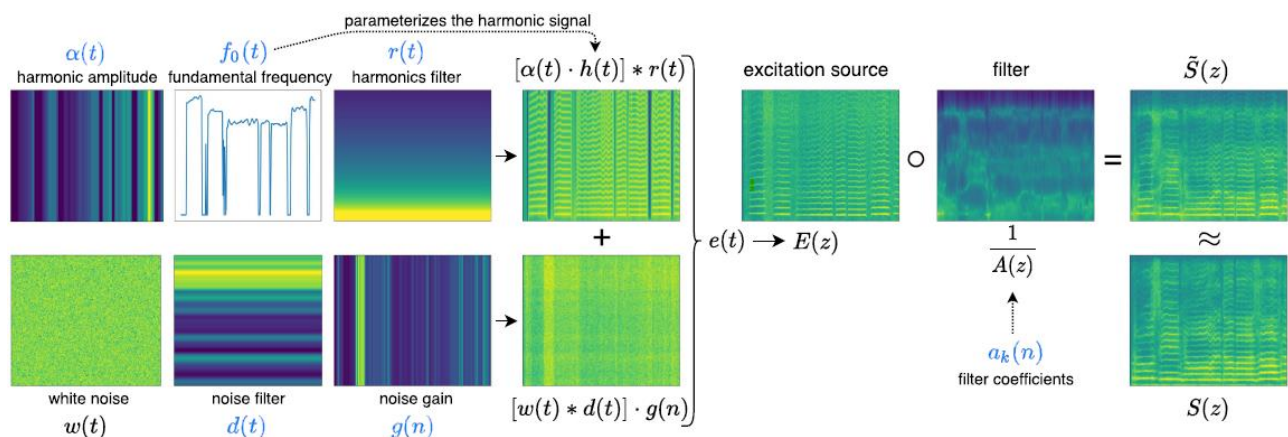


Fig 4: Visão geral da decomposição do modelo fonte-filtro. Os parâmetros do modelo são indicados em fonte azul. O símbolo ' $\odot$ ' denota multiplicação elemento a elemento. Embora a maioria dos componentes seja visualizada por meio de espectrogramas de magnitude, o processamento não é necessariamente realizado no domínio tempo-frequência ([27], Imagem feita pelo autor).

3.3.2.2 Estimação de parâmetros

Assumindo que a frequência fundamental para cada fonte J pode ser obtido do sinal original com um sistema de estimativa de F_0 múltiplo (o autor argumenta sobre a suposição). Após a obtenção de todos os F_0 , cada uma precisa ser designada para uma fonte. Normalmente as estimativas de F_0 são fornecidas a uma taxa de quadros menor que a frequência de amostragem, desta forma as séries temporais F_0 específicas da fonte são reamostradas para a taxa de amostragem usando interpolação bilinear.

A seguir discute-se como os parâmetros restantes são aquiritados com uso de DNN para cada fonte dado seu F0:

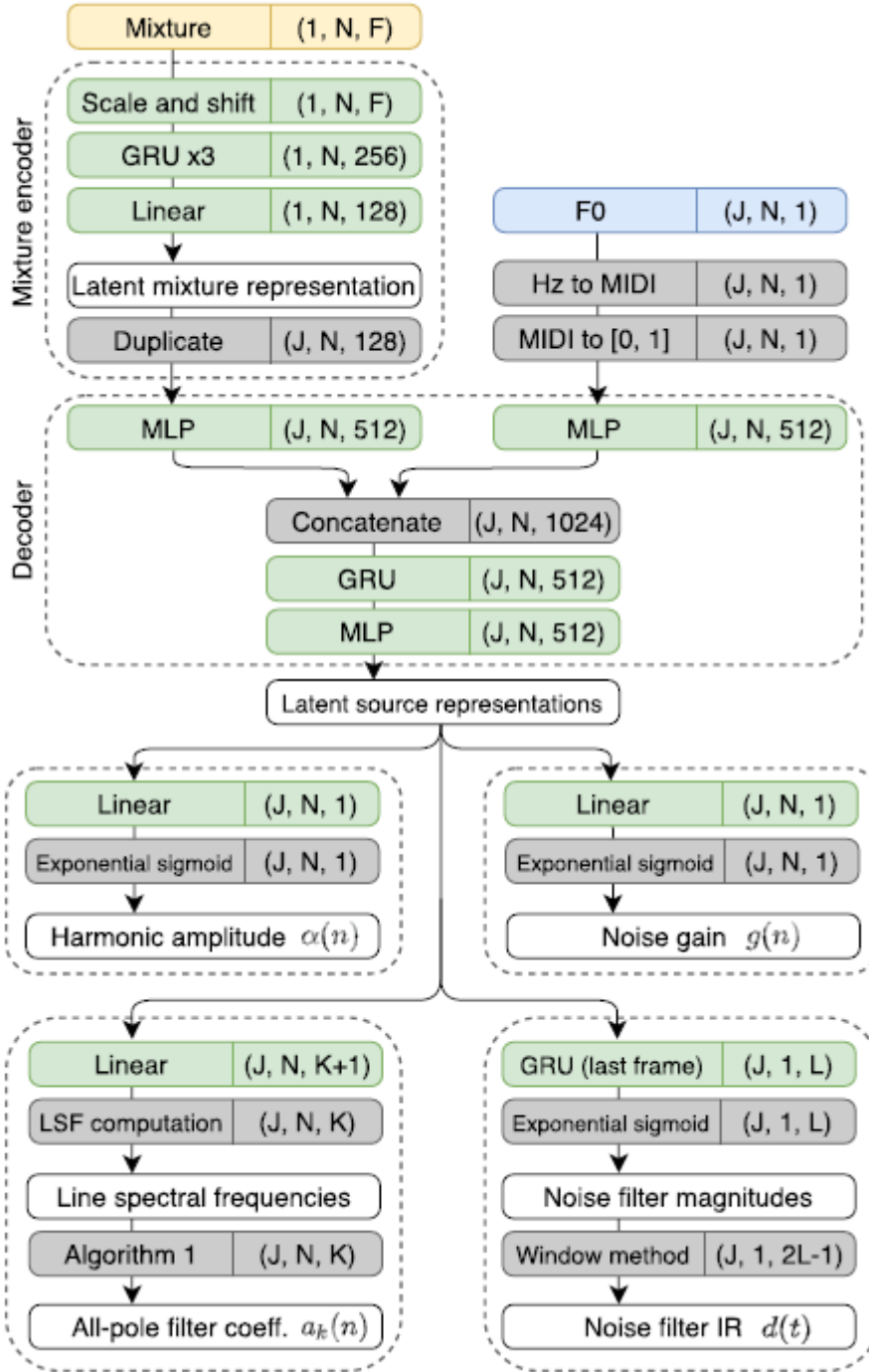


Fig 5: Visão geral das etapas de processamento para a estimativa de parâmetros. As transformações com parâmetros ajustáveis são mostradas em verde, as etapas de processamento pré-definidas em cinza e as saídas (intermediárias) em caixas brancas. A forma de saída de uma transformação é mostrada na parte direita da caixa. ([27], Imagem feita pelo autor).

O modelo de fontes diferenciáveis não impõem nenhuma restrição no tipo de rede neural ou arquitetura usada para estimar os parâmetros, no trabalho é usado uma DNN

simples, usando modelos de fontes paramétricas em uma separação baseada em aprendizagem profunda.

O sinal original é representado pelo logaritmo magnitude de seu espectrograma obtido através de uma STFT de $m(t)$. O espectrograma tem F bins de frequência e N quadros de tempo. Cada espectrograma é normalizado pela subtração da própria média e dividido pelo desvio padrão. Em seguida, cada bin de frequência é escalado e deslocado por escalares aprendidos dedicados.

Na [Figura 5](#) pode-se ver um overview da DNN e os passos do processamento para a estimação dos parâmetros. É usado camadas lineares e Redes Neurais Recorrente unidirecionais (RNN) com Unidades Recorrentes Fechadas (GRUs). O Multi-Layer Perceptron (MLP) consiste de três repetições de camada linear, normalização de camada, ativação Leaky ReLU.

O codificador do sinal original aprende uma Latent representation do sinal e então cria tantas duplicatas quantas forem as fontes. Cada cópia de representação latente é combinada com a informação F0 de uma fonte advinda do decoder. F0 disponibiliza o frame rate da STFT. Os valores de F0 são convertidos de Hertz para MIDI, um número normalizado no intervalo de [0, 1]. O decoder calcula a representação latente separa de cada fonte existente no sinal original. Os parâmetros do modelo de fontes são obtidos a partir da representação de fonte por uma última transformação com os parâmetros aprendidos (Linear layer or GRUs) seguido alguns passos de processamento pré-definidos. A amplitude harmônica de frame-wise $\alpha(n)$ e o ganho de ruído $g(n)$ são calculados com a linear layer com a função de ativação sigmoide exponencial, definida como:

$$y = y_{max} \cdot \text{sigmoid}(x)^{\log(10)} + 10^{-7} \quad (7)$$

- X e Y valor de entrada e saída respectivamente;
- y_{max} escalar que determina o limite superior de Y;

A amplitude harmônica é aumentada para a taxa de amostragem usando janelas de Han sobrepostas que produzem um $\alpha(t)$ suave. O ganho de ruído é necessário somente na taxa de quadros.

O filtro com resposta ao impulso $d(t)$ é invariante no tempo. Logo, a saída da rede a partir da qual $d(t)$ é calculado deve resumir as informações sobre todo o sinal da fonte. É obtido tal resultado processando a representação de fonte latente com uma RNN unidirecional com GRUs e, em seguida, usando apenas a saída no último período de tempo para processamento posterior. Este último frame é processado com a exponencial sigmoide a qual resulta em uma tensor de formato $(J, 1, L)$. O tensor tem L amostras das magnitudes de respostas de frequência unilateral do filtro de ruído para J fontes. As amostras definem um filtro FIR de fase zero de acordo com o método de amostragem de frequência. Utilizando o método de janelamento é obtido a resposta ao impulso $d(t)$.

A resposta ao impulso $r(t)$ do filtro harmônico invariante no tempo podem ser obtidas da mesma forma que $d(t)$ da saída de uma DNN. Os autores salientam que pode-se desejar filtros variantes no tempo usando uma linear layer para última transformação ou usando todos as GRU saídas. No escopo deste projeto o $r(t)$ foi ajustado manualmente.

Precisa-se de mais cuidados para obtenção de filtros de Infinite Impulse Reponse (IIR), assim como $\frac{1}{A(z)}$, a partir da saída da DNN, visto que se deve evitar que o filtro se torne instável. O filtro $\frac{1}{A(z)}$ de ordem K é totalmente definido pelos coeficientes de filtro a_k com $K \in \{1, \dots, K\}$. No entanto, nenhuma condição que garanta a estabilidade pode ser formulada diretamente para os coeficientes do filtro.

Consequentemente, os autores escolheram Line Spectral Frequencies (LSFs) para parametrizarem o filtro de todos os pólos. LSFs estão relacionados com as posições dos pólos do filtro e, por consequência, com a resposta em frequência. Os autores justificam esta escolha pelo fato de fornecerem uma parametrização interpretável e também permitem a formulação de restrições para controlar a resposta do filtro e podem ser interpolados.

$$A(z) = 1 - \sum_{k=1}^K a_k \cdot z^{-k}$$

Um filtro estável de fase mínima $\frac{1}{A_j(z)}$ de ordem K é definido por K LSFs (LSFs são os ângulos ω_k formados com as raízes no círculo unitário).

Para obter tais LSFs segue-se, as latent source representation são transformadas por uma linear layer que produz um tensor de forma $(J, N, K + 1)$. Isto é processado por um exponential sigmoid activation com $Y_{max} = 2$. O tensor resultante pode ser visto como $J \cdot N$ vetores $v \in R^{K+1}$. Os vetores são normalizados de tal forma que a soma das entradas $v_k = \pi$:

$$\bar{v} = \frac{v}{\sum_{k=1}^{K+1} v_k} \cdot \pi \quad (8)$$

Desta forma os K LSFs são obtidos com a soma:

$$\omega_k = \sum_{i=1}^k \bar{v}_i \quad \text{for } k = 1, \dots, K$$

As LSFs podem ser convertidas em coeficientes de filtro usando um algoritmo: Calcular coeficientes de filtro a_k a partir de ω_k .

```

Input:  $(\omega_k)_{k=1:K}$ 
Define:  $x_k = \cos(\omega_k)$ 
Initialize:  $p'_{-1} = q'_{-1} = 0$ ;  $p'_0 = q'_0 = 1$ 
Initialize:  $p'_1 = -2x_1$ ;  $q'_1 = -2x_2$ 
for  $k = 2$  to  $K/2$  do
   $p'_k = -2p'_{k-1}x_{2k-1} + 2p'_{k-2}$ 
   $q'_k = -2q'_{k-1}x_{2k} + 2q'_{k-2}$ 
  for  $i = (k-1)$  to  $1$  do
     $p'_i = p'_i - 2p'_{i-1}x_{2k-1} + p'_{i-2}$ 
     $q'_i = q'_i - 2q'_{i-1}x_{2k} + q'_{i-2}$ 
  end for
end for
for  $k = 1$  to  $K/2$  do
   $p_k = p'_k + p'_{k-1}$ 
   $q_k = q'_k - q'_{k-1}$ 
end for
for  $k = 1$  to  $K/2$  do
   $a_k = (p_k + q_k)/2$ 
   $a_{(K/2+k)} = (p_{(K/2-k+1)} - q_{(K/2-k+1)})/2$ 
end for
Output:  $(a_k)_{k=1:K}$ 

```

[Fig 6: Calculo dos coeficientes do filtro \$a_k\$ a partir de \$\omega_k\$.](#)

Em resumo:

- F0s são estimados a partir do sinal original e atribuídos as fontes usando métodos específicos;
- $a_k(n)$, $\alpha(t)$, $g(n)$ e $d(t)$ são obtidos com uma DNN;
- $r(t)$ é ajustado manualmente neste trabalho, mas outros utilizam DNN para isto.

3.3.2.3 Treinamento não supervisionado

Durante o treinamento a tarefa da DNN é reconstruir o sinal observado estimando os parâmetros correspondentes do modelo de fonte.

$$\tilde{m}(t) = \sum_{j=1}^J \tilde{s}_j(t) \quad (9)$$

- $\tilde{m}(t)$ é a soma dos sinais das fontes geradas pelo modelo de fonte.

Teoricamente o modelo de fonte é capaz de gerar um $\tilde{m}(t)$ que é perceptivamente igual ao $m(t)$. Visto que o offsets de fase absoluta são irrelevantes para percepção humana, o sinal estimado não precisa ter a mesma fase. Desta maneira, a perda de reconstrução é formulada como uma perda espectral multi-scale:

$$L_c = \|M_c - \tilde{M}_c\|_1 + \|\log M_c - \log \tilde{M}_c\|_1 \quad (10)$$

$$L_{rec} = \sum_c L_c \quad (14)$$

- M_c : Espectrograma de magnitude do sinal original.
- $\tilde{M}_c$: Espectrograma de magnitude do sinal estimado.
- $c = [2048, 1024, 512, 256, 128, 64]$: Tamanho da FFT utilizada para calcular STFT (Quadros se sobrepõem em 75%).

A separação das fontes é essencialmente assegurada pela atribuição das F0s às fontes similares à separação informada por pontuação/F0 com NMF. A DNN tem que estimar os parâmetros restantes para cada fonte em ordem de minimizar a perda. No teste, os parâmetros da DNN são fixados e uma solf-mask para a fonte j é obtida por divisão elementar:

$$\frac{\tilde{s}_j}{\sum_{j=1}^J \tilde{s}_j} \quad (15)$$

- $\tilde{s}_j$: Espectrograma de magnitude do sinal da fonte gerada $\tilde{s}_j$.

As estimativas finais da fonte no domínio do tempo, $\tilde{s}_j$ são obtidas por filtragem de Wiener usando as solf-masks.

3.3.2.4 Detalhes de implementação

O método foi desenvolvido utilizando o framework PyTorch [7]. Para os modelos de fontes diferenciáveis foi utilizado a biblioteca DDSP [8], porém o grupo reimplementou em PyTorch com a adição do Algoritmo 1, Figura 6, e um filtro all-pole.

O filtro all-pole neste framework apresenta certos desafios: Primeiramente, a filtragem autorregressiva é lenta, não permite o processo paralelo preciso dos quadros;

também, o filtro é variável no tempo, seus coeficientes são diferentes em cada quadro. Logo, faz-se necessário um cuidado em ordem de garantir uma transição suave entre quadros, para não correr artefatos.

O DNN utiliza uma taxa de quadras com base no tamanho da FFT e pelo tamanho do salto B usados para calcular a STFT do sinal original. Desta forma, o DNN estima um conjunto de K coeficientes de filtro para cada quadro. O filtro all-pole é aplicado a todos os quadros em paralelo usando a equação (3) de diferença para acelerar o processo de filtragem. Os estados iniciais $\tilde{s}(n, t)$ com $t \leq 0$ são definidos como zero para cada quadro. Os quadros de saída são multiplicados por uma janela de Hann e o sinal de saída é obtido pelo método de soma com sobreposição. A escolha do tamanho do salto B deve ser feita de tal forma que a janela de Hann respeite a condição de soma com sobreposição constante. Nos experimentos foi usado $B = T'/2$. É justificado que o uso da janela de Hann em conjunto da sobreposição de 50% torna a transição entre os quadros suave. Os autores salientam que os erros introduzidos ao definir os estados iniciais como zero em vez de levar em conta amostras do quadro anterior (o que é impossível no processamento paralelo) são insignificantes: Os erros são maiores no início de cada quadro, mas a importância é mitigada pela janela; como os coeficientes de filtro são diferentes em cada quadro, a importância das amostras do quadro anterior é reduzida.

Os autores descobriram que é fundamental implementar o Algoritmo 1 com precisão dupla (ponto flutuante de 64 bits), visto que é mais sensível a erros de arredondamento com o aumento da ordem do filtro, o que pode levar a filtros instáveis.

O sinal de excitação $e(t)$ é calculado seguindo: O componente harmônico $\alpha(t) \cdot h(t)$ e o ruído $\omega(t)$ são gerados no domínio do tempo para todo o comprimento do sinal T . Os filtros de resposta ao impulso finito (FIR) invariantes no tempo $r(t)$ e $d(t)$ e o ganho do ruído $g(n)$ são aplicados quadro a quadro no domínio da frequência, seguidos por uma operação de soma com sobreposição.

3.3.3 Experimentos

Os autores avaliaram a abordagem proposta em uma tarefa de separação de conjunto vocal a cappella. O objetivo era estimar os sinais individuais de J cantores a partir de sua mistura, do áudio original. Essa tarefa é uma boa escolha para avaliação,

pois as fontes em conjuntos vocais são homogêneas e correlacionadas. Além disso, a voz cantada é uma fonte musical desafiadora. Ela possui um envelope espectral fortemente variável no tempo e também produz sons sem conteúdo harmônico, como consoantes não vocalizadas. Além disso, apenas pequenas quantidades de dados para treinamento supervisionado estão disponíveis para a separação de conjunto vocal. Isso torna a aprendizagem não supervisionada uma alternativa importante.

3.3.3.1 Dados

Para o treinamento e validação dos dados, foi utilizado o conjunto de dados das "Bach Chorals" (BC) e o conjunto de dados do "Barbershop Quartet" (BQ). O conjunto BC contém 26 corais cantados por um quarteto vocal com as vozes Soprano, Alto, Tenor e Baixo (SATB). Já o conjunto BQ é composto por 22 músicas interpretadas por um quarteto vocal formado pelas vozes tenor, líder, barítono e baixo. Todas as vozes estão disponíveis individualmente para ambos os conjuntos. Isso nos permite comparar a abordagem não supervisionada proposta com as bases de referência supervisionadas.

A partir destas bases de dados foram criadas várias outras sub-bases com diferentes durações e combinações das vozes.

Como dados de teste, utilizamos o Conjunto de Dados de Canto Coral [7]. Ele é composto por três músicas interpretadas por um coro SATB com quatro cantores por voz. Todos os 16 sinais dos cantores estão disponíveis individualmente, o que permite avaliar a separação com métricas objetivas. Adicionamos os sinais dos cantores individuais (no máximo um por voz) para criar as misturas de teste. Para misturas com $J = 4$ cantores, o conjunto de teste tem uma duração de 6 minutos e 48 segundos. Para misturas com $J = 2$ cantores, o conjunto de teste tem uma duração de 40 minutos e 48 segundos, devido a mais combinações possíveis de vozes.

Os dados de treinamento, validação e teste foram reamostrados para uma taxa de amostragem de 16 kHz. Os exemplos de treinamento são trechos de 4 segundos de duração que são selecionados aleatoriamente do conjunto de treinamento. O conjunto de validação e o conjunto de teste são divididos em trechos fixos de 4 segundos de duração. Não há sobreposição em relação a cantores, músicas ou configuração de gravação entre os dados de teste e treinamento. Enquanto os dados de treinamento contêm uma quantidade considerável de reverberação, as gravações de teste possuem reverberação significativamente menor.

3.3.3.2 Configuração experimental

Foi realizado dois conjuntos de experimentos: um utilizando misturas de $J = 2$ cantores para treinamento e teste, e outro utilizando misturas de $J = 4$ cantores. Os valores de frequência fundamental (F0s) são obtidos a partir de... (O texto fornecido foi incompleto e não especifica de onde os valores de F0s são obtidos.)

Os valores de frequência fundamental (F0s) foram obtidos a partir dos sinais das misturas, áudios originais, utilizando o modelo de estimação de múltiplas F0 de Cuesta et al. [9]. Os autores utilizaram o modelo pré-treinado "Modelo 3", que está disponível online. Para a atribuição do F0 à fonte nos dados fornecidos, constatou-se que uma heurística simples é suficiente. Essa heurística é baseada nos mesmos princípios de soluções mais avançadas, como continuidade temporal de frequência, baixa probabilidade de cruzamento de vozes e mínima lacuna temporal dentro de uma voz [10], [11]. O estimador de F0 fornece x valores de F0 em cada quadro de tempo. Primeiro, foi processado todos os quadros em que $x = J$. Os valores de F0 são ordenados de acordo com a magnitude e atribuídos às vozes, assumindo que elas não se cruzam. Em seguida, os quadros restantes são processados. Quando $x < J$, é assumido que algumas vozes estão silenciosas. Foi atribuído cada valor de F0 à fonte que possui o valor de F0 mais próximo em um quadro anterior ou posterior (princípio de continuidade de frequência). O valor zero é atribuído às fontes silenciosas. No caso raro em que $x > J$, é ordenado os valores de acordo com a magnitude e silenciado os J valores de F0 usando o princípio de continuidade de frequência e é atribuído a eles às fontes.

Os espectrogramas das misturas, sinais originais, são calculados usando um tamanho de FFT (transformada rápida de Fourier) de $T = 512$ e um tamanho de salto (hop size) de $B = 256$ amostras. Portanto, eles têm $F = 257$ bins de frequência e $N = 250$ quadros de tempo. Foi fixado a resposta ao impulso $r(t)$ de forma que a resposta em frequência do filtro FIR caia a uma taxa de 6 dB/oitava, com uma frequência de referência de 200 Hz abaixo da qual a resposta é plana. Eles escolheram essa taxa porque ela leva em conta as características espectrais combinadas da fonte glotal e da radiação labial [12]. Salientam que estimar $r(t)$ com a rede neural profunda (DNN) não resultou em melhorias. Definiram a ordem do filtro "all-pole" como $K = 20$. Os espectrogramas dos sinais de fonte sintetizados $\tilde{S}_j$, usados para calcular as máscaras suaves (soft masks), são calculados com um tamanho de FFT de 2048 e um tamanho de salto de 256 amostras.

O treinamento foi realizado com o otimizador ADAM [13], um tamanho de lote (batch size) de 16 e uma taxa de aprendizado de 0.0001. O treinamento foi interrompido após 200 épocas consecutivas sem melhoria na perda de validação. O treinamento do modelo seguiu com a abordagem não supervisionada proposta no conjunto de treinamento completo e no conjunto de treinamento pequeno. Estes experimentos foram chamados de UnSupervised-Full (US-F) e UnSupervised-Small (US-S), respectivamente. Como referência, também treinaram o mesmo modelo de forma supervisionada nos mesmos dados. Nesse caso, a perda é calculada para cada estimativa de fonte individualmente usando seu alvo. A perda total é a soma das "perdas de fonte". Chamaram esses experimentos de SuperVised-Full (SV-F) e SuperVised-Small (SV-S). Como o método proposto depende das estimativas de F0 disponíveis, também foi avaliado sua robustez a erros de estimativa de F0.

3.3.3.3 Linhas de base

Os autores compararam a abordagem não supervisionada proposta com dois métodos livres de aprendizado e uma abordagem supervisionada. As bases de referência também exploram informações de F0 e calculam máscaras suaves para filtragem de Wiener. A primeira base de referência livre de aprendizado foi proposta em [9]. Ela aproxima o espectrograma de magnitude da mistura com uma decomposição simples de NMF (Fatoração de Matriz Não-Negativa, do inglês Non-Negative Matrix Factorization):

$$M \approx \hat{M} = WH \quad (16)$$

Na abordagem proposta em [14], é utilizado uma matriz $W \in R^{F \times R}$ contendo R templates espectrais e uma matriz $H \in R^{R \times N}$ que contém suas ativações ao longo de N quadros temporais. Em [9], W e H são inicializadas utilizando informações de uma partitura musical alinhada. É utilizado um template espectral por semitom. Nos experimentos, há informações disponíveis sobre o F0, que é mais preciso do que uma escala de semitons. Portanto, utilizaram uma escala com precisão de 1/10 de semitom. Os valores de F0 são convertidos de Hertz para números MIDI, que são arredondados para uma casa decimal para esse propósito. Os valores de F0 são utilizados para a inicialização e para a separação, determinando quais ativações pertencem a qual fonte. Após testar diferentes combinações, foi obtido os melhores resultados com um tamanho de FFT de 2048 e um tamanho de salto de 256 amostras para calcular os espectrogramas. Esse método foi chamado de NMF1.

A segunda base de referência livre de aprendizado é o método proposto por Durrieu et al. [15]. A fonte alvo é modelada com um modelo de fonte-filtro e as fontes residuais são modeladas com um NMF convencional. O método aproxima o espectrograma de potência da mistura M_{pow} como:

$$M_{pow} \approx \hat{M}_{pow} = \underbrace{(W^\Gamma H^\Gamma H^\Phi)}_{\text{filter}} \circ \underbrace{(W^{F0} H^{F0})}_{\text{source}} + \underbrace{(W^O H^O)}_{\text{residual}} \quad (17)$$

O método proposto utiliza a multiplicação elemento a elemento denotada por “ $\circ$ ”. A matriz $W^\Gamma \in R^{F \times P}$ contém P átomos espectrais que consistem em janelas de Hann deslocadas com uma sobreposição de 75%, garantindo que toda a faixa de frequência seja coberta. A matriz $H^\Gamma \in R^{P \times K}$ contém as ativações dos átomos espectrais para combinar e suavizar os filtros, enquanto $H^\Phi \in R^{K \times N}$ contém as ativações para combinar os filtros suavizados. A matriz $W^{F0} \in R^{K \times N}$ contém um conjunto fixo de U templates espectrais definidos pelo modelo de fonte glotal KLGLOTT88 [16]. Cada valor de $F0$ é representado por um template espectral em incrementos de $\frac{1}{20}$ de semitom, abrangendo uma faixa de frequência mínima e máxima. As ativações dos templates espectrais são armazenadas em $H^{F0} \in R^{U \times N}$. No artigo [15], as ativações H^{F0} são inicializadas usando informações de $F0$ da fonte predominante estimada por meio do modelo de sinal apresentado na equação (17). Já na abordagem do projeto, foi inicializado as ativações H^{F0} utilizando as informações de $F0$ obtidas a partir da estimativa de múltiplos tons [9]. Para as fontes residuais, as matrizes de templates espectrais $W^O \in R^{F \times R}$ e suas ativações $H^O \in R^{R \times N}$ são inicializadas aleatoriamente. No entanto, os parâmetros foram inicializados utilizando as informações de $F0$ correspondentes às fontes, como também foi feito no NMF1, o que resultou em melhorias. Essa abordagem é denominada NMF2. Os parâmetros a serem estimados são $\{H^\Gamma, H^\Phi, H^{F0}, W^O, H^O\}$. Para obter os melhores resultados no NMF2, utilizaram um tamanho de FFT de 1024 e um tamanho de salto de 128 amostras. Eles salientam que até onde sabem, as duas abordagens são consideradas entre os melhores métodos livres de aprendizado e informados para a separação de fontes musicais e homogêneas.

Além disso, eles realizaram a revisão literária do método de aprendizado profundo supervisionado baseado em $F0$ proposto por Petermann et al. [17] para a separação de conjunto vocal no conjunto de dados. Eles utilizam uma arquitetura clássica U-Net com um mecanismo de controle [18]. As informações de $F0$ são utilizadas para selecionar a

fonte alvo e guiar a separação. Para essa abordagem de referência, os espectrogramas do sinal misturado e da fonte alvo são calculados com um tamanho de FFT de 1024 e um tamanho de salto de 256 amostras. No momento dos testes, o filtro de Wiener é aplicado utilizando todas as J estimativas de fontes para calcular as máscaras suaves. O otimizador ADAM [13] é utilizado no treinamento, com um tamanho de lote de 16 e uma taxa de aprendizado de 0,001. O grupo treinou essa abordagem de referência tanto no conjunto de treinamento completo quanto no conjunto de treinamento reduzido, chamando os experimentos de Unet-F e Unet-S, respectivamente. É importante notar que todas as abordagens utilizam as informações de F_0 para a separação.

3.3.4 Resultados

3.3.4.1 Resultados experimentais

A qualidade da separação foi avaliada utilizando a métrica objetiva Scale-Invariant Source-to-Distortion ratio (SI-SDR) [19]. Ela foi calculada em quadros de avaliação de um segundo de duração, sem sobreposição, conforme geralmente feito para avaliação de separação de fontes musicais [20]. Os resultados para os casos de $J = 2$ e $J = 4$ fontes são mostrados na [Figura 7\(a\) e \(b\)](#), respectivamente. Os pontos de dados nos box-plots e violin-plots são os valores de SI-SDR em dB para todos os quadros de avaliação no conjunto de teste. Quadros da fonte alvo em que a soma dos quadrados das amostras é abaixo de um limite de 10 são considerados silenciosos e, portanto, excluídos da avaliação. Para os métodos em que números aleatórios estão envolvidos, a avaliação foi executada com cinco sementes diferentes para inicializar o gerador de números pseudoaleatórios. Esses métodos são NMF2 (inicialização aleatória de H^r e H^ϕ) e a abordagem proposta (ruído branco aleatório) usada nos experimentos US-F, US-S, SV-F e SV-S.

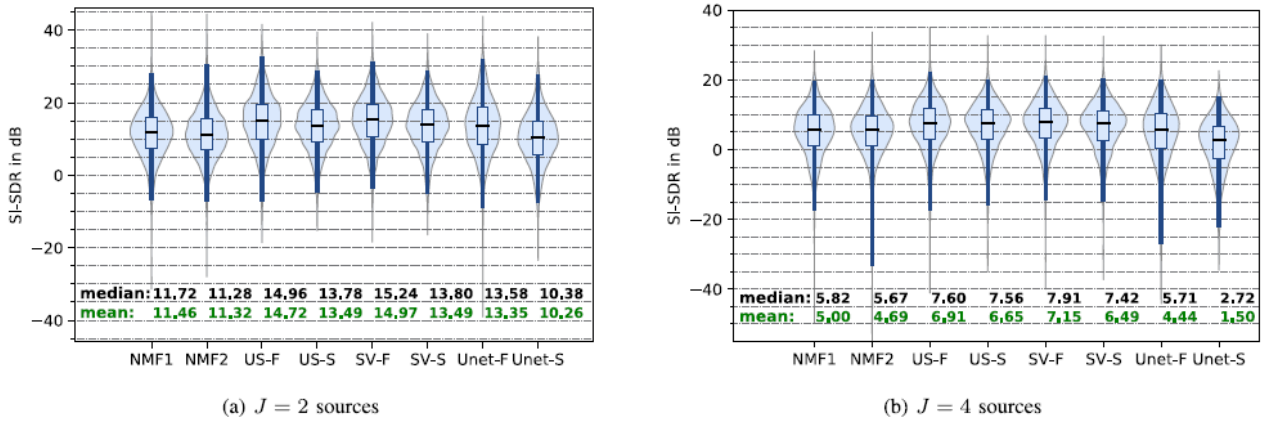


Fig 7: Gráficos de violino e box plot dos valores SI-SDR em dB para todos os quadros de avaliação. As caixas se estendem do primeiro ao terceiro quartil, as medianas são marcadas com uma linha horizontal preta. As barras do box plot (azul escuro) se estendem do primeiro ao percentil 99. Os gráficos de violino se estendem por todo o intervalo de dados. No gráfico (b), o NMF2 possui cinco valores atípicos entre -60 e -80 dB, que não são mostrados. ([27], Imagem feita pelo autor).

Foram realizados t-testes bilaterais [21] para avaliar se as médias das distribuições de pontuação SI-SDR são significativamente diferentes para cada par de experimentos em no estudo. Foi utilizado o teste de Levene [22] para avaliar se um par de distribuições de pontuação SI-SDR possui a mesma variância ou não. Se verdadeiro, a comparação foi feita com um t-test de Student. Se falso, utilizou-se o t-test de Welch [23]. Os valores-p resultantes [21] são mostrados na Figura 8(a) e (b) para $J = 2$ e $J = 4$, respectivamente. A maioria dos valores-p é extremamente pequena, da ordem de 10^{-4} ou menor. Isso indica que as médias correspondentes são significativamente diferentes. Pode-se observar que alguns valores-p são consideravelmente maiores. Nesse caso, é mais provável que as médias verdadeiras não sejam diferentes.

	NMF1	NMF2	US-F	US-S	SV-F	SV-S	Unet-F	Unet-S
NMF1	-	0,2154	$1,2 \cdot 10^{-164}$	$7,8 \cdot 10^{-72}$	$5,3 \cdot 10^{-193}$	$2,3 \cdot 10^{-69}$	$1,5 \cdot 10^{-31}$	$9,0 \cdot 10^{-16}$
NMF2	0,2154	-	$< 10^{-300}$	$3,2 \cdot 10^{-224}$	$< 10^{-300}$	$1,8 \cdot 10^{-220}$	$6,0 \cdot 10^{-52}$	$5,9 \cdot 10^{-19}$
US-F	$1,2 \cdot 10^{-164}$	$< 10^{-300}$	-	$2,3 \cdot 10^{-70}$	$3,9 \cdot 10^{-4}$	$3,9 \cdot 10^{-70}$	$2,2 \cdot 10^{-24}$	$2,8 \cdot 10^{-275}$
US-S	$7,8 \cdot 10^{-72}$	$3,2 \cdot 10^{-224}$	$2,3 \cdot 10^{-70}$	-	$9,7 \cdot 10^{-109}$	0,9507	0,2858	$3,5 \cdot 10^{-157}$
SV-F	$5,3 \cdot 10^{-193}$	$< 10^{-300}$	$3,9 \cdot 10^{-4}$	$9,7 \cdot 10^{-109}$	-	$3,7 \cdot 10^{-108}$	$6,1 \cdot 10^{-34}$	$< 10^{-300}$
SV-S	$2,3 \cdot 10^{-69}$	$1,8 \cdot 10^{-220}$	$3,9 \cdot 10^{-70}$	0,9507	$3,7 \cdot 10^{-108}$	-	0,3006	$2,3 \cdot 10^{-156}$
Unet-F	$1,5 \cdot 10^{-31}$	$6,0 \cdot 10^{-52}$	$2,2 \cdot 10^{-24}$	0,2858	$6,1 \cdot 10^{-34}$	0,3006	-	$5,7 \cdot 10^{-78}$
Unet-S	$9,0 \cdot 10^{-16}$	$5,9 \cdot 10^{-19}$	$2,8 \cdot 10^{-275}$	$3,5 \cdot 10^{-157}$	$< 10^{-300}$	$2,3 \cdot 10^{-156}$	$5,7 \cdot 10^{-78}$	-

(a) $J = 2$ sources

	NMF1	NMF2	US-F	US-S	SV-F	SV-S	Unet-F	Unet-S
NMF1	-	0,1739	$1,6 \cdot 10^{-18}$	$1,2 \cdot 10^{-14}$	$3,0 \cdot 10^{-23}$	$2,9 \cdot 10^{-12}$	0,0684	$1,5 \cdot 10^{-35}$
NMF2	0,1739	-	$2,1 \cdot 10^{-53}$	$2,0 \cdot 10^{-46}$	$5,3 \cdot 10^{-70}$	$1,2 \cdot 10^{-39}$	0,3357	$1,8 \cdot 10^{-43}$
US-F	$1,6 \cdot 10^{-18}$	$2,1 \cdot 10^{-53}$	-	0,0335	0,0554	$5,2 \cdot 10^{-4}$	$4,0 \cdot 10^{-22}$	$1,3 \cdot 10^{-121}$
US-S	$1,2 \cdot 10^{-14}$	$2,0 \cdot 10^{-46}$	0,0335	-	$2,9 \cdot 10^{-5}$	0,164	$2,0 \cdot 10^{-18}$	$1,5 \cdot 10^{-113}$
SV-F	$3,0 \cdot 10^{-23}$	$5,3 \cdot 10^{-70}$	0,0554	$2,9 \cdot 10^{-5}$	-	$2,5 \cdot 10^{-8}$	$2,2 \cdot 10^{-26}$	$3,0 \cdot 10^{-132}$
SV-S	$2,9 \cdot 10^{-12}$	$1,2 \cdot 10^{-39}$	$5,2 \cdot 10^{-4}$	0,164	$2,5 \cdot 10^{-8}$	-	$4,0 \cdot 10^{-16}$	$1,4 \cdot 10^{-107}$
Unet-F	0,0684	0,3357	$4,0 \cdot 10^{-22}$	$2,0 \cdot 10^{-18}$	$2,2 \cdot 10^{-26}$	$4,0 \cdot 10^{-16}$	-	$2,1 \cdot 10^{-21}$
Unet-S	$1,5 \cdot 10^{-35}$	$1,8 \cdot 10^{-43}$	$1,3 \cdot 10^{-121}$	$1,5 \cdot 10^{-113}$	$3,0 \cdot 10^{-132}$	$1,4 \cdot 10^{-107}$	$2,1 \cdot 10^{-21}$	-

(b) $J = 4$ sources

Fig 8: Os valores de p dos testes t de comparação em pares entre as distribuições dos valores SI-SDR para todos os experimentos. ([27], Imagem feita pelo autor).

Em geral, o SI-SDR é maior para a separação de misturas de duas fontes em comparação com o caso de quatro fontes. No entanto, o desempenho relativo dos métodos é o mesmo para ambos os casos, com exceção de que o Unet-F supera o NMF1

e o NMF2 quando $J = 2$, mas não quando $J = 4$. Isso está relacionado à pequena quantidade de dados de treinamento para um modelo de aprendizado profundo supervisionado.

O método proposto não supervisionado (US-F, US-S) apresenta desempenho melhor do que as bases de comparação. Seu desempenho é muito próximo ao do mesmo modelo treinado de forma supervisionada: SV-F é apenas ligeiramente melhor que US-F, enquanto SV-S e US-S têm o mesmo desempenho (valores- p de 0.9507 e 0.164 para $J = 2$ e $J = 4$). Isso significa que o método proposto alcança quase o mesmo desempenho, independentemente de fontes isoladas de destino estarem disponíveis para treinamento ou não. Isso pode ser explicado pelo fato de que as informações de F0 são usadas de maneira muito eficiente pelo método proposto. O F0 parametriza completamente o componente de fonte harmônica $h(t)$ e, portanto, define a fonte correspondente em grande parte. A rede neural precisa determinar os parâmetros restantes que, dadas as informações de F0, podem ser inferidos a partir da mistura. Portanto, os alvos de fonte isolados não carregam informações adicionais importantes.

Uma observação interessante é que o desempenho do método proposto não diminui drasticamente quando a quantidade de dados de treinamento é reduzida em 97% (US-F vs. US-S e SV-F vs. SV-S). Para $J = 2$, é possível observar uma diminuição no SI-SDR, porém ela é menor do que no caso do baseline supervisionado (Unet-F vs. Unet-S). Para $J = 4$, a diferença de desempenho do método proposto é muito pequena ao comparar o treinamento com o conjunto completo e o conjunto de treinamento reduzido. No caso da versão não supervisionada, essa diferença provavelmente não é significativa, pois o valor de p de 0,0335 para a comparação de US-F e US-S é maior do que a maioria dos outros valores de p . Em contraste, o SI-SDR do baseline Unet também diminui consideravelmente para $J = 4$, o que mostra que é benéfico integrar conhecimento do domínio na forma de modelos explícitos de fontes no modelo de separação.

Os autores acreditam que a principal diferença de desempenho entre US-S e Unet-S está relacionada ao uso da informação de F0. No Unet, a informação de F0 é usada globalmente. No caso proposto, a informação de F0 é diretamente explorada para produzir sinais harmônicos usando modelos explícitos de produção de fontes. A rede neural apenas infere as informações restantes (filtro do trato vocal, conteúdo de ruído, etc.). Isso facilita a tarefa para a rede neural, o que pode explicar o desempenho substancialmente superior.

Em resumo, a abordagem proposta de aprendizado profundo baseado em modelo não supervisionado para separação de fontes apresenta melhor desempenho em comparação com abordagens de aprendizado livre e supervisionadas puramente baseadas em dados. Além disso, ela é extremamente eficiente no aprendizado a partir

dos dados. O método é útil em muitos cenários em que é necessário separar fontes homogêneas e/ou quando apenas uma quantidade muito pequena de dados (possivelmente sem referência de verdade) está disponível para treinamento. Além da separação de coros, tais cenários podem incluir a separação de voz principal das vozes de fundo ou de músicas tradicionais com instrumentação menos comum. Uma vez que apenas as misturas são necessárias para o treinamento, o modelo proposto também pode ser treinado diretamente nas misturas a serem separadas. Com recursos computacionais suficientes, a otimização dos parâmetros também pode ser feita diretamente em cada mistura de teste individualmente, tornando o método independente de aprendizado.

3.3.4.2 Robustez para erros de estimativa F0

Neste artigo, os autores propuseram dois experimentos para analisar o impacto dos erros de estimativa multi-F0 no desempenho dos modelos, para o caso mais complexo de 4 fontes.

1. Criação de uma Anotação de Referência de multi-F0: Para este estudo de análise, construíram uma anotação de referência de multi-F0 para cada quadro de tempo (16 [ms]) do conjunto de testes. Como têm acesso às faixas musicais solo individuais para cada sinal de mistura, optaram por uma anotação automática das faixas solo usando um estimador de pitch de última geração (CREPE) [24]. Os modelos propostos são chamados de modelos oráculo quando executados usando essa anotação de referência de multi-F0.
2. Experimento 1: Análise dos Erros de Estimativa de F0: Este experimento tem como objetivo analisar ainda mais os resultados de avaliação obtidos na Seção de experimentos e resultados para o modelo US-F. Usando a anotação de referência de multi-F0 descrita acima como verdade fundamental, podemos identificar os erros cometidos pelo estimador de multi-F0 [9] nos experimentos para cada quadro de tempo de 16 [ms]. É observado um erro de Detecção de Atividade de Voz (VAD) quando uma das fontes é declarada ativa na anotação de referência de multi-F0 e não pelo nosso estimador de multi-F0, ou vice-versa. Quando não há erros de VAD, é possível avaliar a precisão do F0, que é definida como a diferença absoluta em Hertz entre o valor fornecido pela anotação de referência de multi-F0 e o estimador de multi-F0. Cada quadro de avaliação de 1 s do conjunto de testes pode então ser rotulado com o número de erros efetivos de VAD e a precisão média do F0 para as vozes detectadas corretamente. Cada quadro

de avaliação de 1 segundo do conjunto de testes pode ser rotulado com o número de erros efetivos de VAD e a precisão média do F0 para as vozes detectadas corretamente. O desempenho é impactado pelos erros de estimativa de F0, como era esperado. A degradação do desempenho permanece limitada para os quadros com quantidade moderada de erros, e o algoritmo é mais robusto aos erros de VAD. A distribuição de desempenho para os métodos supervisionados é geralmente similar, mas, no caso dos métodos Unet, há mais quadros com baixo SI-SDR quando não há erros de VAD/F0. As abordagens não supervisionadas parecem se beneficiar mais de uma estimativa correta de VAD e F0, como já discutido na Seção V-A, mas são um pouco mais frágeis no caso de erros de estimativa severos. Note que o nível de SI-SDR = -7,23 dB corresponde ao resultado obtido no conjunto de testes por um separador "dummy" no qual todas as fontes estimadas são réplicas atenuadas da mistura.

3. Experimento 2: Robustez a Estimativas Ruidosas de multi-F0: Foi avaliado o impacto de estimativas ruidosas de F0 no desempenho dos modelos, degradando manualmente as anotações de referência de F0. Duas alterações diferentes são consideradas:

- Transposição: Todas as frequências de F0 de referência são deslocadas para cima ou para baixo por um número pré-definido de Hertz ou uma ou várias oitavas.
- Desativação de voz: Todas as vozes de um determinado quadro de tempo de 16 [ms] têm uma probabilidade determinada de serem desativadas (os valores de multi-F0 correspondentes são definidos como zero), o que causa uma taxa pré-definida de erros de VAD.

Dos resultados obtidos para o efeito de transposição, pode-se observar que os modelos oráculo (os modelos propostos usando os valores de F0 de referência sem degradação) obtêm resultados melhores do que os mesmos modelos usando os valores de F0 fornecidos pelo estimador de multi-F0 (mais de +3 dB em média). A degradação do desempenho varia suavemente com a precisão em Hz dos valores de F0, demonstrando o grau de tolerância do algoritmo em relação a erros de estimativa de multi-F0. Para uma perturbação de menos de 5 Hz, o modelo ainda supera todos os modelos de referência. E, finalmente, os modelos não

aumentam o SI-SDR médio ($= -7,23$ dB) do separador "dummy" apenas quando o deslocamento excede aproximadamente 23 Hz (o que corresponde a pelo menos meio tom para todas as notas abaixo de A4 (440 Hz), o que pode ser considerado um erro grave). Para erros de oitava, a degradação é muito rápida quando os valores de F0 são transpostos para cima, mas permanece moderada ao transpor para baixo. Isso pode ser explicado pelo fato de que, quando a frequência fundamental é duas vezes menor que o valor real, todos os harmônicos pares da fonte coincidem exatamente com a posição da fonte alvo verdadeira.

A [Figura 9](#) apresenta os resultados quando uma ou várias vozes são localmente silenciadas. Os modelos permanecem eficientes quando a porcentagem de erros de VAD está abaixo de 20%. A forma aproximadamente linear da curva de degradação ressalta a robustez do modelo a esses erros de detecção. Os modelos treinados no conjunto de treinamento completo (US-F e SV-F) são mais robustos, mas o modelo supervisionado parece ser mais frágil do que o modelo não supervisionado quando treinado com o pequeno conjunto de dados.

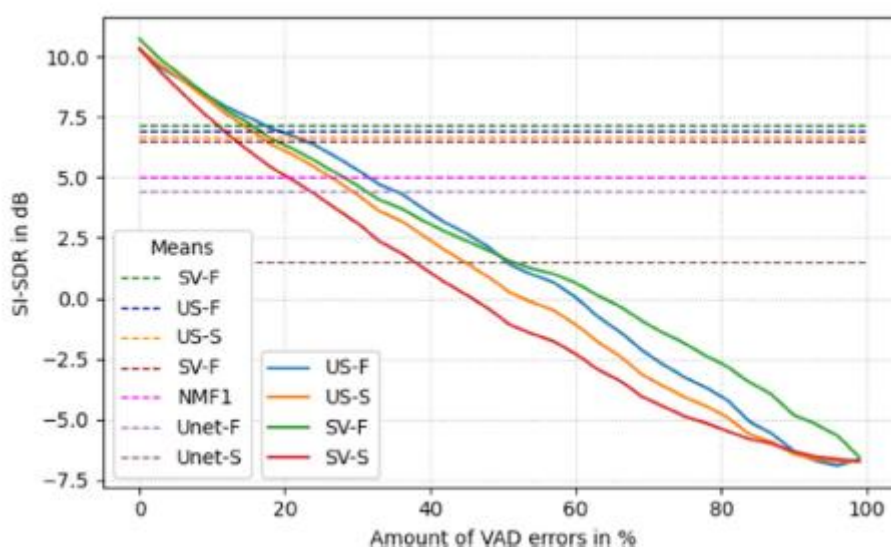


Fig 9: Efeitos de desativação de voz. ([27], Imagem feita pelo autor).

3.3.4.3 Limitações e perspectivas

A avaliação experimental demonstrou muitas vantagens da abordagem proposta em comparação com diversas alternativas. No entanto, existem algumas limitações. Primeiro, as abordagens pressupõem que o número de fontes seja conhecido. Embora

tenhamos mostrado que os métodos são um tanto robustos a erros moderados de VAD, eles estão atualmente limitados em casos em que o número de fontes é desconhecido. No entanto, a limitação mais marcante é mais precisamente que o método requer estimativas de F0 que são atribuídas às fontes. Como em todos os métodos de separação informados por F0, as fontes devem exibir principalmente conteúdo harmônico e ser monofônicas, para que a separação possa ser guiada pela informação de F0. Isso requer que boas estimativas de F0 possam ser obtidas para todas as fontes a partir da mistura, o sinal original. Como mostrado nos experimentos, isso é possível com métodos existentes. Progressos na pesquisa de estimativa de múltiplos F0 podem levar a melhorias adicionais. Uma extensão do método para fontes polifônicas, bem como a estimativa conjunta do F0 juntamente com outros parâmetros das fontes, pode ser uma direção interessante para trabalhos futuros, salienta os autores. Em sua forma atual, o modelo não teria um bom desempenho em misturas mais diversas de fontes musicais, como as contidas, por exemplo, no popular conjunto de dados MUSDB [\[25\]](#), que inclui tambores, fontes inarmônicas e polifônicas. Além disso, efeitos de áudio, como reverberação ou distorção, que podem ter sido aplicados às fontes, devem ser explicitamente modelados nos modelos de fonte e, portanto, devem ser conhecidos antecipadamente. Por fim, a complexidade espacial aumenta linearmente com o número de fontes a serem modeladas.

Nos experimentos mencionados acima, as estimativas finais das fontes foram obtidas por meio de filtragem de Wiener do sinal original. Para isso, foram obtidas máscaras suaves a partir dos sinais de fonte $\hat{S}_j$ gerados pelos modelos de fonte. Também avaliamos a qualidade dos sinais gerados $\hat{S}_j$ como estimativas de fonte. A métrica usada para essa avaliação foi a relação sinal-ruído espectral [\[26\]](#). Ela pode ser vista como um SI-SDR calculado em espectrogramas de magnitude. Utilizamos essa métrica espectral porque sabe-se que a fase dos sinais gerados não é a mesma da fase dos sinais de referência, tornando uma avaliação no domínio do tempo inaplicável. Em termos dessa métrica, a qualidade dessas estimativas de fonte foi inferior às linhas de base e a $\hat{S}_j$ obtido usando máscaras suaves. Isso ocorre porque a síntese dos sinais $\hat{S}_j$ é menos restrita do que o mascaramento da mistura. A saída do mascaramento é limitada pelo conteúdo de frequência da mistura, uma vez que o mascaramento pode apenas manter ou remover tal conteúdo. Em contraste, o conteúdo de frequência que não está presente em nenhuma

fonte pode estar contido em $\hat{S}_j$. De fato, a DNN tende a superestimar o conteúdo de ruído das fontes. Embora isso seja claramente audível em $\hat{S}_j$, nenhum ruído é adicionado em $\hat{S}_j$.

No entanto, acredita-se que as estimativas de fonte geradas por modelos paramétricos são um objetivo válido para pesquisas futuras. Os autores dizem que elas fornecem uma parametrização completa do sinal de mistura que pode ser explorada em tarefas como transferência de timbre ou estilo, transposição e edição de melodia de fontes individuais. O autor comenta que há alguns exemplos de $\hat{S}_j$ e $\tilde{M}$ que podem ser avaliados.

3.3.5 Discussão

Neste trabalho, foi apresentado um método para separação de fontes de áudio (musical) que supera duas limitações dos métodos de aprendizado profundo supervisionados de última geração: eles não separam fontes homogêneas e exigem grandes conjuntos de dados de misturas com as respectivas fontes isoladas para treinamento. Foi proposto um novo método não supervisionado baseado em aprendizado profundo. Ele integra conhecimento de domínio na forma de modelos paramétricos diferenciáveis de fontes em um método orientado por dados e explora informações de F0. Os experimentos mostram que ele supera baselines sem aprendizado e supervisionados. Além disso, o método funciona bem mesmo quando treinado com menos de três minutos de dados de áudio. Isso permite aplicar separação poderosa baseada em aprendizado profundo em domínios onde os dados de treinamento são caros ou inexistentes.

No âmbito do projeto atual, este trabalho proporcionou até então a possibilidade da utilização de um separador de fonte na entrada, ou seja, para focalização do nosso algoritmo apenas no violão, tentando isolar ao máximo a fonte que será trabalhada, visto que este trabalho almeja a transcrição musical em tempo real, a exclusão de ruídos externo com certeza terá grande impacto no resultado.

4 Base de dados

Como ideia inicial o objetivo seria fazer um estimador de erro baseado direto no resultado final do trabalho, sobre a partitura/tablatatura, entretanto o cálculo do erro baseado na comparação dos arquivos MIDI de entrada, e saída se mostrou bem mais promissor. Visto isso, um problema que havia surgido, da possibilidade da existência de uma base de dados extensa, com horas de reprodução, com a música tocada em uma boa qualidade e no tempo do metrônomo, já acompanhada da partitura usada, se resolveu, visto que a comparação de arquivos MIDI é bem mais simples e há muitas bases de dados desse formato, como Free MIDI, MIDI World, Classic Cat, MuseScore, 8notes.

Entretanto, não querendo limitar as soluções, há base de dados apenas com músicas de violão, sendo elas Ultimate Guitar, Songsterr, 911Tabs, Chordie e Tabs4Acoustic, as quais seriam utilizadas como entrada e seus respectivos arquivos MIDI usados no cálculo do erro.

Para o treinamento de uma DNN (Deep Neural Network) para a transcrição musical em tempo real de violão, pode-se considerar as seguintes características para a base de dados:

1. Base de dados de áudio de violão: Uma base de dados que contenha gravações de áudio de trechos de violão tocados em diferentes notas, acordes, ritmos e estilos musicais. Essa base de dados pode ser composta por gravações de performances ao vivo ou de estúdio, com diferentes técnicas de gravação e características musicais variadas.
2. Base de dados de transcrições musicais: Uma base de dados que contenha transcrições musicais precisas dos trechos de violão presentes na base de dados de áudio. Essas transcrições podem incluir informações sobre notas, acordes, durações e outros elementos musicais relevantes.
3. Base de dados de áudio misturado: Uma base de dados que contenha gravações de áudio que envolvam violão tocando junto com outros instrumentos musicais ou acompanhamentos. Essa base de dados pode ajudar a simular cenários mais realistas de transcrição musical em tempo real, onde o violão está presente em uma mixagem de áudio (Está última pode variar de acordo com as escolhas de implementação).

Combinando essas bases de dados, é possível treinar a DNN para aprender padrões e características do som do violão em diferentes contextos musicais. Isso permitirá que a rede neural faça a transcrição em tempo real do violão ao receber o sinal de áudio em tempo real durante a execução. É importante ressaltar a necessidade da diversificação e representatividade das bases de dados o suficiente para abranger uma variedade de estilos musicais, técnicas de violão e variações na execução.

Para uma base de dados que contenha músicas de violão e suas respectivas partituras, pode-se considerar as seguintes opções:

1. MuseScore (<https://musescore.com/>): O MuseScore é uma comunidade online que oferece um vasto acervo de partituras gratuitas e de código aberto. Você pode encontrar uma variedade de músicas para violão, desde peças clássicas até canções populares.
2. IMSLP (<https://imslp.org/>): O IMSLP (International Music Score Library Project) é uma biblioteca online que oferece acesso gratuito a partituras de domínio público. Você pode pesquisar por composições para violão e encontrar partituras de compositores clássicos.
3. Chordify (<https://chordify.net/>): O Chordify é uma plataforma que fornece acordes e partituras para músicas populares. Embora nem todas as músicas tenham partituras completas, você pode encontrar acordes e diagramas de acompanhamento para músicas com violão.
4. Berklee Shares (<https://www.berkleeshares.com/>): Berklee Shares é uma plataforma que oferece recursos educacionais, incluindo partituras e arranjos para guitarra e violão. Embora algumas partituras possam exigir uma assinatura, existem opções gratuitas disponíveis.

Estas são possíveis bases de dados a serem usadas, as escolhas estão em aberto por enquanto, mas com a definição de um escopo mais restrito poder-se-á afirmar isto.

5 Metodologia

5.1 Escolha da base de dados

A transcrição musical em tempo real de violão é um projeto desafiador que busca automatizar o processo de converter a música tocada em um instrumento para partitura. Neste contexto, o MuseScore surge como uma escolha justificada e eficiente para servir como base de dados nesse trabalho, que visa até então gerar um arquivo MIDI como resultado final, em vez de uma partitura. É válido salientar que futuramente pode-se mudar a base, visto que há questões legais envolvidas, mas a priori, mostra-se uma boa opção.

Uma das principais razões para utilizar o MuseScore é a sua vasta biblioteca de partituras. O MuseScore possui uma ampla coleção de partituras disponíveis gratuitamente, abrangendo diversos gêneros musicais e incluindo um número significativo de peças para violão. Ao utilizar o MuseScore como base de dados, podemos aproveitar essa riqueza de partituras para fins de comparação e referência durante o processo de transcrição em tempo real.

Outra vantagem do MuseScore é a sua comunidade ativa de usuários. Essa comunidade contribui constantemente com novas partituras, dicas e truques, fornecendo um valioso suporte e conhecimento para o desenvolvimento de um projeto de transcrição musical em tempo real de violão. Podemos aproveitar a experiência compartilhada pelos usuários do MuseScore para aprimorar nossos algoritmos de transcrição e resolver desafios técnicos que possam surgir ao longo do projeto.

5.2 Restrições impostas a priori

A geração de um arquivo MIDI em um trabalho de transcrição musical apresenta vantagens em relação à geração de uma partitura da música transcrita. O formato MIDI oferece uma representação mais rica, contendo informações detalhadas sobre as notas, instrumentos, dinâmica, articulações e outros aspectos musicais. Isso permite capturar nuances de execução importantes, como tempo, ataque e duração das notas, resultando em uma transcrição mais precisa e expressiva.

Outra vantagem é a facilidade de reprodução do arquivo MIDI. Ele pode ser interpretado e reproduzido em diferentes dispositivos e softwares musicais, como

sintetizadores, sequenciadores e programas de gravação. Isso permite ouvir imediatamente o resultado da transcrição em várias plataformas, agilizando o processo de audição e análise.

A flexibilidade para edições futuras também é uma vantagem do arquivo MIDI. Sua estrutura editável permite ajustar notas, dinâmicas, articulações e até mesmo trocar o instrumento ou a sonoridade. Isso facilita a correção de erros e aprimoramentos na transcrição, sem a necessidade de reescrever a partitura inteira.

Além disso, o formato MIDI é amplamente aceito e compatível com várias ferramentas musicais, como editores de partitura, programas de produção musical, software de notação, DAWs (Digital Audio Workstations) e até mesmo softwares de aprendizado de máquina. Essa compatibilidade possibilita explorar recursos e possibilidades oferecidos por essas diferentes ferramentas para análises, arranjos, mixagens e outras manipulações musicais.

Embora a geração de uma partitura seja importante, a geração de um arquivo MIDI como resultado da transcrição musical apresenta vantagens significativas. A representação mais rica, a facilidade de reprodução, a flexibilidade para edições futuras e a integração com outras ferramentas musicais tornam o arquivo MIDI uma escolha preferencial em trabalhos de transcrição musical, especialmente quando se considera o uso de técnicas de aprendizado de máquina, como as Redes Neurais Profundas. No entanto, é importante ressaltar que a geração final da partitura não é descartada, mas será realizada em uma etapa separada, posterior à detecção e processamento dos dados musicais.

5.3 Separador de fontes

O processamento de separação de fontes (source separation) no contexto da transcrição musical de violão pode ser extremamente interessante e útil. A separação de fontes refere-se à técnica de separar diferentes elementos sonoros presentes em uma gravação, isolando-os uns dos outros. No caso da transcrição musical de violão, essa técnica pode ajudar a separar o som do violão de outros sons ou instrumentos presentes na gravação, como vozes ou acompanhamentos, melhorando assim:

1. Clareza na transcrição
2. Redução de ruídos e interferências
3. Foco na análise do violão

4. Transcrição de acompanhamentos

No entanto, é importante ressaltar que a separação de fontes não é uma tarefa trivial e pode apresentar desafios, especialmente em gravações com sobreposições complexas de sons. O resultado da separação dependerá da qualidade da gravação original e da técnica utilizada. É necessário escolher cuidadosamente as ferramentas e algoritmos adequados para obter os melhores resultados na separação de fontes. Desta forma, esta etapa por enquanto é encarada como uma implementação secundária.

5.4 Utilização de BiLSTM

A utilização de BiLSTM (Bidirectional Long Short-Term Memory) para transcrição musical oferece vários benefícios. As redes neurais recorrentes são capazes de capturar padrões musicais complexos ao considerar a estrutura temporal da música. A arquitetura bidirecional permite que o modelo leve em conta o contexto passado e futuro das notas, resultando em previsões mais precisas. Além disso, a BiLSTM lida bem com variações de ritmo e tempo, aprende relações de longo prazo e captura a polifonia das gravações musicais. No entanto, é importante destacar que a escolha da arquitetura depende do contexto da tarefa de transcrição musical e das características dos dados disponíveis.

5.5 Transformar MIDI em partitura

Existem várias bibliotecas disponíveis para transformar arquivos MIDI em partituras via programação, como o Music21 (Python), JFugue (Java), MIDIUtil (Python) e MIDI.js (JavaScript). Essas bibliotecas oferecem recursos para a leitura, manipulação e conversão de arquivos MIDI em uma representação de partitura. A escolha da biblioteca dependerá da linguagem de programação e dos recursos específicos desejados.

6 Desenvolvimento

6.1 Criação da base de dados

Como supracitado na seção [5.1](#) deste trabalho, foi escolhido a utilização do MuseScore como base de dados visto seus benefícios.

Consequentemente, a confecção desta fase foi feita de maneira ruidosa, uma vez que a plataforma é alimentada pela comunidade e não há uma padronização, desta forma, além de filtrar a base de dados por músicas de violão solo foi necessárias várias horas dedicadas a escuta-las, visando verificar se em toda extensão da música havia apenas um instrumento, anotando a existência de partitura e/ou tablatura e descartando os sons que continham metrônomo de fundo, assim como acompanhamento de acordes tocados por outros instrumentos, ou até mesmo sons que não estavam descritos na partitura.

Consequentemente, esta etapa demandou mais semanas de desenvolvimento do que o planejado, mas devidos os esforços, foi concluída. Seguindo a metodologia de escolha supracitada, foi criado uma base de dados de músicas de violão solo em MIDI contendo 169 peças, considerando que a extração de features trabalha no janelamento das músicas, acredito ser um bom número inicial, podendo ser acrescido na fase de treinamento baseado do desempenho da Bi-LSTM.

É válido frisar que não existe uma base de dados de músicas de violão solo em extensão MIDI na internet, esse fator acrescentou uma dificuldade não prevista de antemão.

6.2 Pré-processamento dos dados

Um arquivo MIDI é um formato que armazena dados digitais sobre uma música, informações que a compõem, estas sendo notas musicais, durações, dinâmica, volume, andamento entre outras informações, porém isto não é uma onda. Desta forma, em ordem de preparar uma base de dados, faz-se necessário a conversão de MID para o formato WAV, visto que os métodos de extração de featureres são aplicados sobre curvas.

Desta forma, sabe-se a necessidade da conversão, porém como fazê-la? A resposta não é tão complexa, porém necessita de alguns apontamentos. O FluidSynth é uma ferramenta essencial para transformar arquivos MIDI em faixas de áudio prontas para

serem ouvidas. No processo de conversão, o FluidSynth depende de um componente crítico chamado soundfont.

Os soundfonts são arquivos que contêm amostras de instrumentos musicais digitais, bem como as instruções sobre como essas amostras devem ser usadas para criar o som. Cada instrumento, como piano, violino, flauta, entre outros, tem suas próprias amostras que representam diferentes notas, volumes e nuances musicais. Os soundfonts variam em termos de qualidade e alcance, indo de soundfonts simples com apenas alguns instrumentos básicos até soundfonts abrangentes com uma ampla variedade de sons e articulações.

Mudar o soundfont durante a conversão de um arquivo MIDI para WAV pode ter um impacto negativo significativo no resultado de uma transcrição musical em tempo real. Ao alterar o soundfont, as amostras e as características musicais podem variar consideravelmente, resultando em uma mudança no timbre, qualidade e até mesmo nas nuances musicais. Isso não apenas compromete a coesão da música, mas também pode fazer com que as notas soem incorretas ou desarticuladas. Para uma transcrição musical em tempo real precisa e expressiva, manter o mesmo soundfont é fundamental para garantir que o desempenho musical seja representado de maneira fiel e autêntica.

Visto isso, durante o desenvolvimento, foi utilizado o soundfont `generaluser.sf2`, o qual será mantido, inclusive caso necessário expandir a base de dados.

Além da conversão para WAV, os sons tiveram sua taxa de amostragem normalizada para 44100, assim como todos foram convertidos para Mono. A escolha de trabalhar com músicas no formato mono em uma transcrição de música em tempo real é uma consideração significativa para a eficácia do processo. Isso ocorre porque o formato mono, que consiste em um único canal de áudio, oferece várias vantagens em comparação com o formato estéreo, que possui dois canais. Essas vantagens são fundamentais para a precisão da análise, a eficiência do sistema e a qualidade da transcrição.

Uma das principais razões para dar preferência ao áudio mono é a simplicidade de processamento. Em um sinal mono, todas as informações sonoras estão contidas em um único canal, o que torna a análise mais direta e menos complexa. Em uma transcrição de música em tempo real, onde a velocidade de resposta é crucial, a simplicidade de

processamento contribui para manter a latência baixa, garantindo que as notas musicais sejam transcritas com precisão e em tempo real.

Além disso, o uso de áudio mono resulta em economia de recursos computacionais. O processamento de áudio estéreo exige mais recursos, já que envolve a análise de dois canais separados. Em um cenário de transcrição em tempo real, minimizar a carga de processamento é essencial para manter a eficiência do sistema. Isso também contribui para a redução de custos em termos de recursos de hardware e potência de processamento.

Por fim, a concentração nas notas musicais é o objetivo principal de uma transcrição musical. Trabalhar com áudio mono permite uma análise mais precisa das informações musicais essenciais, facilitando a detecção e transcrição das notas, tornando o resultado mais fiel à interpretação musical original.

6.3 Extração de características

Inicialmente, esta etapa parecia simples, visto que seria apenas aplicar a transformada de Fourier e organizar os dados, porém, o cenário mudou. Como dito, a ideia inicial era utilizar apenas o STFT, porém com base no trabalho [\[3\]](#), buscava-se seguir a mesma forma de análise, ou seja, encarar cada nota única do violão como sendo um instrumento isolado, a partir desta ideia foi levantado todas as notas do braço do violão com as respectivas classificações MIDI:

Traste	E2	A2	D3	G3	B3	E4
0	E2 (40)	A2 (45)	D3 (50)	G3 (55)	B3 (59)	E4 (64)
1	F2 (41)	A#2/Bb2 (46)	D#3/Eb3 (51)	G#3/Ab3 (56)	C4 (60)	F4 (65)
2	F#2/Gb2 (42)	B2 (47)	E3 (52)	A3 (57)	C#4/Db4 (61)	F#4/Gb4 (66)
3	G2 (43)	C3 (48)	F3 (53)	A#3/Bb3 (58)	D4 (62)	G4 (67)
4	G#2/Ab2 (44)	C#3/Db3 (49)	F#3/Gb3 (54)	B3 (59)	D#4/Eb4 (63)	G#4/Ab4 (68)
5	A2 (45)	D3 (50)	G3 (55)	C4 (60)	E4 (64)	A4 (69)
6	A#2/Bb2 (46)	D#3/Eb3 (51)	G#3/Ab3 (56)	C#4/Db4 (61)	F4 (65)	A#4/Bb4 (70)
7	B2 (47)	E3 (52)	A3 (57)	D4 (62)	F#4/Gb4 (66)	B4 (71)
8	C3 (48)	F3 (53)	A#3/Bb3 (58)	D#4/Eb4 (63)	G4 (67)	C5 (72)
9	C#3/Db3 (49)	F#3/Gb3 (54)	B3 (59)	E4 (64)	G#4/Ab4 (68)	C#5/Db5 (73)
10	D3 (50)	G3 (55)	C4 (60)	F4 (65)	A4 (69)	D5 (74)
11	D#3/Eb3 (51)	G#3/Ab3 (56)	C#4/Db4 (61)	F#4/Gb4 (66)	A#4/Bb4 (70)	D#5/Eb5 (75)
12	E3 (52)	A3 (57)	D4 (62)	G4 (67)	B4 (71)	E5 (76)
13	F3 (53)	A#3/Bb3 (58)	D#4/Eb4 (63)	G#4/Ab4 (68)	C5 (72)	F5 (77)
14	F#3/Gb3 (54)	B3 (59)	E4 (64)	A4 (69)	C#5/Db5 (73)	F#5/Gb5 (78)
15	G3 (55)	C4 (60)	F4 (65)	A#4/Bb4 (70)	D5 (74)	G5 (79)
16	G#3/Ab3 (56)	C#4/Db4 (61)	F#4/Gb4 (66)	B4 (71)	D#5/Eb5 (75)	G#5/Ab5 (80)
17	A3 (57)	D4 (62)	G4 (67)	C5 (72)	E5 (76)	A5 (81)
18	A#3/Bb3 (58)	D#4/Eb4 (63)	G#4/Ab4 (68)	C#5/Db5 (73)	F5 (77)	A#5/Bb5 (82)
19	B3 (59)	E4 (64)	A4 (69)	D5 (74)	F#5/Gb5 (78)	B5 (83)

[Fig 10: Mapeamento das notas do braço de um violão clássico de 19 trastes em afinação padrão.](#)

Com base neste mapeamento foi constatado que um violão clássico acústico de 19 traste em afinação padrão pode-se reproduzir 44 notas únicas, ou seja, de E2 (40) até B5 (83) sem considerar a diferenciação das oitavas (este foi o principal motivo do descarte da possível geração de tablaturas, para diferenciar oitavas deve-se-á criar uma RNN apenas para classificar a posição exata da nota no braço do violão, um possível tema de trabalho de graduação que pode usar a base de dados que eu criei), vale salientar que este trabalho visa apenas criar o MIDI base para confecção da partitura, porém sem considerar dinâmica, altura e outras informações que poderiam ser acrescentadas em trabalhos de mestrado e/ou doutorado.

6.3.1 Criação da matriz X de treinamento

O algoritmo desta etapa realiza o carregamento de cada música em WAV, como mono, em um loop. Com base na duração da música e na janela de tempo de análise escolhida, 0,05 segundos, tem-se o número desejado de frames que deve existir no resultado do mel-espectrograma, deve-se ter em mente que no tempo 0,05 segundos equivale a 4 frames na frequência, devido a isto, tem-se que multiplicar o número de frames desejados calculados com a duração da música e tempo da janela por 4 como se

segue:

$$\text{int}\left(\frac{\text{Duração da música}}{0,05}\right) * 4 = \text{número de frames desejados para a música em análise}$$

Como pode ser notado, não é utilizado STFT, isso foi necessário visto que o objeto de trabalho é música, a STFT abrange uma faixa muito grande de frequências, o que acaba gerando dados de análise desnecessários, em busca de melhorar o processamento e simultaneamente reter apenas dados relevantes, foi alterado a empreitada para utilização do mel-espectrograma.

Consequentemente, com base no número desejado de frames calcula-se o hop_length e n_fft a fim de ajustar os frames advindos do mel-espectrograma referente a música analisada em questão. Também é válido salientar que o áudio passa por uma normalização assim como o mel-espectrograma por uma padronização, estas etapas podem ser vistas com detalhe no código em anexo.

Em última análise, cada mel-espectrograma no fim passa por um ajuste de dimensão, visando reter a quantidade de frames indicada pelo número de frames desejados supracitado, visto que as janelas são grandes e com sobreposição e, também, considerando que as janelas cortadas variam de 1 até no máximo 8 constatado por empirismo, é uma informação que se pode retirar sem muitos encargos. Este ajuste de dimensão é necessário visto que o número de amostras de uma janela no domínio de tempo de 0,05 segundos são 4 frames no domínio da frequência, esta análise fica mais clara ao se ler a etapa de preparação dos vetores de respostas Y.

Após a extração do mel-espectrograma de cada músicas nos moldes supracitados, é realizado o algoritmo de fato para aglutinação dos mel-espectrogramas de cada música, chegando-se no resultado mostrado na [figura 1](#). O ajuste de acréscimo de silêncio, vetores zero, para se ter uma dimensão inteira para se iterar em janelas de 5 segundos no trabalho em questão, 0,05 segundos neste, ou no domínio do tempo, de 4 em 4 frames, não foi necessário, visto que a etapa anterior de cálculo de hop_length e n_fft já mantém as dimensões sendo múltiplas de 2.

6.3.2 Criação da matriz Y de treinamento

O algoritmo desta etapa realiza o carregamento de cada música em MIDI, em um loop. Desta forma, com base na iteração das mensagens do MIDI é criada uma lista de tuplas que indica o tempo e a(s) nota(as) tocadas em cada instante.

Consequentemente, é criada uma lista com N listas, por sua vez, o N advém do número de eventos desejados (Seria o número de frames desejados em X, porém sem multiplicar por 4):

$$\text{int} \left(\frac{\text{Duração da música}}{0,05} \right) = \text{número de eventos}$$

À lista principal é adicionada uma sub-lista de dimensão 45 para cada evento da música, ou seja, 1ª posição sendo o tempo em que a nota(as) ocorreram e as demais 44 sendo preenchidas de 0 ou 1 de acordo com quais notas foram tocadas neste tempo. Este conjunto de tempo e notas tocadas é chamado de evento.

Por fim, Y trata-se de uma matriz (N eventos x 45), em que se tem empilhado os eventos de todas as músicas. Como já salientado, a matriz X de treinamento foi confeccionada seguindo a [figura 1](#).

6.3.3 Discutindo X e Y

É válido esclarecer o que se busca na relação entre X e Y. No domínio do tempo, a ideia foi trabalhar com janelas de 0,05 segundos, esse janelamento de 50 milissegundos foi escolhido pois consegue-se isolar ao máximo a ocorrência das notas, por sua vez, no domínio da frequência, uma janela de 0,05 segundos equivale a 4 frames advindos do mel-espectrograma, desta forma, processando as mesmas músicas na mesma ordem tanto no algoritmo de X, quanto de Y tem-se que para as 4 primeiras linhas da matriz X a resposta esperada é uma linha da matriz Y e assim consequentemente, de tal forma que a quantidade de linhas de X é 4 vezes a quantidade de linhas de Y. O grande problema nesta empreitada é realizar o cálculo do `hop_length` e `n_fft` a fim de se ter a quantidade de frames advindas do mel-espectrograma em mesma quantidade à calculada para o número desejados de frames, visto que cada música tem duração variável, essa análise é necessária visto que a remoção empírica das janelas sobressalentes, em resumo, é de

certa forma a remoção de dados que poderiam ser analisados pela BiLSTM, por mais que o descarte dos frames sobressalentes sejam pequenos, de no máximo 8 janelas, ou seja, duas janelas de 0,05 segundos no domínio do tempo, quanto mais próximo a quantidade de frames advindas da transformada for do desejado calculado, melhor, menos descarte, mais dados dentro da análise.

É notório que ao longo da confecção dos algoritmos, diversos métodos foram implementados, inclusive com empreitadas melhores do que o ajuste incerto de frames da transformada, porém, nenhum deles, dos ademais que foram implementados, têm desempenho melhor que o atual.

Além da extração dos espectrogramas, os mesmos foram ordenados como sugerido pelo trabalho [3], o que pode ser visto na [figura 1](#). Esta etapa gerou um pouco de atraso no cronograma, visto que o gasto computacional para fazer a extração e manipulação das 169 músicas necessitou de mais memória RAM e armazenamento do que o existente, desta forma, foi necessário a compra de novas componentes, o que levou ao gasto do tempo de entrega e de instalação, porém, com os contratempos, a etapa foi realizada com sucesso.

6.4 Escolha e desenvolvimento do modelo de aprendizado

Como supracitado na metodologia, foi escolhido a utilização da BiLSTM devido a comprovação empírica dos trabalhos discutidos na bibliografia, porém, não houve tempo para realizar uma reprodução dos resultados dos trabalhos estudados.

Consequentemente, com o fim de utilizar a BiLSTM em um sentido prático e, aproveitando, já confeccionar uma base de código para desenvolver a proposta do atual trabalho, utilizando o trabalho [3], o qual a autora disponibilizou os códigos e indicou as bases de dados utilizadas, foi feito um ajuste para que o código da rede neural do mesmo se adequasse a empreitada deste desenvolvimento.

Embora seja um trabalho que se deleita sobre instrumentos de percussão, a metodologia de tratamento é similar, lá a rede neural visa classificar a probabilidade de determinado(os) instrumento(os) de percussão ter(em) sido tocado(os), aqui, visa classificar a probabilidade de determinada(as) nota(as) do violão ter(em) sido tocada(as), o que corrobora para a realização desse passo.

6.5 Treinamento

A etapa de treinamento do modelo foi realizada, embora tenha enfrentado desafios significativos ao longo do desenvolvimento. A complexidade do processo de coleta, pré-processamento e extração de características contribuiu para um cronograma mais extenso do que o planejado, impactando o tempo disponível para o treinamento e ajuste do modelo.

Infelizmente, devido a restrições temporais, não foi possível atingir a acurácia desejada no modelo. A limitação de tempo também afetou a estimativa de desempenho e a implementação de melhorias necessárias para alcançar resultados mais satisfatórios.

A tabela de cronograma apresenta o progresso realizado em cada fase, destacando as tarefas concluídas e aquelas que ainda estão pendentes. Apesar dos obstáculos encontrados, houve dedicação e esforço para superar os desafios, refletindo na conclusão de várias etapas do projeto.

7 Reflexão sobre o Desenvolvimento

O desenvolvimento do projeto foi uma jornada desafiadora, marcada por obstáculos inesperados que demandaram ajustes no planejamento inicial. A complexidade da criação da base de dados, o pré-processamento detalhado e as escolhas na extração de características foram elementos críticos para o sucesso do modelo, mas também contribuíram para a extensão do tempo dedicado a essas fases.

A implementação da BiLSTM, escolhida com base em trabalhos anteriores, foi bem-sucedida, mas o treinamento e a avaliação do modelo enfrentaram limitações devido à pressão temporal. A falta de tempo para ajustes finos e otimizações impactou diretamente nos resultados finais.

7.1 Perspectivas Futuras

Apesar das dificuldades enfrentadas, o trabalho desenvolvido até o momento fornece uma base sólida para futuras melhorias e iterações. Recomenda-se a continuação do projeto com foco nas seguintes áreas:

Treinamento Estendido: Realizar um treinamento mais extenso para permitir que o modelo aprenda padrões mais complexos e melhore sua acurácia.

Ajuste de Hiperparâmetros: Explorar diferentes configurações de hiperparâmetros para otimizar o desempenho do modelo.

Expansão da Base de Dados: Considerar a inclusão de mais dados de treinamento para aumentar a diversidade e representatividade da base.

Avaliação Iterativa: Realizar avaliações iterativas do modelo, ajustando conforme necessário para alcançar melhores resultados.

Essas sugestões visam aprimorar o modelo e permitir a obtenção de resultados mais robustos em trabalhos futuros. Apesar das limitações, o projeto contribuiu significativamente para o entendimento das complexidades envolvidas na transcrição automática de músicas para violão solo além de criar uma base de dados inédita de música MIDI de violão solo.

8 Ferramentas

Ferramentas utilizadas

- Anaconda (Framework para desenvolvimento)
- Spyder (IDE de desenvolvimento)
 - Bibliotecas:
 - Os (Manipulação de arquivos)
 - Numpy (Manipulação de dados)
 - Mido (Manipulação de MIDI)
 - Librosa (Biblioteca extensa para tratamento de ondas sonoras)
 - pretty_midi (Manipulação de MIDI)
 - Tpython (Visualização de resultados)
 - Keras (Pacote de rede neural)
 - Matplotlib (Visualização de resultados),
 - Pandas (Manipulação de dados).
- Audacity (Visualização dos arquivos de som tanto WAV quanto MIDI)
- ChatGPT (Estudo e resumo de parágrafos em fases iniciais)
- MuseScore (Fonte de dados principal)
- Google Drive (Comprado uma licença para armazenamento da base de dados)
- Google colab (Após perceber que não seria suficiente, a plataforma passou a ser o computador do próprio desenvolvedor do projeto)

9 Conclusão

O desenvolvimento desta pesquisa representou uma jornada desafiadora na busca por uma solução inovadora e eficaz para a transcrição musical em tempo real de violão solo. Inicialmente, a opção por uma abordagem baseada na comparação de arquivos MIDI revelou-se mais promissora do que a análise direta de partituras ou tablaturas. Ao criar uma base de dados de músicas de violão solo em MIDI provenientes do MuseScore, obtivemos uma diversidade de amostras essenciais para o treinamento e validação do modelo.

Percebemos adicionalmente as dificuldades extras associadas à criação de tablaturas. Isso implicaria levar em consideração a diferenciação de oitavas no violão. Portanto, isso abre espaço para um novo projeto de conclusão de curso. Este seria focado na classificação precisa do local no braço do violão onde cada nota é tocada.

O processo de desenvolvimento envolveu a criação da base de dados de músicas de violão solo em formato MIDI, enfrentando desafios inesperados, como a escassez desse tipo específico de base na internet. Durante a etapa de pré-processamento, foram necessários cuidados na conversão de arquivos MIDI para WAV, escolha do soundfont adequado e normalização da taxa de amostragem.

A extração de características foi otimizada com a adoção do mel-espectrograma em vez da STFT, proporcionando uma representação mais eficiente dos dados musicais. A construção das matrizes de treinamento (X e Y) envolveu a ordenação dos mel-espectrogramas, ajustes de dimensões (com considerações sobre possíveis impactos no desempenho) e a concatenação de eventos musicais para formar a matriz Y.

A escolha da arquitetura de rede neural, BiLSTM, foi fundamentada em trabalhos anteriores, com adaptações para atender às especificidades deste projeto. No entanto, o treinamento do modelo enfrentou limitações temporais, resultando em uma acurácia abaixo do desejado. A falta de tempo para ajustes finos e otimizações impactou diretamente nos resultados.

Apesar dos desafios, este trabalho estabeleceu uma base sólida para futuras melhorias. Recomenda-se a continuação do projeto com treinamento estendido, ajuste de hiperparâmetros, expansão da base de dados e avaliações iterativas. A otimização do código também se apresenta como uma área de foco para garantir eficiência computacional.

Em última análise, este estudo contribuiu significativamente para o avanço do entendimento sobre a transcrição automática de músicas para violão solo, além de criar uma base de dados única nesse contexto. As perspectivas futuras visam refinar e aprimorar a abordagem proposta, pavimentando o caminho para soluções mais robustas e eficazes na transcrição musical em tempo real de violão solo.

10 Cronograma

Tarefa	Descrição	Semanas de desenvolvimento	Estado
Coleta de dados	Confecção da base de dados (Preparação das músicas e arquivos MIDI respectivos)	2	Concluído (Com acréscimo)
Pré-processamento de dados	Remoção de ruídos, normalização, etc...	2	Concluído
Extração de características	Escolha e extração das features	2	Concluído (Com acréscimo)
Escolha e desenvolvimento do modelo de aprendizado	Implementação da BiLSTM	1	Concluído (Com acréscimo)
Treinamento	Realização do treinamento do algoritmo (Conjunto de dados de treino, estimativa do erro)	3	Parcialmente concluído (Com acréscimo)
Avaliação e ajuste	Realização dos testes de desempenho (Conjunto de dados de avaliação)	3	Não concluído
Teste e avaliação	Realização da avaliação (Conjunto de dados de validação)	3	Não concluído

É importante ressaltar que essas etapas são uma visão geral e simplificada do processo de desenvolvimento de um algoritmo de transcrição musical. A complexidade e as especificidades cresceram significativamente de acordo com o desenvolvimento, inclusive por ser uma área não explorada e sem base. Além do mais, as etapas adjacentes como a utilização de detector de fontes e geração da partitura a partir do MIDI não entraram no cronograma.

11 Referências bibliográficas

- [1] Guitars AI. Basics of Automatic Music Transcription using Python. 11 de dez. De 2020. Disponível em: <<https://www.youtube.com/watch?v=evYU6nGXlY4>>;
- [2] Jeong, Dasaem. Real-time Automatic Piano Music Transcription System. ISMIT MTL, 2020. Disponível em: <https://program.ismir2020.net/lbd_444.html>. Acesso em: 26, fevereiro e 2023.
- [3] Hu, Yue. Practical Introduction to Automation Music Transcription. Towardsdatascience, 2020. Disponível em: <<https://towardsdatascience.com/practical-introduction-to-automation-music-transcription-3ad8ad40eab6>>. Acesso em: 26, fevereiro e 2023.
- [4] T. Kwon, D. Jeong, and J. Nam, “Polyphonic piano transcription using autoregressive multi-state note model,” in Proc. of the 21st Int. Society for Music Information Retrieval Conf. (ISMIR), 2020.
- [5] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, “Onsets and frames: Dual-objective piano transcription,” in Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR), October 2018, pp. 50–57.
- [6] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in Proc. of the 7th International Conference on Learning Representations (ICLR), 2019.
- [7] A. Paszke et al., “Pytorch: An imperative style, high-performance deep learning library,” Adv. Neural Inf. Process. Syst., vol. 32, 2019.
- [8] J. Engel et al., “DDSP: Differentiable digital signal processing,” in Proc. Int. Conf. Learn. Representations, 2019.
- [9] H. Cuesta, B. McFee, and E. Gómez, “Multiple f0 estimation in vocal ensembles using convolutional neural networks,” in Proc. Int. Soc. Music Inf. Retrieval Conf., 2020, pp. 302–309.
- [10] R. Schramm et al., “Multi-pitch detection and voice assignment for a cappella recordings of multiple singers,” in Proc. Int. Soc. Music Inf. Retrieval Conf., 2017, pp. 552–559.
- [11] A. McLeod and M. Steedman, “HMM-based voice separation of MIDI performance,” J. New Music Res., vol. 45, no. 1, pp. 17–26, 2016.
- [12] G. Fant, Acoustic Theory of Speech Production. Berlin, Germany: Walter de Gruyter, 1970.
- [13] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Representations, 2015.
- [14] S. Ewert and M. Müller, “Using score-informed constraints for NMFbased source separation,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2012, pp. 129–132.
- [15] J.-L. Durrieu, B. David, and G. Richard, “A musically motivated mid-level representation for pitch estimation and musical audio source separation,” IEEE J. Sel. Topics Signal Process., vol. 5, no. 6, pp. 1180–1191, Oct. 2011.
- [16] D. H. Klatt and L. C. Klatt, “Analysis, synthesis, and perception of voice quality variations among female and male talkers,” J. Acoust. Soc. Amer., vol. 87, no. 2, pp. 820–857, 1990.

- [17] D. Petermann, P. Chandna, H. Cuesta, J. Bonada, and E. Gomez, "Deep learning based source separation applied to choir ensembles," in Proc. Int. Soc. Music Inf. Retrieval Conf., 2020, pp. 733–739.
- [18] G. Meseguer-Brocal and G. Peeters, "Conditioned-U-Net: Introducing a control mechanism in the U-Net for multiple source separations," in Proc. Int. Soc. Music Inf. Retrieval Conf., 2019, pp. 159–165.
- [19] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR—half-baked or well done?," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2019, pp. 626–630.
- [20] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in Proc. Int. Conf. Latent Variable Anal. Signal Separation, 2018, pp. 293–305.
- [21] D. S. Moore and S. Kirkland, *The Basic Practice of Statistics*, vol. 2. New York, NY, USA: WH Freeman, 2007.
- [22] H. Levene, "Robust tests for equality of variances," *Contributions Probability Statist. Essays Honor Harold Hotelling*, pp. 279–292, 1961.
- [23] B. L. Welch, "The generalization of 'STUDENT'S' problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.
- [24] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., 2018, pp. 161–165.
- [25] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: <https://doi.org/10.5281/zenodo.1117372>.
- [26] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 3, pp. 1066–1074, Mar. 2007.
- [27] SCHULZE-FORSTER, K.; RICHARD, G.; KELLEY, L.; DOIRE, C. S. J.; BADEAU, R; *Unsupervised Music Source Separation Using Differentiable Parametric Source Models*; 2023.

12 Anexo

12.1 MatrizX

Foram removidas as variáveis de diretório

```

8  import os
9  import mido
10 import numpy as np
11 from mido import MidiFile
12 import pretty_midi
13
14 frame_length = 0.05
15
16 # Lista dos nomes dos arquivos WAV na pasta de origem
17 arquivosWav = sorted([arquivo for arquivo in os.listdir(wavFilesPath) if arquivo.endswith('.wav')])
18
19
20 # Define a taxa de amostragem padrão para todos os arquivos
21 sr = 44100 # Altere para a taxa de amostragem desejada
22
23 # Segmentação em janelas com sobreposição
24 durationWindow = 0.05
25 #-----
26
27 featuresList = []
28 for musicName in arquivosWav:
29     # Carregue o arquivo de áudio usando o pydub e defina a taxa de amostragem para a taxa padrão
30     audio, sr = librosa.load(wavFilesPath + musicName, sr=sr, mono=True)
31
32     musicDuration = librosa.get_duration(y=audio, sr=sr)
33
34     # Calcular o número total de vetores desejado
35     desired_num_frames = int(np.round(musicDuration / durationWindow)) * 4
36
37     # Calcular hop_length com base no número desejado de vetores
38     hop_length = int(sr * musicDuration / desired_num_frames)
39
40
41     # Frame length
42     n_fft = hop_length * 2 # Pode ser ajustado conforme necessário
43
44     # Normalização
45     normalized_audio = librosa.util.normalize(audio)
46
47     melspectrogram = np.abs(librosa.feature.melspectrogram(y=normalized_audio, n_fft=n_fft, hop_length=hop_length))
48
49     # Padronização
50     mean = np.mean(melspectrogram)
51     std = np.std(melspectrogram)
52     standardized_melspectrogram = (melspectrogram - mean) / (std + 1e-5)
53
54
55     # Armazenar as características em um dicionário
56     features = {
57         'musicName': musicName,
58         'standardized_melspectrogram': standardized_melspectrogram[:, 0:desired_num_frames],
59     }
60
61     featuresList.append(features)
62
63 #-----
64
65 # Inicialize a matriz vazia com o dobro do tamanho máximo para acomodar zeros
66 xMatrix = None
67 segmentLength = librosa.time_to_frames(durationWindow)

```



```

32         notes_per_musics.append(sorted(note_on_messages, key=lambda x: x[0]))
33         note_on_messages = []
34
35     return classifi_notes(notes_per_musics, musicPath, frame_length)
36
37
38 def classifi_notes(notes_per_musics, musicPath, frame_length = 0.05):
39     # Obtenha a duração total em segundos
40     midi_data = pretty_midi.PrettyMIDI(musicPath)
41     musicDuration = midi_data.get_end_time()
42
43     # Calcular o número total de vetores desejado
44     desired_num_frames = int(np.round(musicDuration / frame_length))
45
46     # Instanciar uma lista com N vetores de 45 elementos de conteúdo zero
47     discretizedMusic = [[0] * 45 for _ in range(desired_num_frames)]
48
49     for (time, nota) in notes_per_musics[0]:
50         occurrenceAddress = int(time / frame_length)
51         classificartionColumn = nota - 39
52         discretizedMusic[occurrenceAddress][0] = time
53         discretizedMusic[occurrenceAddress][classificartionColumn] = 1
54
55     return discretizedMusic[:,0:desired_num_frames]
56
57
58 def process_midis(arquivosMidi):
59     result_matrix = []
60
61     for musicName in arquivosMidi:
62         if musicName.endswith('.mid'):
63             musicPath = os.path.join(midFilesPath, musicName)
64
65             result_matrix.extend(process_midi_file(musicPath))
66
67     return result_matrix
68
69
70 arquivosMidi = sorted([arquivo for arquivo in os.listdir(midFilesPath) if arquivo.endswith('.mid')])
71 yMatrix = process_midis(arquivosMidi)

```