Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Ricardo Correia Nascimento dos Santos

Uso de *Deep Learning* na extração automática de *features* de áudio: um experimento com solfejos

São Paulo
2020

Ricardo Correia Nascimento dos Santos

Uso de *Deep Learning* na extração automática de *features* de áudio: um experimento com solfejos

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação.

Data da aprovação 31 / 08 / 20

Prof. Dr. Marcelo Novaes de Rezende (Orientador)
Mestrado Engenharia de Computação

Membros da Banca Examinadora:

Prof. Dr. Marcelo Novaes de Rezende (Orientador)
Mestrado Engenharia de Computação

Prof. Dr. Eduardo Takeo Ueda (Membro)
Mestrado Engenharia de Computação

Prof. Dr. Antonio Carlos Tonini (Membro)
ESEG - Escola Superior de Engenharia e Gestão

Ricardo Correia Nascimento dos Santos

Uso de *Deep Learning* na extração automática de *features* de áudio: um experimento com solfejos

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação.

Área de Concentração: Engenharia de Software

Orientador: Prof. Dr. Marcelo Novaes de Rezende

São Paulo
Junho/2020

Ficha Catalográfica
Elaborada pelo Departamento de Acervo e Informação Tecnológica – DAIT
do Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT

S626u **Santos, Ricardo Correia Nascimento dos**
Uso de Deep Learning na extração automática de features de áudio: um experimento com solfejos . / Ricardo Correia Nascimento dos Santos. São Paulo, 2020.
162p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Marcelo Novaes de Rezende

1. Rede neural convolucional 2. Rede neural recorrente 3. Processamento de sinais de áudio 4. Solfejo 5. Recuperação de informação musical 6. Tese I. Rezende, Marcelo Novaes de, orient. II. IPT. Coordenadoria de Ensino Tecnológico III. Título

2020-36 CDU 004.41(043)

Bibliotecária responsável: Maria Darci Cornellas Narciso – CRB 8/3569

DEDICATÓRIA

Dedico este trabalho a minha esposa Cibele, a minha pequena filha Beatriz e meu pequenino filho Nicolas pela paciência e compreensão durante meus longos períodos de isolamento para elaboração da Dissertação.

Aos meus pais, Bernardo e Ilda pelo constante apoio e incentivo.

A Deus, pois Ele é fiel.

AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Marcelo Novaes de Rezende por aceitar este desafio e conduzir o trabalho de revisão de texto e demais correções realizadas durante a preparação desta Dissertação, além dos ensinamentos transmitidos na disciplina de *Aprendizagem de Máquina* que ajudaram a vislumbrar uma nova perspectiva profissional.

Aos professores Dr. Eduardo Takeo Ueda e Dr. Antonio Carlos Tonini pelas correções e demais considerações realizadas nas etapas de Qualificação e Defesa desta Dissertação.

Aos professores do Curso de Mestrado em Engenharia da Computação do IPT.

À Coordenação do Curso e Equipe da Secretaria por toda ajuda e atenção dispensada.

Aos colegas de Mestrado e do trabalho, amigos e familiares que de alguma forma contribuíram para o sucesso desta jornada.

RESUMO

A voz humana pode ser usada como parte da entrada de dados em aplicativos de análise e síntese de sons, sistemas de busca de música e em tarefas de composição musical. Portanto, existe uma necessidade social de se tratar algoritmicamente a voz humana para permitir o processamento computacional deste tipo de sinal de áudio. Nesta dissertação, a proposta é aplicar técnicas de *Deep Learning* para extrair automaticamente as *features* presentes na voz humana que são mais relevantes para a identificação de notas musicais e melodias contidas em exercícios de solfejo. No experimento realizado, foi desenvolvido um protótipo de *software* capaz de auxiliar no processo de sintetização de sons de instrumentos musicais a partir da voz humana cantada. Para isto, foi considerado que uma sequência de sons pode ser analisada como uma série temporal e que as Redes Neurais Convolucionais (CNN) e as Redes Neurais Recorrentes (RNN) do tipo *Long-Short Term Memory* (LSTM) possuem propriedades que sugerem a possibilidade de seu uso no contexto musical. A metodologia utilizada no experimento consistiu em realizar o aprendizado supervisionado das redes CNN e LSTM por meio da técnica conhecida como *transfer learning*. A rede CNN utilizada foi retreinada com o *dataset MedleyDB* (desconsiderando qualquer aprendizado anterior) para realizar a extração de melodias à partir de exercícios de solfejo, atuando como *salience function*. Este modelo foi combinado com três redes LSTM distintas treinadas com o *dataset VocalSet* para classificar as notas musicais, aprender o timbre de um instrumento musical (piano) e sintetizar o som das notas musicais identificadas no solfejo em formato WAVE. Como resultado do experimento, esta abordagem de uso combinado de redes CNN e LSTM mostrou-se eficiente para realizar a transcrição da melodia detectada nos exercícios de solfejo (realizados individualmente) para um som com timbre de instrumento musical real. Foi alcançada uma média de $71,90\%$ de acertos para classificação multiclasse (128 categorias) *note-by-note* nos exercícios de canto da *escala musical* (entre *C4* a *D5*) e canto de *arpeggios*, ambos os exercícios, selecionados de um subconjunto de solfejos do *dataset VocalSet*. O mesmo experimento realizado para classificação multiclasse (também com 128 categorias) *frame-by-frame* resultou em $57,24\%$ de acertos nos mesmos exercícios de solfejo citados, próximo dos $60,00\%$ apresentados na literatura. Estes resultados foram comparados com estudos que representam o *estado da arte* em tarefas de *transcrição de melodia* de *voz humana cantada*, sendo que o modelo *note-by-note* foi comparado à acurácia geral (*overall acuracy*) e o modelo *frame-by-frame* considerou uma métrica baseada em *f-measure*.

Palavras-chave: Extração de Descritores de Áudio; Recuperação de Informação Musical; Transcrição Melódica; Solfejo; LSTM; CNN.

USE OF DEEP LEARNING IN THE AUTOMATIC AUDIO FEATURE EXTRACTION: AN EXPERIMENT WITH SOLFEGGIO

The human voice can be used as part of data input in synthesis applications and sound analysis, music search systems and in musical composition tasks. Therefore, there is a social need to treat the human voice algorithmically to allow the computational processing of this type of audio signal. In this dissertation, the proposal is to apply Deep Learning techniques to automatically extract the features present in the human voice that are most relevant for the identification of notes music and melodies contained in *solfeggio* exercises. In the experiment, it was developed a software prototype capable of assisting in the process of synthesizing sounds from singing human voice to real musical instruments timbre. For this subject, it was considered that a sequence of sounds can be analyzed as a time series and that the *Convolutional Neural Networks* (CNN) and *Recurrent Neural Networks* (RNN) of type *Long-Short Term Memory* (LSTM) have properties that suggest the possibility of its use in the musical context. The methodology used in the experiment consisted of conducting *supervised learning* for the CNN and LSTM networks through the technique known as *transfer learning*. The CNN network used was retrained from the scratch with the *MedleyDB* dataset to extract melodies from *solfeggio*, acting as *salience function*. This model was combined with three LSTM networks trained with the *VocalSet* dataset to classify musical notes, learn the timbre of a musical instrument (piano) and synthesize the sound of the identified musical notes in *solfeggio* to the WAVE format. As a result of the experiment, the approach of combined use of CNN with LSTM networks proved to be efficient for transcribing of the melody detected in the *solfeggio* exercises (performed individually) to a sound with real musical instrument timbre. An average of 71.90% hits were achieved for *note-by-note* muticlass classification (with 128 categories) in singing exercises on the *musical scale* (between C4 to D5) and *arpeggio* singing, both exercises, selected from a subset of *solfeggio* of the *VocalSet* dataset. The same experiment carried out for *frame-by-frame* multiclass classification (also with 128 categories) resulted in 57.24% of correct answers in the same aforementioned *solfeggio* exercises, close to the 60.00% presented in the literature. These results were compared with studies that represent the state of the art in tasks of *transcribing melody* of *human sung voice*, the model emph note-by-note was compared to the *overall accuracy* and the model *frame-by-frame* considered a metric based on *f-measure*.

Keywords: Audio Features Extraction; Musical Information Retrieval; Melodic Transcription; Solfeggio; LSTM; CNN.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ANN	*Artificial Neural Networks*
ANSI	*American National Standards Institute*
AS	*Audio Segmentation*
ASR	*Automatic Speech Recognition*
BPM	*Battiti per minuto*
CIFAR	*Canadian Institute for Advanced Research*
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
CQT	*Constant-Q Transform*
DFT	*Discrete Fourier Transform*
DL	*Deep Learning*
DNN	*Deep Neural Network*
DTW	*Dynamic Time Warping*
ESR	*Environmental Sound Recoginition*
FLAC	*Free Lossless Audio Codec*
FP	*Fingerprinting*
GPU	*Graphic Processing Unit*
HMM	*Hidden Markov Model*
IEEE	*Institute of Electrical and Electronics Engineers*
IPT	Instituto de Pesquisas Tecnológicas do Estado de São Paulo
KNN	*K-Nearest Neighbour*
LDA	*Linear discriminant analysis*
LSTM	*Long Short-Term Memory*
LTU	*Linear Threshold Unit*
MAE	*Mean Absolute Error*
MIDI	*Musical Instrumental Digital Interface*
MIR	*Music Information Retrieval*
ML	*Machine Learning*
MLP	*Multilayer Peceptron*
MPEG	*Moving Picture Experts Group*
MP3	*MPEG 1 Layer-3*
MSE	*Mean Squared Error*
PCM	*Pulse-code modulation*
PICO	*Population, Intervention, Control e Outcome*
RELU	*Rectified Linear Units*
RMSE	*Root Mean Absolute Error*
RNN	*Recurrent Neural Network*
SGD	*Stochastict Gradient Descent*
STFT	*Short Time Fourier Transform*
SVM	*Support Vector Machine*
VAD	*Voice Activity Detection*
VAMP	*Vamp audio analysis plugin system*
VAR	*Variety*
WAVE	*WAVEform audio format*
WAV	Extensão de arquivos *WAVE*

SUMÁRIO

1 INTRODUÇÃO

Considerando o surgimento de computadores modernos, com maior capacidade de processamento e manipulação de dados durante a década de 1990, pode-se destacar o desenvolvimento de áreas de pesquisas ligadas a inteligência artificial, tais como o aprendizado de máquina e sua relação interdisciplinar com a computação musical.

Assim, esta seção introdutória expõe uma visão abrangente sobre a evolução destas tecnologias e áreas de pesquisas afins para estabelecer as conexões existentes entre os diversos tópicos a serem apresentados nesta dissertação.

1.1 Contexto e Motivação

O uso de tecnologias computacionais em música tornou-se cada vez mais comum a partir da década de 1990 e contempla diversos cenários que vão desde a realização de tarefas como composição musical e edição profissional de música até o ensino de Educação Musical na Rede de Ensino Básica (REPSOLD, 2018). Esta tendência de aumento no uso de tecnologias musicais se acentuou com a popularização da internet, surgimento de plataformas de *streaming* de música, Educação a Distância (*online*), modernização de computadores, *tablets* e celulares (MELO FILHO, 2017).

Com a adoção de novas tecnologias musicais, foi necessário aprimorar a forma de representação digital de som para facilitar tarefas de indexação, busca e armazenamento de arquivos de áudio e organização dos metadados com informações básicas tais como título da música ou faixa de áudio, artista, álbum musical e ano de lançamento.

Surgiram, então, diferentes tipos de formato digital para música. Assim, as representações digitais do som puderam ser armazenadas em aquivos como, por exemplo, *WAV, MP3 e MIDI* (VAIL, 2014).

Todavia, a representação digital do som em formato bruto, à *priori*, não fornece informações ou características (*features*) sobre o estilo ou gênero musical, ritmo, harmonia, melodia e outros detalhes que porventura estejam presentes neste som de maneira a permitir seu processamento por algoritmos de *Machine Learning*.

Por isso, a área de pesquisa conhecida como Recuperação de Informação Musical (MIR) combinada com *Machine Learning* pretende explorar técnicas e métodos para extrair estas informações ou características (*features*) de maneira que possam ser utilizadas

em algoritmos de *Machine Learning* e, assim, resolver problemas relacionados a transcrição de áudio, reconhecimento de gênero musical, recomendação musical, separação de vozes, reconhecimento de instrumentos musicais, identificação de notas musicais e acordes, busca por voz cantada (*query by humming*), detecção de ritmo, timbre, harmonia e melodia, similaridade melódica, impressão digital de áudio para identificação exclusiva de fragmentos de música, análise e sintetização de sons musicais e tarefas similares (MIR, 2019).

Segundo Géron (2017, p. 25-26), as atividades de seleção e extração manual de *features* em projetos de *Machine Learning* fazem parte de um processo chamado de *feature engineering*. Géron (2017, p. 25-26) ainda afirma que estas atividades são críticas para o sucesso do projeto e as define como:

- Seleção de *Features* (ou *Feature Selection*): é a seleção de quais *features* são mais importantes dentro de um conjunto de *features* disponíveis para o treinamento de um algoritmo de *Machine Learning*;
- Extração de *Features* (ou *Feature Extraction*): são combinações de diferentes *features* para compor uma nova *feature* mais adequada ou útil para o algoritmo de *Machine Learning* em questão.

Em sua tese de doutorado, Schramm (2015a) aborda técnicas de MIR e *Machine Learning* para seleção e extração manual de *features* de áudio (*feature engineering*) que permitam realizar uma classificação nota-a-nota da melodia cantada em práticas de solfejos. Em seu estudo, Schramm (2015a) também realiza capturas de imagens de vídeo durante a execução do solfejo para interpretar o movimento rítmico das mãos, que servem para marcar o compasso da música. Assim, Schramm (2015a) faz uma proposta de um sistema audiovisual capaz de classificar automaticamente se as notas cantadas por um praticante de solfejo estão corretas em relação a um conjunto de notas musicais de referência e dentro do ritmo e compasso musical esperado.

Uma importante *feature* de áudio para a classificação de notas musicais é denominada *pitch* e através desta é possível correlacionar a altura de uma nota musical com a percepção humana do som que representa aquela nota musical utilizando a grandeza física definida como *frequência do som* (expressa em *Hertz*).

Na proposta de Schramm (2015a), é utilizado o algoritmo pYIN para extrair a frequência fundamental do som e determinar as notas musicais com a ajuda de um classificador baseado em *Hidden Markov Models (HMM)* para realizar a transcrição melódica do solfejo. O algoritmo pYIN é uma versão modificada do original YIN e possui melhor acurácia durante a tarefa de *Pitch Tracking* que é a tarefa de rastrear uma sequência de *pitches* enquanto um conjunto de dados de áudio é observado ou processado (SIMSEKLI, 2010).

A melodia obtida à partir do solfejo junto com o movimento rítmico das mãos são comparados com a partitura alvo do exercício escolhido através de uma versão modificada do algoritmo *Dynamic Time Warping (DTW)*, que é usado como forma de sincronização temporal (SCHRAMM, 2015a) entre a performance do cantor e a partitura para avaliar a qualidade de execução do solfejo (SCHRAMM, 2015a).

Ao final do processo de extração de melodia com as técnicas demonstradas em Schramm (2015a), é gerado um som em formato MIDI (VAIL, 2014) com as notas musicais identificadas durante o exercício de solfejo e indicação visual dos acertos e erros aproximados da pessoa que cantou a melodia original.

Por sua vez, na tese de doutorado de Salamon (2013b), o autor utiliza os conceitos e técnicas de MIR para a extração automática de melodias. No trabalho dele, são aplicadas diferentes abordagens para realizar a autocorrelação dos sinais de áudio juntamente com o uso de filtros de frequências para extrair *features* baseadas na frequência fundamental do som tais como *f0*, *spectral peaks* e *pitch* (altura da nota musical percebida pelo ouvido humano) e obter a melodia executada diretamente do áudio bruto. Uma das principais contribuições acadêmicas do trabalho de Salamon (2013b) é o método de extração de melodias através da proposta de uma nova função denominada *salience function* (SALAMON, 2013b, p. 170) que é responsável por extrair e representar a altura das notas musicais (*pitch*) mais predominantes em relação ao cantor principal de uma música ou instrumentos musicais tocando a melodia principal daquela peça musical.

Para a etapa de classificação das notas musicais obtidas, Salamon (2013b, p. 140) realizou testes com diferentes classificadores usuais em *Machine Learning* como *Support Vector Machine*, Árvores de Decisão, Regressão Logística, *K-Nearest Neighbour* e Redes Bayesianas já implementados no *framework* de *Machine Learning* conhecido como *Weka* (WAIKATO, 2019) utilizando os parâmetros originais.

Outra contribuição importante do trabalho de Salamon (2013b) do ponto de vista técnico foi a implementação de uma ferramenta de *software* apelidada de *MELODIA*

(SALAMON, 2013b, p. 238). *MELODIA* (SALAMON, 2013a) é um *plugin* de *software* que permite adicionar funcionalidades de extração automática de melodia em *softwares* musicais compatíveis com o padrão de *plugins VAMP* (MARY, 2012).

Ambos os trabalhos, de Schramm (2015a) e Salamon (2013b), empregam alguma forma de *feature engineering* seguido de algoritmos usuais de *Machine Learning*. Em tarefas similares de classificação de notas musicais, é comum aplicar técnicas de MIR para realizar a seleção e extração manual de um conjunto de *features* a partir de trechos de áudio bruto ou preprocessado e, em seguida, aplicar algum algoritmo de *Machine Learning*. Assim, diversos algoritmos de aprendizado supervisionado podem ser utilizados na tarefa de classificação, tais como *Support Vector Machine (SVM)* discutido no artigo de Poliner e Ellis (2005), *K-Nearest Neighbor (KNN)*, *Linear Discriminant Analysis (LDA)* e *Adaboost* (SENAC et al., 2017).

Mas com o avanço das técnicas de *Deep Learning* em conjunto com a evolução das Placas de Aceleração Gráfica (GPU) que podem ser utilizadas para computação geral e acelerar o treinamento das Redes Neurais Profundas via *hardware* (LECUN; BENGIO; HINTON, 2015), cada vez mais estão sendo empregadas Redes Neurais Convolucionais como mostrado em Senac et al. (2017) e Costa, Oliveira e Silla Jr (2017) para resolver tarefas envolvendo música como, por exemplo, de classificação de gênero musical, ou ainda para tarefas de extração automática de *features* de áudio para transcrição de melodia conforme Bittner et al. (2017).

Tendo em vista a predominância do tema ao longo dos anos e a evolução tecnológica alcançada na área de *Machine Learning* e *Deep Learning*, esta dissertação pretende investigar arquiteturas de Redes Neurais Convolucionais (CNN) para realizar a extração automática de *features* de áudio que permitam a classificação de notas musicais, reduzindo a necessidade de realizar *feature engineering*. Após isso, explorar o uso de Redes Neurais Recorrentes do tipo *Long Short-Term Memory (LSTM)* para sintetização de áudio e predição de sequências de som similar a um instrumento musical real. Estes tipos específicos de Redes Neurais Profundas tem sido apresentadas na literatura como alternativas viáveis para lidar com tarefas de processamento de áudio e música que usam técnicas de *Machine Learning*.

Portanto, é relevante analisar a viabilidade de uso destas Redes Neurais Profundas, CNN e LSTM, combinadas de forma similar à apresentada em Brownlee (2017, cap. 8) para as tarefas de classificação de notas musicais e transformação de timbres de solfejo

em sons de Instrumentos Musicais com o uso de técnicas de *Machine Learning* e *Deep Learning*.

1.2 Problema

Em matéria de música, a voz humana também é considerada um instrumento musical capaz de reproduzir sons e melodias musicais com variada expressividade sonora e artística (STOWELL, 2010). Assim, destaca-se a questão da altura de uma nota musical e timbre correspondente que podem ser determinadas por *features* como *pitch*, *spectral envelope* e frequência fundamental do som (*f0*) (SALAMON, 2013b). Estas são importantes *features* para determinar as notas musicais referentes a cada trecho de uma melodia cantada por uma voz humana, por exemplo, como no caso dos solfejos (SCHRAMM, 2015a).

De acordo com Mitrović, Zeppelzauer e Breiteneder (2010), um dos principais desafios encontrados no desenvolvimento de *softwares* de recuperação de informações de áudio é a determinação de quais características (*features*) ou descritores de áudio presentes nos sinais sonoros são mais adequados para determinadas tarefas. Embora já exista uma ampla gama de técnicas e métodos publicados na literatura corrente sobre este tema, não é uma atividade trivial consolidar todo este conhecimento em uma taxonomia comum que auxilie na compreensão e seleção destas *features* de áudio por meio da atividade conhecida como "*feature engineering*" (GÉRON, 2017).

Um ponto de atenção em relação à atividade de *feature engineering* realizada por um ser humano está relacionada à dificuldade de selecionar manualmente as *features* de áudio necessárias para uma dada tarefa de classificação ou regressão em áudio. Por exemplo, podem ocorrer divergências na percepção de uma dada característica musical, dificuldade em lidar com a complexidade relacionada a combinações de diferentes *features* de áudio e hiperparâmetros, ou ainda, podem ocorrer erros na anotação de notas musicais de um trecho de som que são ouvidas pelos especialistas (por exemplo, músicos ou *designers* de som) responsáveis em criar os rótulos (ou *labels*, em inglês) para a tarefa de classificação de áudio em algoritmos supervisionados (BITTNER et al., 2014).

Para Hipke et al. (2014), o uso de sistemas interativos (em tempo real) no treinamento de algoritmos de *Machine Learning* pode ajudar na escolha dos hiperparâmetros e combinações de diferentes *features* de áudio durante a criação de classificadores de áudio e apresenta um sistema capaz de capturar a vocalização de sons percussivos e classificá-los

corretamente de acordo com os timbres típicos de uma bateria, tais como aqueles emitidos pelo bumbo, chimbal, prato de ataque e caixas de tons da bateria.

Neste sentido, Stowell (2010) destaca outro problema importante no processamento de áudio musical que diz respeito à extração de *features* de áudio que identifiquem corretamente o timbre de uma voz humana. Stowell (2010) apresenta uma proposta de sistema de *Machine Learning* para controlar um sintetizador de áudio por meio da análise do timbre da voz humana, possibilitando remapear o timbre original associado à nota cantada por uma voz humana por um timbre de um instrumento musical específico como, por exemplo, percussão, bateria, piano ou guitarra. Para isto, propõe o uso de algoritmos de *Machine Learning* baseados em Árvores de Regressão para rotular os dados automaticamente.

Segundo Schramm (2015a) e Salamon (2013b), também existem alguns problemas que surgem durante a análise das notas cantadas ou em vocalizações que estão relacionados à correspondência entre a altura da nota sonora emitida pela voz humana, sua duração (medida em segundos) e a altura da nota musical de referência mais próxima, considerando uma escala sonora de 12 notas musicais (usada frequentemente em músicas ocidentais).

As Redes Neurais são importantes modelos matemáticos (paramétricos) usados em *Machine Learning* e *Deep Learning* para o aprendizado de tarefas complexas, por exemplo, como a visão computacional e o processamento automático da fala humana por um computador. Estes modelos matemáticos costumam ser implementados em *softwares* que representam cada unidade computacional da Rede Neural como nós interligados (ou conectados entre si) que possuem funções matemáticas associadas a cada nó e pesos (ou parâmetros) vinculados a cada conexão, as quais têm seus pesos ajustados à partir de amostras de dados. À medida que estes pesos são ajustados, ocorre o aprendizado de máquina propriamente dito e estas Redes Neurais podem ser utilizadas na execução das tarefas citadas.

A principal diferença entre as Redes Neurais Artificiais clássicas criadas no início do *Machine Learning* e as Redes Neurais Profundas usadas no *Deep Learning* são as quantidades de nós e camadas presentes na arquitetura de cada uma. Geralmente, uma Rede Neural é considerada profunda quando possuem mais que três camadas (GÉRON, 2017).

À partir de então, surgiram diferentes tipos de arquiteturas e famílias de Redes Neurais Artificiais como, por exemplo, as Redes Neurais Convolucionais e as Redes Neurais Recorrentes. A primeira foi desenvolvida com a intenção de resolver problemas na área

da Visão Computacional, enquanto a segunda é voltada para problemas de predição de sequências e séries temporais. Ambas são estudadas na subseção 2.4.1 e na subseção 2.4.2, respectivamente.

O Quadro 1 lista vários problemas identificados nesta seção sob a forma de perguntas para que se possa evidenciar a problemática envolvida no tema de seleção e extração de *features* de áudio em músicas, ou seja, apresentando alguns dos principais problemas relacionados às questões de pesquisa desta Dissertação.

Quadro 1 – Problemas na extração de *features* de áudio em músicas

Problemas	**Descrição**
P1. Seleção de *Features* de Áudio	Quais *features* de áudio são mais relevantes para a tarefa de classificação de notas musicais?
P2. Extração de *Features* de Áudio	Quais *features* de áudio podem ser combinadas para formar uma nova *feature* de áudio mais adequada a uma tarefa específica em MIR?
P3. Dificuldades com *Feature Engineering*	Como reduzir a necessidade de intervenção manual na obtenção das *features* de áudio e evitar os erros humanos?
P4. Hiperparâmetros	Como reduzir a quantidade de hiperparâmetros e simplificar os ajustes necessários?
P5. Variedades de Timbres	Como lidar com diferentes timbres da voz humana e instrumentos musicais?
P6. Classificação de Notas Musicais	Como classificar corretamente uma nota musical em exercícios de Solfejo?
P7. Extração de Melodia e *Pitch Tracking*	Como capturar a melodia principal automaticamente em exercícios de Solfejo?

Fonte: Elaborado pelo autor.

Portanto, busca-se compreender nesta dissertação se técnicas mais avançadas de *Machine Learning* e *Deep Learning* baseadas em Redes Neurais Artificiais podem ser utilizadas na solução total ou parcial dos problemas de extração de *features* de áudio em músicas e responder as seguintes questões de pesquisa:

- Quais arquiteturas de Redes Neurais Convolucionais são mais eficazes para extrair as *features* de música automaticamente e permitir a classificação das notas musicais em exercícios de solfejo, partindo do áudio bruto?
- Como representar um sinal de áudio utilizando Redes Neurais Recorrentes para remapear ou associar o timbre original de uma voz humana cantada em exercícios de solfejo para o timbre característico de algum instrumento musical real?

Estas questões de pesquisa são versões refinadas da questão de pesquisa original que foi usada durante a Revisão Sistemática da Literatura descrita no Apêndice A, na seção A.1.

O artigo apresentado por Rigaud e Radenen (2016) destaca possibilidades de uso de Redes Neurais Profundas para a transcrição de melodias em *voz humana cantada* e será discutido brevemente na subseção 2.8.5, mas com base nos estudos levantados na literatura existente, não foram encontrados experimentos específicos que apliquem Redes Neurais Convolucionais e Redes Neurais Recorrentes de forma combinada (BROWNLEE, 2017, cap. 8) para resolver o problema do reconhecimento de notas e melodias em exercícios de solfejo. Isto não significa que esta abordagem seja inédita, pois a inexistência de resultados deste tipo de experimento durante a execução da revisão sistemática pode acontecer caso os critérios de busca utilizados não tenham correspondentes diretos nas bases científicas como *IEEE Xplore*, *ACM Library* e outras, embora esta pesquisa tenha sido realizada com uma ampla gama de palavras-chaves como pode ser verificado no Apêndice A sobre a Revisão Sistemática da Literatura, na subseção A.1.1.

1.3 Objetivo

O objetivo desta dissertação é propor uma arquitetura híbrida de Rede Neural Convolucional combinada com uma Rede Neural Recorrente que permita a transcrição de trechos de áudio capturados durante a execução de um solfejo diretamente para uma melodia com timbre equivalente àquele executado por um instrumento musical real, por exemplo, como um piano ou uma guitarra específica.

As duas principais hipóteses consideradas neste estudo são:

- O uso de Redes Neurais Convolucionais pode obter resultados satisfatórios na extração automática de *features* de áudio, reduzindo a necessidade de *feature engineering*.
- É possível utilizar Redes Neurais Recorrentes para prever ou associar sequências de notas musicais à partir de exercícios de solfejo.

1.4 Contribuições

A principal contribuição desta dissertação está na abordagem de utilização dos modelos de Redes Neurais Convolucionais propostos em Bittner et al. (2017) para executar

tarefas de classificação de notas musicais e transcrição melódica com o uso combinado de Redes Neurais Recorrentes do tipo LSTM, conforme sugerida em Brownlee (2017, cap. 8) e Hawthorne et al. (2018) com uma nova arquiteturas otimizada (ou seja, usa o mínimo necessário de quantidade de camadas e neurônios) para o caso específico do solfejo.

Outras contribuições desta dissertação são:

- Apresentação de uma Revisão Sistemática específica para tarefas de extração de *features* de áudio usadas em algoritmos de *Machine Learning* e *Deep Learning*.
- Criação de um protótipo funcional baseado na proposta de experimento desta dissertação que utiliza *Deep Learning* para transcrição melódica de solfejos executados em diferentes velocidades e técnicas vocais.
- Transcrição de áudio em formato bruto (WAVE) para áudio final com timbre de instrumento musical real sem a utilização de um arquivo *MIDI* intermediário. A vantagem em não utilizar o formato *MIDI* como resultado final é a independência em relação a um conjunto pré-fabricado de sons, conhecido como *Wavetable* (VAIL, 2014), que pode não possuir a qualidade final ou característica musical desejada pelo usuário final.

1.5 Método de Trabalho

A abordagem de desenvolvimento desta pesquisa é composta por 5 etapas distintas:

1. Levantamento bibliográfico sobre Processamento de Sinais de Áudio, conceitos e técnicas de *Machine Learning* e *Deep Learning*, incluindo a Revisão Sistemática da literatura existente.
2. Seleção de Técnicas de pré-processamento de dados de áudio mais relevantes para treinamento dos algoritmos de *Machine Learning* e *Deep Learning*. Nesta etapa, também foi feita a escolha da arquitetura mais adequada das camadas de *Encoder* e *Decoder* das Redes Neurais Profundas, momento no qual é realizado uma proposta de experimento.
3. Implementar um protótipo executável de *software* para avaliar a viabilidade prática do estudo e validar conceitos básicos para realização do experimento e escolha do conjunto de dados (*dataset*) a ser utilizado.

4. Reunir e documentar os resultados descobertos durante a execução do experimento, relatando as situações encontradas durante o treinamento das redes neurais profundas, conjunto de hiperparâmetros escolhidos, arquitetura final das redes neurais profundas e desempenho alcançado ao realizar predições com o modelo treinado (o termo *predição* também é conhecido como *inferência*).

5. Síntese e comparações finais dos resultados obtidos em relação às Questões de Pesquisa, ao Estado da Arte e hipóteses levantadas inicialmente.

1.6 Organização da Dissertação

A seção 2 apresenta o resultado da etapa de revisão da literatura, onde os conceitos básicos e principais estudos encontrados são apresentados, além de discutir as técnicas consideradas como estado da arte.

A seção 3 trata da apresentação da proposta de uso combinado de redes CNN e LSTM para extração de melodia e sintetização de sons de instrumentos musicais a partir de exercícios de solfejo e apresenta os conjuntos de dados sugerido para o experimento.

A seção 4 discute a execução do experimento, descrevendo o ambiente computacional utilizado, arquitetura de referência para implementação do modelo proposto e reúne os resultados obtidos.

A seção 5 encerra a dissertação apresentando uma conclusão geral sobre a pesquisa, contribuições alcançadas, limitações verificadas no modelo proposto e sugestão de trabalhos futuros.

Demais detalhes sobre o Protocolo de Pesquisa da Revisão Sistemática e trechos de códigos-fontes utilizados são adicionados nos Apêndices e Anexos correspondentes.

2 REVISÃO DA LITERATURA

Antes de discorrer sobre as *features* de áudio em geral ou explorar aspectos específicos destas *features* em música, é necessário estabelecer alguns conceitos básicos que ajudam a entender a natureza do sinal sonoro e como suas propriedades podem ser estudadas no contexto de recuperação de informação musical, notadamente, com o uso de técnicas de *Machine Learning* e *Deep Learning*.

Nesta seção, também são apresentados os principais conceitos sobre *Machine Learning* e *Deep Learning* que são necessários para a compreensão desta Dissertação como um todo, baseando-se na literatura pesquisada.

2.1 Conceitos Básicos sobre Áudio e Música

É importante saber que as *features* de um áudio representam propriedades específicas do sinal de áudio e que estas propriedades variam conforme o tipo do sinal em questão. Por isso, para compreender o tema de extração e seleção de *features* de música é necessário recorrer a alguns conceitos preliminares que estão definidos na literatura básica.

O *survey* sobre *features* de áudio conduzido por Mitrović, Zeppelzauer e Breiteneder (2010) fornece informações relevantes para o entendimento das relações existentes entre áudio, música e *Machine Learning*.

Os demais conceitos básicos sobre áudio e música são abordados nas subseções a seguir.

2.1.1 Tipos de Sinais de Áudio

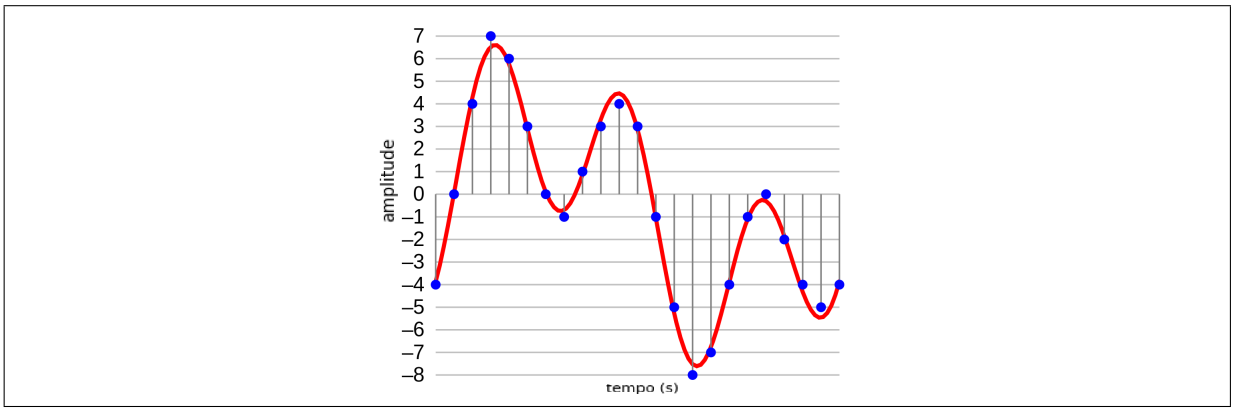
Existem distinções entre os diferentes tipos de sinais de áudio que podem ser interpretados como um ruído (barulho) ou como um som que emite um tom característico similar àqueles produzidos por um ser humano, um instrumento musical ou outra fonte sonora conhecida. Geralmente, os tons característicos atribuídos a determinadas fontes sonoras possuem uma frequência perceptível e reconhecível pelo nosso cérebro, pois seguem um padrão regular que nos permite classificá-los corretamente quando estimulam nossa audição, de acordo com as definições dada pela ANSI, Bioacoustical Terminology (1995 *apud* MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010, p. 11).

A Figura 1 ilustra uma amostragem de onda sonora simples, *em vermelho (linha cur-*

vilínea), representada em formato digital, *com pontos circulares na cor azul*, seguindo o padrão de arquivo *WAV* sem compressão (*PCM*), com 4-bits de resolução após quantização (VAIL, 2014).

Um tom puro é formado por uma onda sonora simples, isto é, criada por variações instantâneas da pressão do ar atmosférico ao longo de um intervalo de tempo com frequência regular, enquanto um tom complexo é formado por diferentes componentes de ondas sonoras que possuem frequências distintas uma das outras, de acordo com as definições dada pela ANSI, Bioacoustical Terminology (1995 *apud* MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010, p. 11).

Figura 1 – Representação digital de uma Onda sonora



Fonte: Adaptado de Widrow e Kollár (2008)

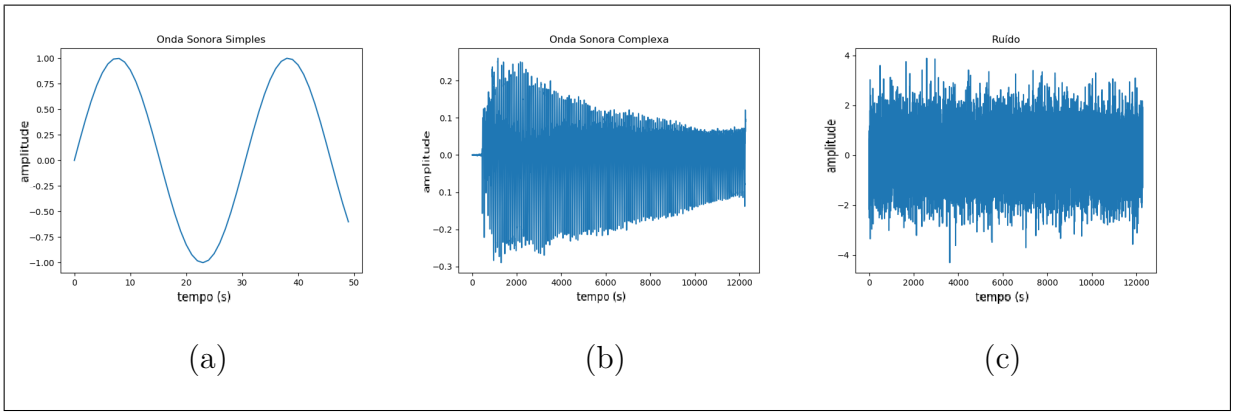
Uma comparação visual entre os tipos de onda sonora aparece na Figura 2 (a), que é um tipo de onda sonora simples e na Figura 2 (b), na qual é mostrada uma onda sonora complexa. Na Figura 2 (c) é mostrado um exemplo de ruído sonoro.

No caso dos tons complexos, estes ainda podem ser subclassificados como *harmônicos* e *inarmônicos*. Esta subclassificação permite diferenciar as interações e distâncias intervalares que podem ocorrer entre os componentes de frequência que constituem um sinal sonoro.

Segundo Mitrović, Zeppelzauer e Breiteneder (2010), o sinal de áudio harmônico é composto por várias ondas sonoras de diferentes frequências, mas que são múltiplas da frequência fundamental (que é a frequência dominante do sinal em questão).

Normalmente, o sinal de áudio harmônico produzido por um instrumento musical como o piano, ilustrado na Figura 2 (b), é estável o suficiente para produzir um som considerado agradável aos ouvidos de um ser humano (embora esta percepção seja subjetiva).

Figura 2 – Tipos de Ondas Sonoras



Fonte: Elaborado pelo autor.

Por sua vez, o sinal de áudio inarmônico é composto de ondas sonoras cujas frequências não possuem relação com a frequência fundamental como, por exemplo, os sons produzidos por instrumentos musicais percussivos (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010).

No caso do ruído sonoro (ou barulho) ilustrado na Figura 2 (c), este tende a causar uma sensação de desconforto aos ouvidos de um ser humano (também avaliado de forma subjetiva de indivíduo para indivíduo) (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010).

2.1.2 Atributos dos Sinais de Áudio

Mitrović, Zeppelzauer e Breiteneder (2010) destacam que do ponto de vista da psico-acústica, todos os sons possuem um grupo de atributos em comum que são denominados: duração (ou *duration*), volume (ou *loudness*), altura (ou *pitch*) e timbre. Estes atributos são descritos como:

- Duração – É o tempo em segundos decorrido entre o início e fim de um sinal de áudio, sendo que pode ser composto de subintervalos distintos de tempo que caracterizam as fases de ataque, decaimento, sustentação e repouso do som de acordo com a fonte sonora que o produziu (BENWARD; SAKER, 2009).

- Volume (ou *intensidade*) – Está relacionado ao nível de intensidade de pressão sonora que um som exerce em nossos ouvidos, sendo comumente definido em uma escala que vai de “suave” ou “baixo” até “forte” ou “alto”. Geralmente o volume é medido

em uma unidade logarítmica conhecida como decibéis (*dB*) (BENWARD; SAKER, 2009).

- Altura (ou *pitch*) – Está diretamente relacionada à frequência de um sinal de áudio e é medida em *Hertz* (*Hz*). Às vezes é utilizada intercambiavelmente como frequência fundamental de um som por não possuir uma definição única na literatura. Apesar disto, é percebida como uma faixa de som audível ao ouvido humano que vai do "grave" ao "agudo". Quanto menor a frequência da onda sonora, mais grave será o som enquanto uma maior frequência corresponde a sons mais agudos (BENWARD; SAKER, 2009).

- Timbre – É a característica sonora intrínseca de cada sinal de áudio que permite alguém diferenciar dois sons diferentes, ainda que possuam mesma intensidade e altura. Por exemplo, cada instrumento musical e cada voz humana possui um timbre característico que os diferenciam entre si (BENWARD; SAKER, 2009).

2.1.3 Notas Musicais

Os atributos de áudio citados na subseção 2.1.2 (isto é, duração, intensidade, altura e timbre) permitem compreender os conceitos básicos sobre as Notas Musicais (BENWARD; SAKER, 2009).

Cada nota musical está associada a uma duração, intensidade e altura específica podendo variar seu timbre de acordo com o instrumento musical ou voz humana na qual é produzida. A altura (no sentido de frequência do som) das notas musicais são medidas em *Hertz* (*Hz*) e de acordo com as escalas musicais adotadas no ocidente são nomeadas como: *dó*, *ré*, *mi*, *fá*, *sol*, *lá* e *si* (BENWARD; SAKER, 2009).

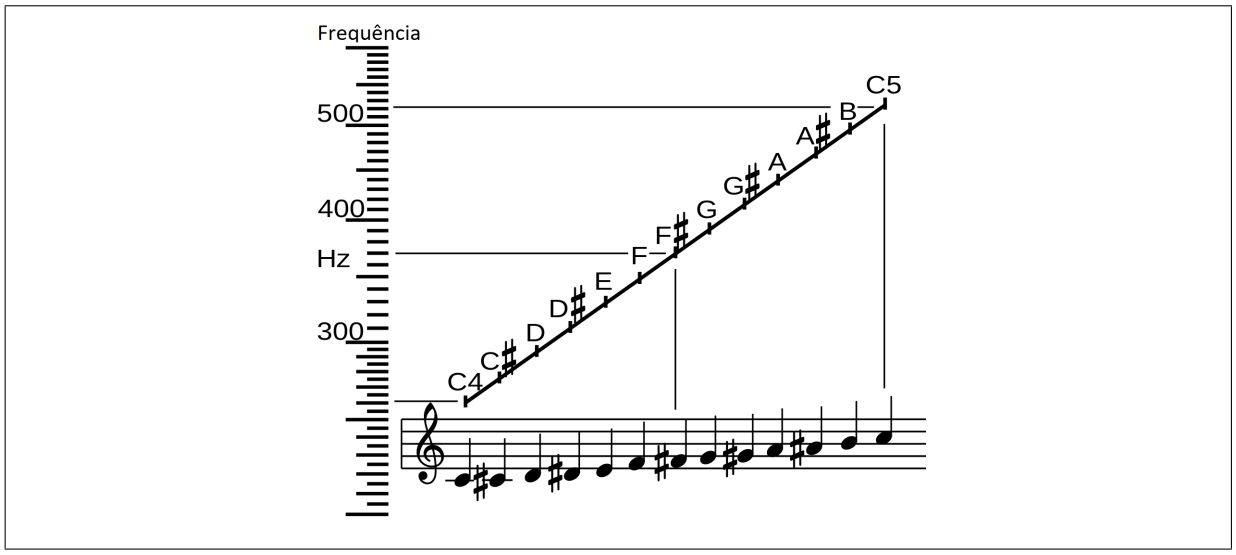
A escala musical mais simples é composta de uma sequência de notas musicais que se repetem de *dó* a *dó* separadas por intervalos de oito notas (também conhecido como *oitava*) (BENWARD; SAKER, 2009).

Em música, a *partitura* é a notação gráfica mais tradicional para se representar a escrita das notas musicais de forma padronizada. Em relação à *partitura*, também pode-se utilizar uma notação alternativa equivalente ao nome da nota musical que são formadas pelas letras maiúsculas *C*, *D*, *E*, *F*, *G*, *A*, *B*, representando *dó*, *ré*, *mi*, *fá*, *sol*, *lá* e *si*, respectivamente. Inclui-se outros símbolos especiais, como por exemplo, o sustenido (♯)

e o bemol (♭) que representam acidentes musicais que elevam ou reduzem o tom (ou frequência) de uma nota (BENWARD; SAKER, 2009).

A Figura 3 mostra a relação entre as Notas Musicais, sua escrita na *partitura* e sua relação com as frequências sonoras medidas em *Hertz* (*Hz*). As notas musicais apresentadas na Figura 3 começam no *dó central* que está escrito como *C4* e segue até a nota *dó* da oitava seguinte, representada por *C5*.

Figura 3 – Frequência em *Hz* das Notas Musicais



Fonte: Adaptado de Portal da Música (WIKIPEDIA, 2019)

2.1.4 Melodia, Ritmo e Harmonia

Existem conceitos musicais que são abstratos mas que são perceptíveis aos nossos sentidos e cognição humana. Assim sendo, embora possam divergir em suas definições formais e interpretação conforme variação nas opiniões dos ouvintes e autores acadêmicos da área de Música, algumas ideias essenciais já foram estabelecidas a respeito da natureza da Melodia, Ritmo e Harmonia.

A Melodia consiste em uma sequência de sons musicais, isto é, compostos por Notas Musicais com ordem previamente estabelecidas dentro de uma peça musical ou frase musical, respeitando critérios rítmicos e harmônicos de acordo com as convenções e restrições culturais existentes (SALAMON, 2013b).

O Ritmo determina a duração e a acentuação dos sons e das pausas (ou seja, dos silêncios existentes em determinados trechos de uma música). Uma importante observação

sobre o ritmo é que o gênero ou estilo musical é caracterizado conforme sua dinâmica e andamento. O ritmo é medido pelo *pulso* ou a *batida* (BENWARD; SAKER, 2009).

A Harmonia surge quando duas ou mais notas soam juntas de acordo com os diferentes modos de execução de maneira a produzir um agrupamento de sons considerados agradáveis por um ouvinte (embora esta percepção seja subjetiva). A Harmonia também pode ser referida como a dimensão vertical da música enquanto a Melodia representa a dimensão horizontal (BENWARD; SAKER, 2009).

No campo da Música, os conceitos sobre a Harmonia ajudam nos estudos relacionados à formação de acordes com base nas Teorias e Formalismos próprios desta área. Mas, de maneira simplificada, pode-se considerar que os acordes possuem uma estrutura musical preestabelecida e são formados por duas ou mais notas musicais que são executadas simultaneamente (admitindo variações no modo como são configuradas e tocadas).

2.1.5 Instrumentos Musicais Harmônicos, Inarmônicos e Melódicos

Segundo Lee (2019), o sistema de classificação de instrumentos musicais conhecido como *Hornbostel-Sachs* é o mais completo criado até hoje na área da Organologia, que é a disciplina responsável por estudar, descrever e classificar os instrumentos musicais.

Mas devido à natureza distinta de cada instrumento musical e complexidade dos sistemas de classificação formais como o *Hornbostel-Sachs*, esta dissertação adota uma maneira simplificada de classificação de instrumentos musicais de acordo com o tipo de timbre e frequências sonoras que produz, visto que esta perspectiva mostrou-se mais adequada para este estudo (respeitando os limites de seu escopo).

De acordo com Hopkin (1996), as diferentes frequências individuais de sons que podem estar presentes em uma nota musical são chamadas de *parciais* (*partials*, em inglês), sendo a frequência mais baixa definida como *frequência fundamental*. Assim, os *instrumentos musicais harmônicos* são aqueles capazes de produzir vários sons harmônicos cujas frequências são múltiplas da frequência fundamental.

Hopkin (1996) caracteriza ainda os *instrumentos musicais inarmônicos* como barulhentos ou ruidosos, pois produzem frequências de sons que não são múltiplas da frequência fundamental. Nesta classe de instrumentos, pode-se citar a bateria e o tambor como exemplos.

Por último, temos os *instrumentos musicais melódicos*. Segundo Vilela (2010), estes

instrumentos musicais são capazes de produzir apenas uma Nota Musical por vez, como é o caso da voz humana e da flauta.

Uma observação válida aqui é que instrumentos de cordas como a guitarra, o violão e o piano são classificados tanto como instrumentos harmônicos quanto instrumentos melódicos, pois são capazes de produzir os dois tipos de sons.

2.2 Seleção e Extração manual de *features* de Áudio

Cada tarefa específica de recuperação de informações baseadas em áudio requer um conjunto de *features* de áudio que sejam mais adequadas para cada situação ou problema que se quer resolver e ferramentas de *software* específicas (GIANNAKOPOULOS, 2015).

Por isso, no *survey* realizado por Mitrović, Zeppelzauer e Breiteneder (2010, cap. 3), a preocupação central foi reunir o máximo de *features* que frequentemente são utilizadas em diferentes domínios de aplicação, organizando estas *features* sob uma hierarquia que considera as propriedades formais estabelecidas na literatura especializada. Consequentemente, foi proposta uma nova taxonomia decorrente da pesquisa de mais de 70 features de áudio empregadas em trabalhos científicos que representavam o estado da arte da época (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010, cap. 4-5).

O Quadro 2 enumera as propriedades formais mais frequentes das *features* de áudio e seus valores ou tipos possíveis conforme identificadas por Mitrović, Zeppelzauer e Breiteneder (2010).

Uma *feature* de áudio pode possuir uma propriedade de *Representação de Sinal* correspondente. Assim, de acordo com o Quadro 2, uma *feature* pode ser agrupada conforme a maneira que codifica o sinal de áudio.

Quadro 2 – Propriedades formais para *features* de áudio

Propriedade	**Valores Possíveis**
Representação de Sinal	Codificado linearmente, Compressão com Perdas
Domínio	Temporal, Frequência, Correlação, Cepstral, Modulação de Frequência, Reconstrução de Espaço de Fase e Autodomínio
Escala Temporal	Intra-quadro, Inter-quadro, Global
Significado Semântico	Perceptivo, Físico
Modelo subjacente	Psicoacústico, Não-Psicoacústico

Fonte: Adaptado de Mitrović, Zeppelzauer e Breiteneder (2010, cap. 3).

Por exemplo, é possível utilizar o formato de áudio digital conhecido como FLAC para representar o sinal de áudio usando uma codificação linear (preservando todos os aspectos do sinal de áudio). De forma similar, pode-se utilizar os formatos de áudio digital chamados MPEG-2 ou MP3 para representar o sinal de áudio adotando esquemas de compressão com perdas (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010).

De acordo com Mitrović, Zeppelzauer e Breiteneder (2010), outra propriedade importante de uma *feature* de áudio é o *Domínio*. O *Domínio* diz respeito ao tipo de representação do sinal de áudio que uma *feature* provê após o processo de extração desta e da complexidade computacional associada.

Assim, no caso de uma onda sonora que é representada no *domínio temporal* a *feature* associada é a *Amplitude* (isto é, a variação da pressão do ar ao longo de um intervalo de tempo) (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010).

Uma questão importante sobre o *Domínio* de uma *feature* de áudio está relacionada à quantidade de transformações necessárias para se extrair uma dada *feature* e sua representação final.

A fórmula descrita na Equação 2.1 ajuda a entender a questão da extração de uma determinada *feature* de áudio e sua correta classificação em relação às propriedades mostradas na Quadro 2. O termo *feature vector* também é utilizado para expressar um vetor genérico contendo *features* extraídas para uma dada tarefa (GÉRON, 2017).

No caso de tarefas de classificação e regressão em áudio, é válido ressaltar que muitas *features* de áudio existentes no *Domínio da Frequência* aplicam alguma variante da Transformada Discreta de Fourier.

A fórmula mais comum para cálculo da Transformada Discreta de *Fourier* (DFT) em uma sequência de quadros de som é dada pela Equação 2.1 (versão simplificada) conhecida como *Short Time Fourier Transform (STFT)* (SERRA, 1989).

$$X(n,i) = \sum_{m=0}^{K-1} x_{[n+m]} w_{[m]} \cdot e^{-j\omega_i m} \tag{2.1}$$

A aplicação desta fórmula em um sinal de áudio converte a representação do som no *domínio temporal* para o *domínio de frequência*, permitindo a análise de quais frequências (medidas em *Hz*) compõe o espectro do sinal de áudio avaliado. O sinal de áudio resultante é representado em quadros sequenciais de tempo (*frames*) com duração em milissegundos determinada por uma função janela (denotada pela letra w) (SERRA, 1989).

Quando operações matemáticas baseadas na transformada discreta de *Fourier* (ou uma de suas variantes) são aplicadas a um sinal de áudio, o resultado desta transformação gera *features* de áudio que representam informações sobre o espectro do som, isto é, a interpretação destes dados deve ocorrer no *domínio da frequência* (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010).

Porém, é necessário ressaltar que nem todas as operações baseadas em *Fourier* resultam imediatamente no domínio final da *feature* que está sendo extraída, deixando os dados com uma representação intermediária. Portanto, para se classificar e selecionar uma *feature* de áudio corretamente deve-se aplicar todas as transformações e cálculos matemáticos necessários a fim de considerar apenas a representação dos dados no *Domínio* final (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010).

Outra propriedade importante das *features* de áudio é a *Escala Temporal* que pode ser do tipo intra-quadro, inter-quadro ou global. Diferentes tipos de *Escala Temporal* ocorrem porque há certas *features* de áudio que podem ser extraídas a partir de fragmentos parciais do áudio, enquanto *features* de áudio mais complexas podem requerer o áudio inteiro.

Neste caso, um *quadro* (traduzido do inglês *frame*) possui uma determinada duração de tempo, tipicamente medida em milissegundos. Desta maneira, é possível analisar um sinal de áudio dentro de um mesmo intervalo de tempo, ou ainda, utilizar vários quadros para se obter a representação de áudio desejada ao extrair-se uma dada *feature* de áudio. Seguindo esta lógica, quando uma informação ou representação de áudio só pode ser obtida após o processamento do sinal de áudio inteiro, se diz que a *Escala Temporal* da *feature* em questão é do tipo *global* (MITROVIĆ; ZEPPELZAUER; BREITENEDER, 2010).

Ainda de acordo com o Quadro 2, pode-se observar uma propriedade chamada *Significado Semântico*. Se uma dada tarefa de recuperação de informação de áudio envolver aspectos dependentes da percepção humana, então se diz que a *feature* de áudio em questão possui *Significado Semântico* do tipo *perceptivo* (traduzido do inglês *perceptual*).

Por fim, *features* de áudio que são extraídas com base em modelos matemáticos que tentam aproximar ou imitar o sistema sensorial auditivo do ser humano são *features* cuja a propriedade denominada *Modelo Subjacente* que está listada na Quadro 2 é do tipo *Psicoacústico*, do contrário, classifica-se como *Não-Psicoacústico*.

A partir dos conceitos apresentados por Mitrović, Zeppelzauer e Breiteneder (2010), resumidos no Quadro 2, pode-se concluir que ao se identificar as propriedade de um dado conjunto de *features* de áudio é possível conduzir o processo de seleção de *features* para

uma dada tarefa de extração de *features* e seu uso em algoritmos de *Machine Learning*.

Para os experimentos realizados nesta dissertação, as principais *features* de áudio investigadas estão nos domínios temporais e de frequência. Contudo, vale notar que *features* do domínio *Cepstral* podem ser úteis para capturar informações sobre Timbres.

De forma similar, pode-se explorar *features* cujo *domínio* possibilite representações de áudio no *Autodomínio* para lidar com redução de dimensionalidade ou aplicação de filtros de áudio.

Sobre a voz humana cantada ou falada, é importante notar que após o processo de extração de *features* de áudio ter sido realizado adequadamente, um sinal de áudio contendo fala humana pode ser expressado como uma sequência de vetores contendo *features* de áudio (SAKOE; CHIBA, 1978).

O Anexo A contém informações e figuras complementares sobre o conjunto de *features* mais frequente em processamento de áudio e música de acordo com Mitrović, Zeppelzauer e Breiteneder (2010).

Para capturar as informações harmônicas que estão presentes em um sinal de áudio monofônico como o solfejo, é útil aplicar outra transformação do sinal conhecida como *Constant-Q Transform* (CQT). Através de métodos baseados em CQT, também é possível converter a representação de um som do *domínio temporal* para sua forma correspondente no *domínio de frequência* (SCHÖRKHUBER; KLAPURI, 2010).

Os detalhes sobre a formulação matemática da transformação CQT estão além do escopo desta dissertação e as devidas explicações e seu uso em música podem ser encontradas nos artigos escritos por Schörkhuber e Klapuri (2010) e Bittner et al. (2017), respectivamente. No entanto, a biblioteca de *software* desenvolvida por McFee et al. (2015) fornece uma implementação da CQT que é baseada em DFT conforme especificada por Schörkhuber e Klapuri (2010).

2.3 Algoritmos de *Machine Learning*

Para Alpaydin (2010, p. 3), *Machine Learning* é uma técnica de programação de computadores que permite otimizar o desempenho de um algoritmo, dado um conjunto de critérios, com base em exemplos de dados ou experiências passadas. Assim, o objetivo é descobrir um modelo matemático, cujo algoritmo exato é desconhecido, por meio da otimização de parâmetros do modelo que serão ajustados à medida que os dados ou

exemplos são utilizados em seu treinamento. Uma vez que o modelo esteja pronto, poderá ser utilizado para realizar inferências sobre um dado problema em busca de soluções eficientes.

Raschka (2015), destaca o fato de que o *Machine Learning* evoluiu como um subcampo específico da Inteligência Artificial como forma de aprendizagem automática que adquire conhecimentos a partir de dados com o objetivo de realizar predições (termo sinônimo ao termo *inferência*, de uso comum na estatística).

Por sua vez, Samuel (1959 *apud* GÉRON, 2017) define *Machine Learning* como a ciência e a arte de programação de computadores de forma que estes aprendam a partir dos dados sem serem explicitamente programados. Ou ainda, seguindo uma definição mais formal:

> "É dito que um programa de computador aprende com a experiência E com respeito a alguma tarefa T e alguma métrica de desempenho P, se este desempenho na tarefa T, como medido por P, melhorar com a experiência E."
>
> Mitchell (1997 *apud* GÉRON, 2017)

O fato comum entre estas definições está em aceitar que existe uma família de algoritmos de computador que depende de dados para aprender a executar uma dada tarefa e com base nas explicações de Raschka (2015) e Géron (2017), observa-se que os tipos de aprendizagem de máquina possuem quatro categorias quanto à sua forma de treinamento:

- **Aprendizado Supervisionado**
  - Existe um conjunto de *features* já selecionadas e dados previamente classificados através de um rótulo, classe ou categoria no caso de uma tarefa de classificação ou um valor contínuo que representa o resultado esperado, aproximado ou exato, para uma tarefa de regressão.
- **Aprendizado Não-Supervisionado**
  - Não há rótulos ou classificação prévia dos dados.
  - Os dados são agrupados de acordo com certos padrões ou características similares que são identificadas automaticamente à partir das amostras de dados (também conhecidas como *instâncias*).

- **Aprendizado Semi-Supervisionado**
  - É um caso especial de aprendizado Supervisionado no qual nem todos os dados do conjunto de treinamento estão rotulados. Geralmente os algoritmos de aprendizagem que caem nesta categoria são uma combinação de algoritmos supervisionados com não-supervisionados.
- **Aprendizado por Reforço**
  - Um agente inteligente interage com o ambiente onde está sendo executada ou simulado e aprende por reforço, isto é, mediante alguma estratégia de recompensa ou penalização.

De acordo com LeCun, Bengio e Hinton (2015), a forma mais comum de *Machine Learning* recai na categoria de *Aprendizado Supervisionado*. Assim, quando se deseja construir um sistema capaz de classificar corretamente um conjunto de imagens, por exemplo, contendo um carro, um gato ou um cachorro é necessário apresentar um conjunto de dados previamente rotulados para que o sistema possa aprender durante a fase de treinamento.

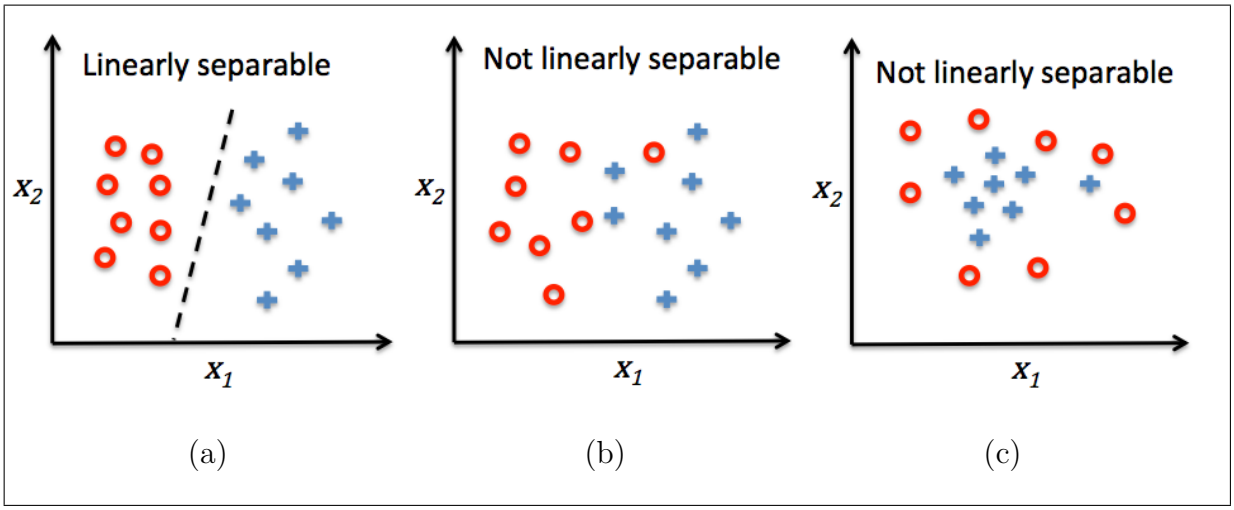
Para entender alguns problemas de classificação que são resolvidos por algoritmos de *Machine Learning* é necessário distinguir quais dados são linearmente separáveis ou não, conforme ilustrado na Figura 4.

Os símbolos com circunferências (*em vermelho*) e cruzes (*em azul*) que aparecem na Figura 4 denotam amostras de dados pertencentes a classes distintas de um problema de classificação binária. A linha reta tracejada no gráfico da Figura 4 (a) é chamada de *fronteira de decisão* (*Decision Boundary*) e separa perfeitamente as amostras no plano bidimensional de acordo com as classes existentes.

Nos gráficos exibidos na Figura 4 (b) e Figura 4 (c), nota-se que não é possível traçar uma linha reta como fronteira de decisão perfeita de forma que cada amostra seja classificada corretamente.

No caso das tarefas de regressão em problemas de Aprendizado Supervisionado, o objetivo é fazer a predição de variáveis que possuem valores contínuos ao invés de valores discretos como nas tarefas de classificação (RASCHKA, 2015). Um exemplo típico de tarefas de regressão são aquelas nas quais utilizam-se algoritmos de *Machine Learning* capazes de realizar regressão linear para, por exemplo, fazer a predição do preço de um

Figura 4 – Dados Lineares e Não-Lineares



Fonte: Raschka (2015, p. 23).

carro de acordo com um dado conjunto de *features* (poderia ser o peso do carro, ano de fabricação e a potência do motor, por exemplo).

O problema da regressão linear simples (que possui somente uma variável independente ou explanatória e uma variável de resposta) pode ser representado pelo modelo linear que aparece na Equação 2.2.

$$\hat{y} = w_0 + w_1 \cdot x \tag{2.2}$$

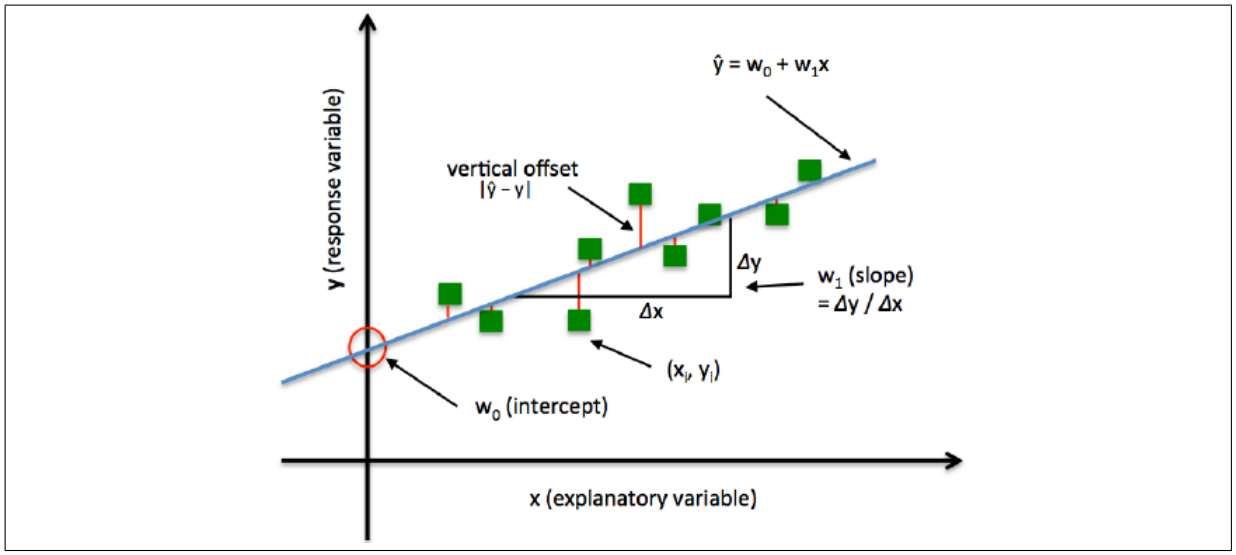
Na Equação 2.2, o $\hat{y}$ é a *variável de resposta* que recebe o valor contínuo referente à predição, o x é a variável independente (ou explanatória) e os parâmetros w_0 e w_1 da possuem valores desconhecidos que serão aprendidos durante o treinamento do modelo.

A partir da Equação 2.2, pode-se entender que uma regressão linear simples busca encontrar os parâmetros w_0 e w_1 ideais de forma a traçar uma reta, conhecida como *reta de regressão*, que melhor se ajuste a um dado conjunto de dados que respeitem uma distribuição linear (RASCHKA, 2015).

Um modelo esquemático de regressão linear simples é mostrado na Figura 5. O eixo x representa a *feature* selecionada (ou variável explanatória) e o eixo y representa a variável de resposta (predição). O ponto na qual a reta da equação $\hat{y} = w_0 + w_1 \cdot x$ intercepta o eixo y (destacada com a circunferência de cor *vermelha*) é chamado *intercepto* e é descrito no modelo da Equação 2.2 através do parãmetro w_0 enquanto a inclinação da reta traçada é dada pelo parâmetro w_1.

O modelo de regressão linear tenta encontrar o ajuste ideal da reta em relação ao

Figura 5 – Modelo de Regressão Linear Simples



Fonte: Raschka (2015, p. 278).

conjunto de dados (que aparecem com o símbolo de um quadrado na cor *verde*) por meio de otimização das distâncias entre a reta e cada amostra do conjunto de dados (a distância a ser otimizada é representada por um traço vertical na cor *vermelha*).

Quando existe mais de uma variável independente ou várias *features* em um modelo de regressão linear, pode-se generalizar este modelo através da Equação 2.3 e este passa a ser chamado de *regressão linear múltipla*, sendo w_0 o intercepto e $x_0 = 1$.

$$\hat{y} = w_0 \cdot x_0 + w_1 \cdot x_1 + \cdots + w_m \cdot x_m = \sum_{i=0}^{n} w_i x_i = w^T x \tag{2.3}$$

2.4 Redes Neurais Artificiais e *Deep Learning*

LeCun, Bengio e Hinton (2015) afirmam que as técnicas de *Machine Learning* são limitadas quando trata-se de processamento de dados naturais não-estruturados ou dados em forma bruta. Para isto, um conjunto de métodos mais avançados, conhecidos sob o termo *Representation Learning*, são usados para permitir que o computador descubra automaticamente novas representações à partir de dados brutos e consigam, por exemplo, realizar tarefas de classificação.

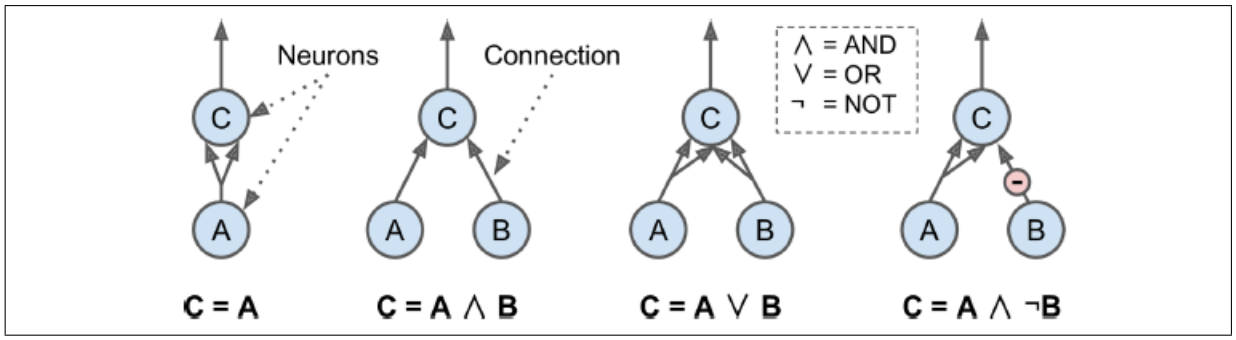
Neste contexto, LeCun, Bengio e Hinton (2015) definem *Deep Learning* como um conjunto de métodos que utiliza *Representation Learning* dentro de uma abordagem que possui múltiplos níveis de representações, criando abstrações à partir da composição de

vários módulos não-lineares que transformam dados brutos em representações de alto nível. Assim é possível extrair *features* automaticamente à partir de dados como imagens e áudio.

As Redes Neurais Artificiais (ANN) são algoritmos inspirados no funcionamento do cérebro humano e o modelo matemático subjacente fornece o embasamento teórico para compreender os algoritmos de *Deep Learning*.

Segundo Raschka (2015), os cientistas Warren McCulloch e Walter Pitts, inspirados por um neurônio biológico, propuseram um modelo simplificado de um neurônio artificial por volta do ano de 1940. Este modelo possuía uma ou mais entradas binárias e uma saída binária. Assim, era capaz de executar simples computações lógicas do tipo *E*, *OU* e *NÃO*, conforme exemplificado na Figura 6 (GÉRON, 2017).

Figura 6 – Neurônio Artifical Simples



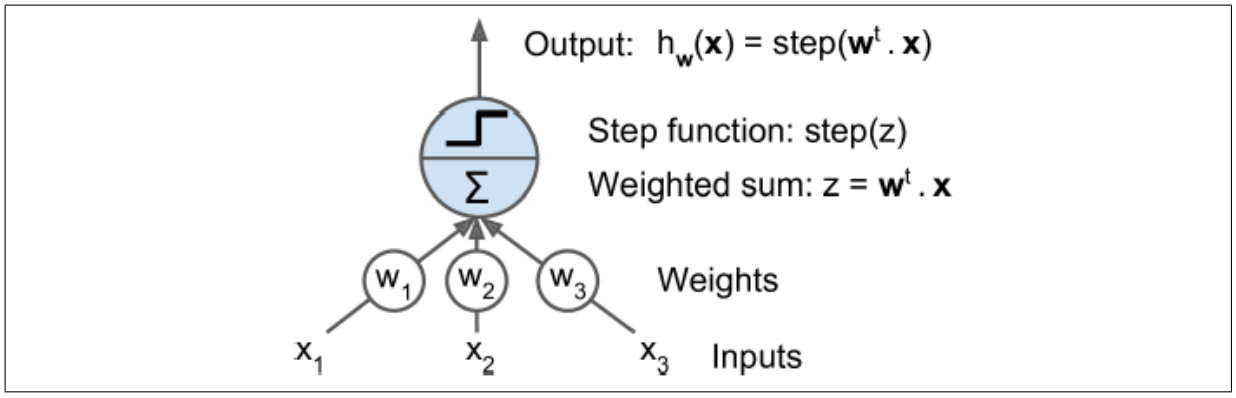
Fonte: Géron (2017, p. 256).

A partir deste modelo primitivo, o matemático Frank Rosenblatt criou a primeira arquitetura de Rede Neural Artificial em 1957 chamada de *Perceptron*. Com alguns avanços notáveis em relação a versão inicial do neurônio artificial criado por McCulloch-Pitts, esta rede era composta de unidades computacionais denominada *linear threshold unit (LTU)* que passou a operar valores numéricos reais, eliminando a limitação de aceitar entradas e saídas exclusivamente binárias (0 ou 1). O modelo matemático que define a LTU é expressado pela Equação 2.4, na qual uma soma ponderada é feita para se obter o valor de entrada da rede z (GÉRON, 2017, p. 257).

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = W^T \cdot X \tag{2.4}$$

Após a realização da soma ponderada da Equação 2.4, a função de ativação conhecida como *step function* é aplicada na saída z da LTU para obter o resultado final, conforme o esquema ilustrado na Figura 7.

Figura 7 – Neurônio Artifical do tipo LTU

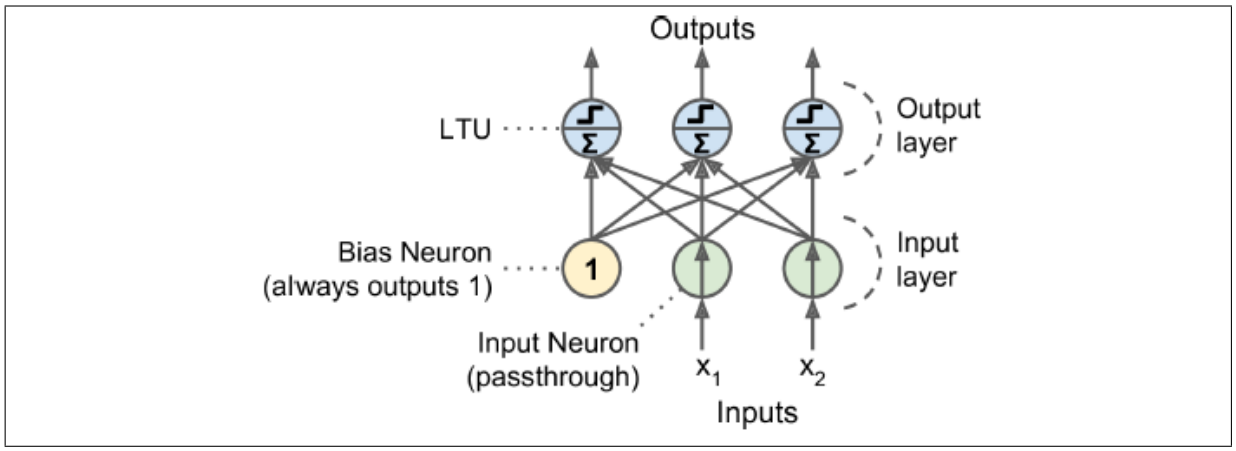


Fonte: Géron (2017, p. 257).

O *Perceptron* criado por Rosenblatt é formado agregando várias unidades LTU conforme a Rede Neural Artificial ilustrada na Figura 8. No exemplo, é mostrado um diagrama do *Perceptron* com duas entradas e três saídas.

As funções do tipo *step function* usualmente utilizadas no *Perceptron* são as chamadas *Heavside step function* e *Sign step function* (GÉRON, 2017).

Figura 8 – Rede Neural do tipo *Perceptron*



Fonte: Géron (2017, p. 261).

A partir desta configuração básica da Rede Neural Artificial surgiram outras propostas de arquiteturas de Redes Neurais Artificiais que empregavam mais camadas ocultas em sua topologia como é o caso da Rede Neural denominada *Multi-Layer Perceptron (MLP)* que é bastante difundida na literatura como sendo precursora das Redes Neurais Profundas, que é quando uma Rede Neural Artificial possui duas ou mais camadas ocultas conforme explicado por LeCun, Bengio e Hinton (2015) e Géron (2017, p. 261).

Deste ponto em diante da história, em 1974 especificamente e a partir de 1986 quando se estabeleceu novas técnicas de treinamento de Redes Neurais Profundas com a recriação do algoritmo *Backpropagation* por Rumelhart, Hinton e Williams (1986 *apud* GÉRON, 2017) é que a área de *Deep Learning* passou a desenvolver-se até chegar no grau de maturidade que temos nos dias atuais.

Mais informações sobre Redes Neurais Artificiais e suas vertentes podem ser encontradas nos livros de Haykin (2007), Raschka (2015), Géron (2017), Goodfellow (2016), dentre outros.

2.4.1 Redes Neurais Convolucionais

O surgimento das Redes Neurais Convolucionais foi motivado a partir do estudo do córtex cerebral de gatos e macacos entre os anos de 1958 e 1959 com o objetivo de mapear o funcionamento da visão nestes animais e avançar no entendimento da visão do ser humano (GÉRON, 2017).

Com o avanço dos computadores e tecnologias de aceleração gráficas, a CNN tem sido usada com sucesso em várias atividades de reconhecimento de imagens no campo da visão computacional(LECUN; BENGIO; HINTON, 2015). O funcionamento desta rede imita o comportamento do córtex cerebral visual ao permitir que apenas uma pequena região chamada de campo receptivo local seja estimulada quando exposta à presença de certos padrões de imagens (GÉRON, 2017).

Então, percebeu-se que seria possível explorar novas arquiteturas de construção de Redes Neurais Artificiais com essas características do córtex cerebral visual. Esta organização da rede mostrou-se eficiente por utilizar menos conexões entre os neurônios e permitir a formação de uma rede hierárquica com capacidade de aprendizado de padrões de baixo, médio e altos níveis de abstração.

Quando esta família de algoritmos é utilizada para processar uma imagem genérica contendo uma foto de uma pessoa, um cachorro ou um carro, esta rede consegue extrair informações semânticas que partem de estruturas mais básicas como o contorno de uma linha, para depois aprender características de iluminação e cores de uma cena e gradualmente ir juntando todas estas partes até alcançar a compreensão do todo, limitado apenas à capacidade disponível de memória computacional e velocidade de processamento.

É válido ressaltar que são necessários milhares de dados dependendo da tarefa de

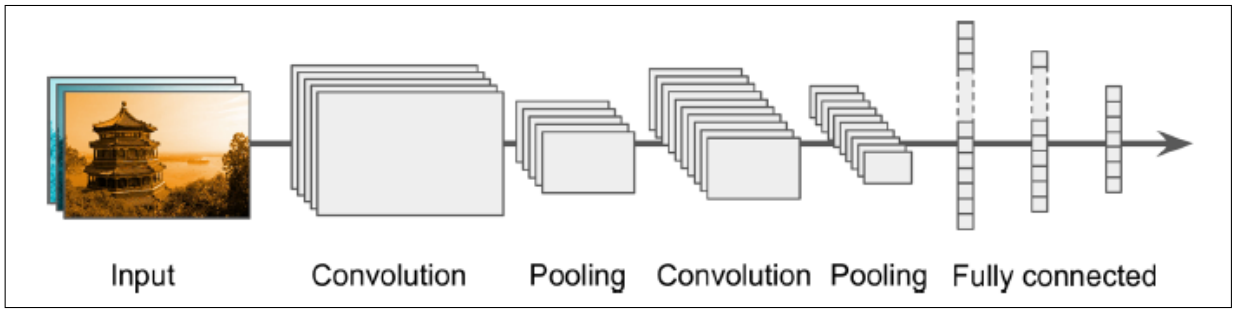
classificação de imagens ou vídeos que se quer realizar para que se obtenha resultados eficazes (LECUN; BENGIO; HINTON, 2015).

Com o crescimento do interesse e aumento de pesquisas sobre o uso de CNNs descobriu-se que esta rede também é eficiente para certas tarefas de processamento de áudio. Assim, diferentes implementações de CNN foram contempladas com suporte a dados unidimensionais como o som, além das estruturas bidimensionais e tridimensionais (LECUN; BENGIO; HINTON, 2015).

Um exemplo típico de arquitetura de Rede Convolucional é ilustrada na Figura 9 com as camadas convolucionais e de *pooling* intercaladas entre si para realizar uma tarefa de classificação de imagem (GÉRON, 2017).

Na camada final da Rede Convolucional que aparece na Figura 9 existe uma Rede Neural do tipo *Fully Connected* que é responsável por obter a saída da rede (GÉRON, 2017).

Figura 9 – Rede Neural Convolucional



Fonte: Géron (2017, p. 365).

A principal diferença entre a Rede Convolucional e outras Redes Neurais está no fato da CNN aprender novas características a partir de uma operação matemática denominada *convolução* (Géron, 2017). A convolução matemática é um tipo de função que realiza multiplicações entre matrizes de filtros conhecidos como *kernels* que atuam somente em determinadas regiões do campo receptivo local em busca de padrões.

O resultado do processamento dos filtros dá origem aos chamados mapas de *features* que são extraídas após a realização de convoluções discretas, normalmente, em espaços unidimensionais, bidimensionais ou tridimensionais (GÉRON, 2017).

A Rede Neural Convolucional, geralmente, é construída com várias camadas convolucionais que ajudam no aprendizado de um conjunto de pesos ideais para cada filtro de acordo com a tarefa de classificação em questão (GÉRON, 2017).

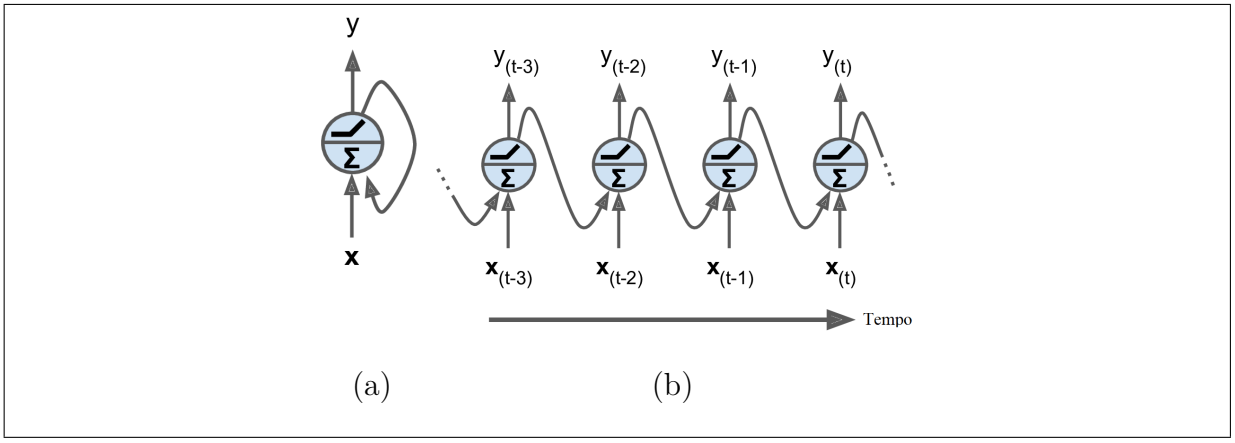
Outro bloco básico da Rede Convolucional é o *pooling* que é uma operação vetorial aplicada em camadas auxiliares que possuem o propósito específico de redução de dimensionalidade dos dados e redução no tempo de processamento dos dados, otimizando os recursos computacionais.

2.4.2 Redes Neurais Recorrentes

Dados estruturados como sequências possui uma ordem explícita para os eventos observados no decorrer do tempo e esta ordem deve ser estritamente respeitada (BROWNLEE, 2017).

Uma Rede Neural Recorrente (RNN) é similar a uma rede do tipo *FeedForward*, sendo a que a principal diferença está na sua capacidade de enviar o sinal de saída ***de volta*** para a entrada através de neurônios recorrentes, conforme ilustrado na Figura 10.

Figura 10 – Neurônio Recorrente

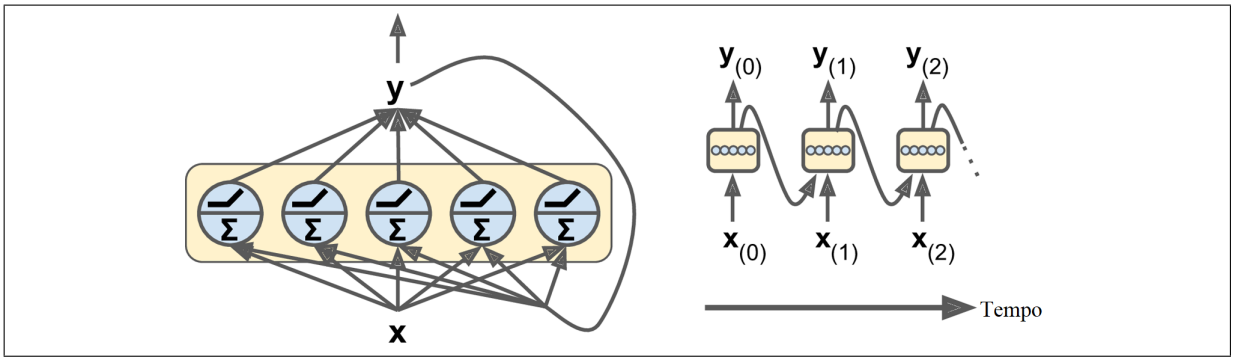


Fonte: Adaptado de Géron (2017, p. 380).

O Neurônio Recorrente exibido na Figura 10 (a) ilustra o sinal de saída retornando para a entrada do neurônio. Este mesmo neurônio é exibido na Figura 10 (b) na forma conhecida como *unrolled through time* (ou em tradução literal: *desenrolado através do tempo*), na qual é possível acompanhar os estados internos do neurônio recorrente em cada instante de tempo t.

A Rede Neural Recorrente do tipo *Long Short-Term Memory (LSTM)* ilustrada na Figura 11 é um caso especial de RNN que é eficaz em tarefas de classificação de sequências, tarefas de regressão envolvendo sequências, séries temporais, tradução de textos e modelagem de sinais acústicos segundo Brownlee (2017) e Géron (2017).

Figura 11 – Rede Neural Recorrente



Fonte: Adaptado de Géron (2017, p. 380).

Brownlee (2017) explica que a *predição de sequência* é um tipo de problema que é resolvido por aprendizado supervisionado. Sendo assim, em problemas de *predição de sequência*, uma sequência arbitrária de dados pode ser fornecida como entrada para um *modelo preditivo* apropriado e este pode produzir uma sequência de tamanho variado ou fixo.

Segundo Everitt (1995), uma *série temporal* é uma sequência de dados obtidos em intervalos regulares de tempo de acordo com um período específico. Neste sentido, de acordo com Brownlee (2017), uma sequência de dados de áudio ou uma onda sonora também podem ser consideradas como uma *série temporal*.

Gerhard (2002) sugere ainda que um som pode ser considerado como uma série de *pitches* (disponível em um *feature vector* previamente criado) que podem ser analisados através de técnicas de *pitch tracking* (SALAMON, 2013b, p. 26).

2.5 Medidas de Similaridade

Em alguns algoritmos de *Machine Learning* é necessário utilizar alguma métrica de distância ou critério específico para indicar similaridades entre as amostras de um conjunto de dados ou entre *features vectors* (GÉRON, 2017).

Considere dois vetores contendo um conjunto de *features* definidos como a_i e $b_j \in \mathbb{R}^d$ (onde d significa a quantidade de *features* em cada vetor). Então, duas sequências A e B contendo *features vectors* podem ser definidas por:

$$\begin{aligned} A &= a_1, a_2, \cdots, a_i, \cdots, a_n \\ B &= b_1, b_2, \cdots, b_j, \cdots, b_n \end{aligned} \tag{2.5}$$

A questão é: *como medir a similaridade entre duas sequências de vetores?*

Segundo Tsiporkova (2006) e Shou, Mamoulis e Cheung (2005), uma abordagem possível para medir a similaridade entre sequências é utilizar o algoritmo chamado *Dynamic Time Warping (DTW)*.

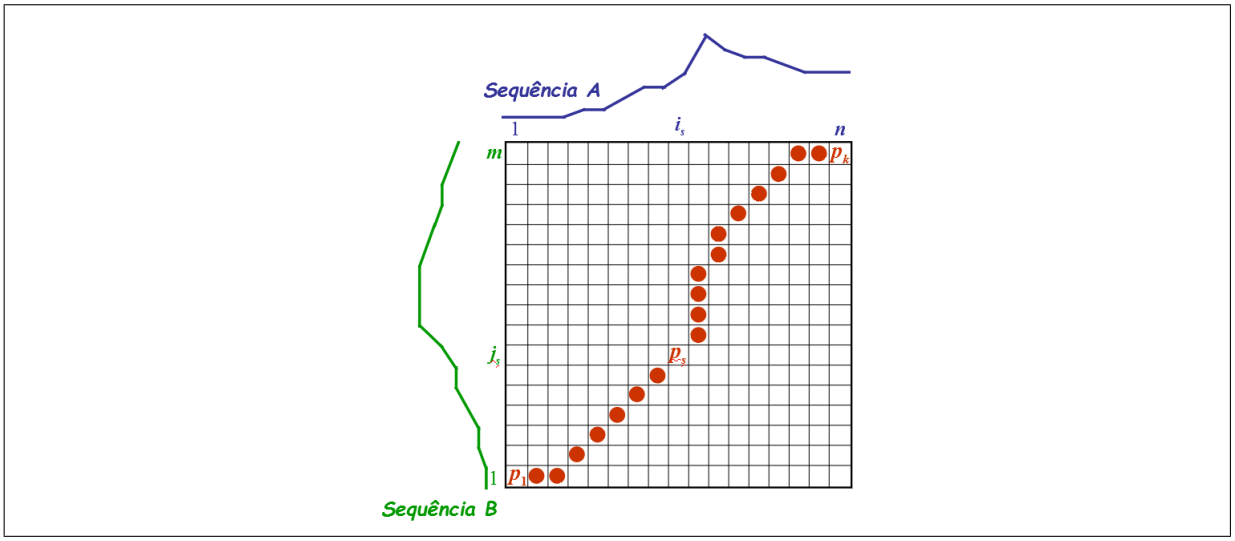
O DTW foi criado por Sakoe e Chiba (1978 *apud* TSIPORKOVA, 2006) no ano de 1978 e foi desenvolvido originalmente para reconhecimento de fala e alinhamento de séries temporais.

O objetivo do DTW é alinhar duas seqüências quaisquer que contenham *features vetors* ainda que existam diferenças entre os tamanhos das sequências, permitindo medir a similaridade entre as sequências com base numa função de distância que desconsidera as flutuações naturais da taxa de velocidade na qual a voz é reproduzida. Estas flutuações que ocorrem no eixo do tempo quando são eliminadas através de técnicas de normalização do tempo (SAKOE; CHIBA, 1978).

O algoritmo do DTW funciona de forma iterativa, distorcendo o eixo do tempo através de uma função chamada *Warping*, até encontrar uma correspondência ideal (de acordo com uma métrica adequada) entre as duas sequências (TSIPORKOVA, 2006).

Considerando duas sequências A e B de tamanhos similares, a Figura 12 ilustra o caminho ótimo obtido com o uso do DTW, na qual os círculos vermelhos que aparecem na Figura 12 representam a menor distância encontrada pelo algoritmo DTW entre as sequências A e B.

Figura 12 – Função *Warping*



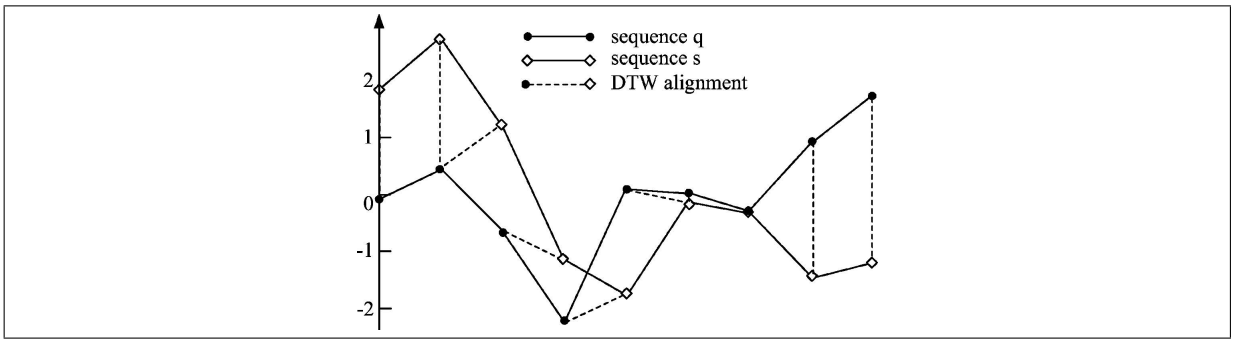
Fonte: Adaptado de (TSIPORKOVA, 2006).

As duas sequências são dispostas próximas a uma matriz com $n \times m$ dimensões. A sequência A é distribuída horizontalmente no topo da matriz e a sequência B é distribuída verticalmente no lado esquerdo da matriz.

O cálculo de distância utilizado pelo DTW é uma versão modificada da distância Euclidiana que permite comparar sequências que podem variar no tempo (ou na velocidade, no exemplo do som) (SHOU; MAMOULIS; CHEUNG, 2005).

A Figura 13 mostra um exemplo de duas sequências $\vec{q}$ e $\vec{s}$ de diferentes tamanhos e o respectivo alinhamento (linha tracejada) aproximado pelo algoritmo DTW, servindo como uma forma de medida de similaridade entre as duas sequências.

Figura 13 – Exemplo de duas sequências $\vec{q}$ e $\vec{s}$ de tamanhos diferentes



Fonte: Shou, Mamoulis e Cheung (2005).

2.6 Problemas relacionados ao Conjunto de Dados

A quantidade e a qualidade do conjunto de dados ou *dataset* utilizado para treinamento dos algoritmos de *Machine Learning* e *Deep Learning* é de fundamental importância, pois afeta diretamente a capacidade de generalização dos modelos. Os tipos mais comuns de problemas que ocorrem na preparação dos modelos são comentados brevemente na subseção 2.6.1 e na subseção 2.6.2 e compreendem:

- *Overfitting* e *Underfitting*
- *Viés* e *Variância*

Existem algumas técnicas que buscam diminuir os problemas relacionados à falta de disponibilidade de dados para treinamento dos modelos de *Machine Learning* e *Deep Learning* como o uso de *Cross Validation* que é explicado em Raschka (2015) ou técnicas

mais avançadas que lidam com *séries temporais* como a técnica de *Data Augmentation* discutida por Fawaz et al. (2018).

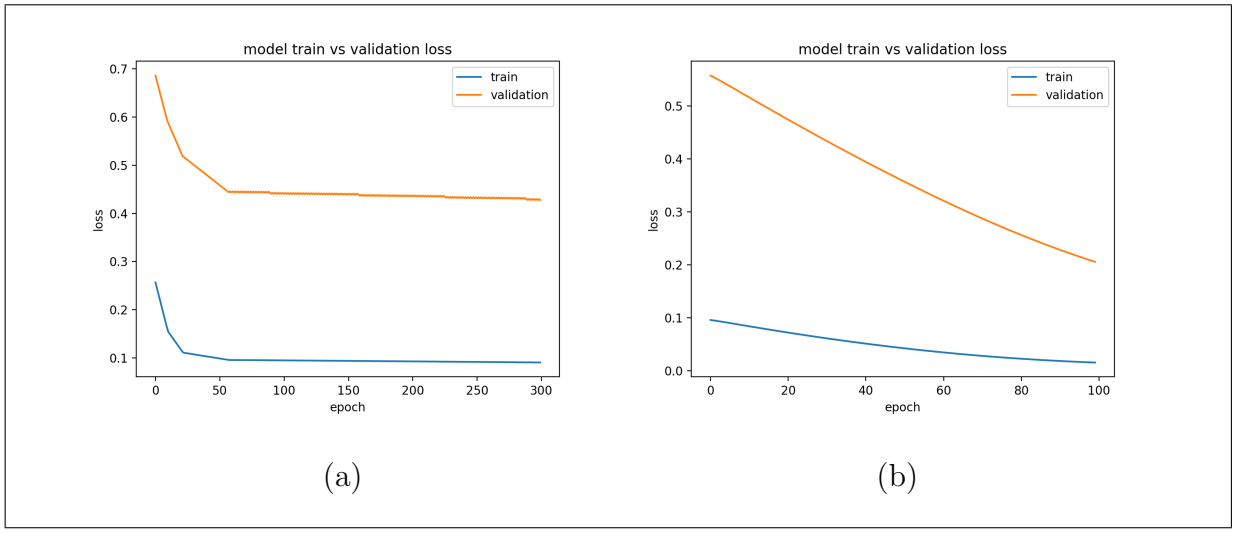
2.6.1 *Overfitting e Underfitting*

Quando o modelo não possui flexibilidade o suficiente para representar um *dataset*, dizemos que este modelo sofre de *Underfitting*. A principal característica neste cenário é que o gráfico de treinamento costuma exibir um pior resultado para o *dataset* de treinamento em relação ao *dataset* usado para validação ou testes (BROWNLEE, 2017).

Quando um modelo de *Machine Learning* ou *Deep Learning* ajusta-se demasiadamente a um *dataset* o modelo afetado só obtém resultados aceitáveis no *dataset* de treinamento utilizado, tendo um péssimo desempenho no *dataset* de testes, de validação ou em dados reais apresentados ao modelo. Este cenário é conhecido como *Overfitting* (BROWNLEE, 2017).

No caso de *Underfitting*, além da questão do modelo utilizado não ser complexo ou flexível o suficiente para representar os dados, em alguns casos este sintoma ocorre por causa do interrompimento prematuro durante o treinamento do modelo (BROWNLEE, 2017). Os sintomas de *Underfitting* são ilustrados na Figura 14 (a) e ocorre quando o modelo não possui flexibilidade. Na Figura 14 (b) é mostrado um cenário no qual o treinamento foi interrompido precocemente.

Figura 14 – Gráfico de Modelo com *Underfitting*

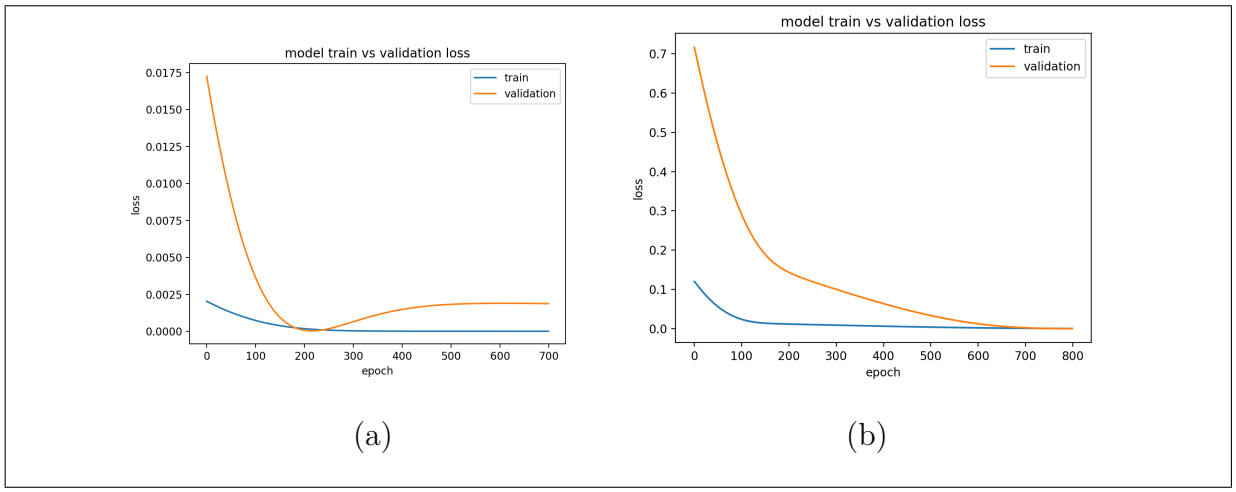


Fonte: Brownlee (2017).

Para resolver as situações de *Overfitting*, geralmente, é necessário usar técnicas de penalização do modelo ou ainda aumentar o conjunto de dados quando possível. Uma técnica mais simples consiste em parar o treinamento do modelo antes de ocorrer o problema de *Overfitting* (ou sobreajuste) (BROWNLEE, 2017).

A Figura 15 (a) mostra uma situação de *Overfitting* e a Figura 15 (b) apresenta o resultado de um modelo equilibrado que não sofre nem de *Overfitting*, nem de *Underfitting*.

Figura 15 – Gráfico de Modelo com *Overfitting* e Modelo Ideal



Fonte: Brownlee (2017).

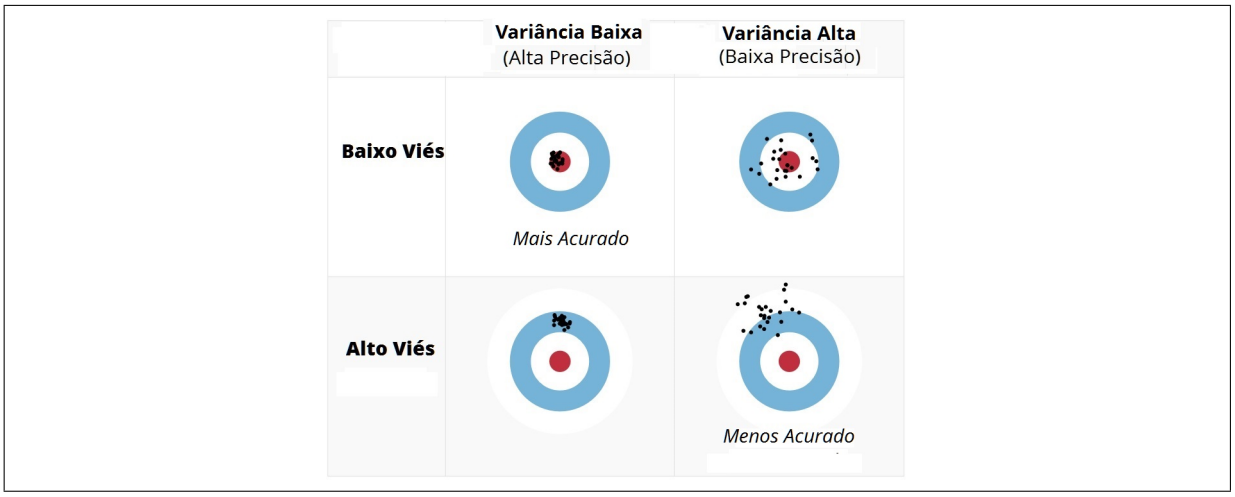
Através da análise exploratória dos dados e técnicas de estatística descritiva é possível verificar alguns aspectos quantitativos de um *dataset* que ajudam a delimitar os possíveis algoritmos de *Machine Learning* e *Deep Learning* que são mais adequados para o *dataset* em questão, conforme a complexidade dos dados, bem como analisar outros fatores como viés e variância que são discutidos na subseção 2.6.2 seguinte.

2.6.2 Viés e Variância

Modelos que apresentam muitas variações nas predições quando se troca o *dataset* utilizado na fase de treinamento e teste, demonstram indícios de que a capacidade de generalização foi comprometida pelo problema de variância que é quando o modelo é sensível à mudanças bruscas nos *datasets* utilizados para a preparação do modelo (RASCHKA, 2015). O desempenho de um modelo pode ser inferior ou abaixo do esperado quando as premissas (ou *assumptions*) realizadas sobre a complexidade ou distribuição (estatística) dos dados em um *dataset* não condizem com a realidade (RASCHKA, 2015).

Os problemas de viés (ou *bias*) e variância influenciam as situações de *underfitting* e *overfitting*, respectivamente (RASCHKA, 2015). Um resumo sobre viés e variância pode ser visto na Figura 16.

Figura 16 – Viés e Variância



Fonte: Software Livre Brasil (BREANDAN, 2019).

Na Figura 16 observa-se que quanto mais baixa for a variância dos dados e quanto mais baixo for o viés, mais acurado e mais preciso será o modelo construído. De outro modo, quanto mais alta for a variância dos dados e mais alto o viés sobre a complexidade destes, menos acurado será o modelo final.

2.7 Revisão Sistemática da Literatura

Segundo Nakagawa et al. (2017), uma revisão sistemática é uma forma de identificar, avaliar e interpretar um conjunto de pesquisas científicas relevantes para uma determinada questão de pesquisa ou tópicos específicos de uma área de pesquisa ou algum fenômeno de interesse.

Seguindo uma visão alternativa, Kitchenham e Charters (2007) definem que os estudos originais que contribuem para a elaboração de uma revisão sistemática são denominados estudos primários, enquanto que a revisão sistemática em si mesma é considerada um estudo secundário.

Espera-se que a revisão sistemática ajude a identificar o referencial teórico necessário para responder as principais questões de pesquisa desta dissertação (vide Quadro 1.2) e confirmar ou refutar as hipóteses consideradas neste estudo (vide seção 1.3).

Outras definições importantes sobre a revisão sistemática incluem a abordagem proposta por Kitchenham e Charters (2007) cujos principais elementos são:

- Protocolo de Revisão – especifica a principal questão de pesquisa a ser estudada e os métodos que serão utilizados para realizar a revisão.
- Estratégia de Busca – que tem o objetivo de descobrir a maior quantidade possível de estudos relevantes na literatura.
- Documentação da Estratégia de Busca – possibilita que outros pesquisadores possam avaliar o rigor, completude e reprodutibilidade do processo (tendo em mente as possíveis restrições ou particularidades encontradas em certas bibliotecas digitais que podem impedir a replicação do cenário de pesquisa).
- Critérios de Inclusão e Exclusão – determinam explicitamente quais são os requisitos para avaliar se um dado estudo primário deverá ser incluído ou não na revisão.
- Resultado Esperado – define a informação a ser obtida a partir dos estudos primários e os critérios de qualidade aplicados.

Durante a realização desta pesquisa foi necessário adaptar ambas as visões apresentadas por Nakagawa et al. (2017) e Kitchenham e Charters (2007) sobre a revisão sistemática para incorporar aspectos específicos desta dissertação, além de complementar o levantamento bibliográfico realizado com artigos e trabalhos científicos adicionais que foram descobertos após a revisão sistemática. O Apêndice A contém as principais definições básicos sobre Revisão Sistemática da Literatura e detalha o Protocolo de Pesquisa utilizado neste estudo.

2.8 Estado da Arte

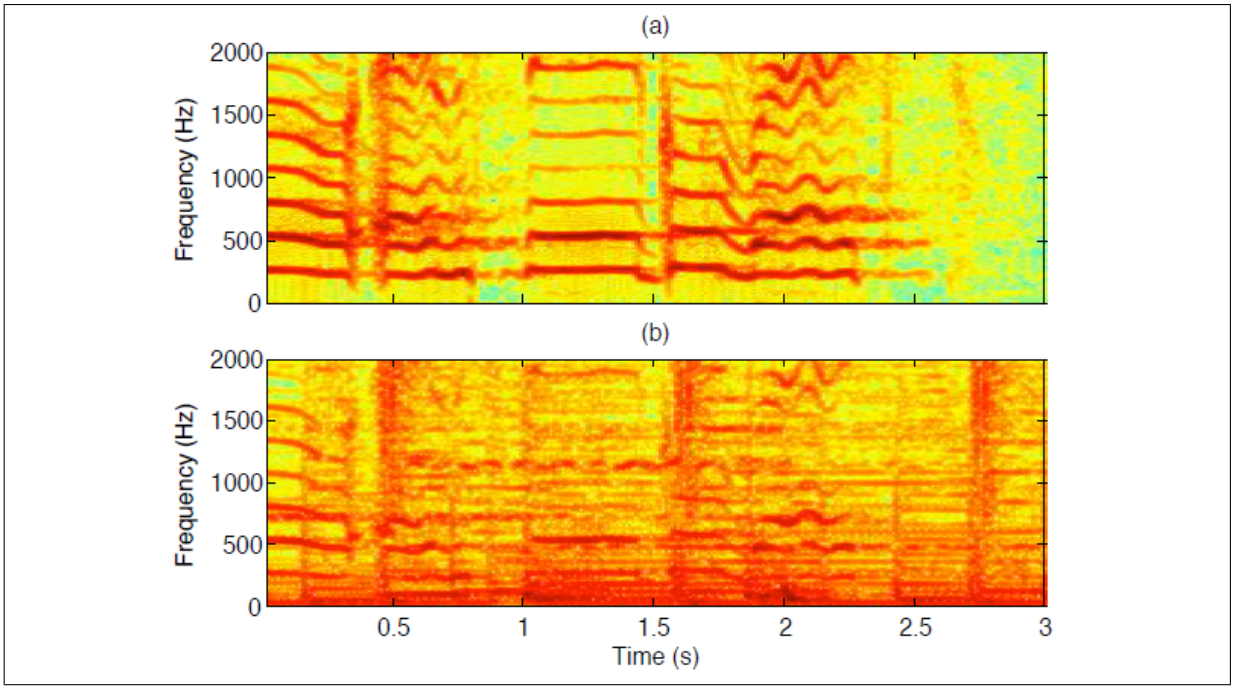
Esta seção discute o estado da arte em relação a tarefas de extração de melodia e transcrição melódica que norteiam este trabalho de dissertação, abrangendo as técnicas correntes em *Music Information Retrieval*, *Machine Learning* e *Deep Learning*.

2.8.1 Extração de Melodia

Em sua tese de Doutorado sobre Extração de Melodia, Salamon (2013b) destaca a importância da melodia como um fenômeno que faz parte da experiência humana através da música. É ressaltado que a habilidade humana de reconhecer as sequências de notas musicais que formam uma melodia possui alta complexidade quando comparada à capacidade dos sistemas computacionais que são voltados para processamento de música, cuja tarefa de reconhecimento de melodias não ocorre de maneira trivial.

Uma importante contribuição apresentada por Salamon (2013b) em sua tese está na abordagem utilizada para realizar a extração automática de melodia diretamente de um sinal de áudio polifônico, isto é, um sinal de áudio bruto que possui vários sons de instrumentos musicais diferentes acompanhando uma voz humana cantada dentro de um dado intervalo de tempo (SALAMON, 2013b, p. 10). Este problema é ilustrado na Figura 17 através de um espectrograma de áudio que foi extraído com base em Transformadas de Fourier do tipo STFT (vide a Equação 2.1).

Figura 17 – Espectrograma de um sinal de áudio polifônico



Fonte: Salamon (2013b, p. 12).

A Figura 17 ilustra um espectrograma com a presença de conteúdos harmônicos em um sinal de áudio polifônico, sendo que as características deste tipo de sinal aumentam

a complexidade da extração melódica. Na Figura 17 (a) é mostrado um sinal melódico enquanto a Figura 17 (b) mostra este mesmo sinal misturado a outros sons produzidos pelo acompanhamento de instrumentos musicais (SALAMON, 2013b).

Outros tipos de sinais de áudio mais simples que contém melodias também são abordados no estudo feito por Salamon (2013b). É o caso dos sons monofônicos e homofônicos, que são definidos como um som musical formado apenas por uma linha melódica única sem acompanhamento no primeiro caso, ou um som musical composto por uma linha melódica principal e acompanhamento de acordes, no segundo caso (SALAMON, 2013b).

Os desafios mais importantes que são encontrados na extração de melodia em sons polifônicos estão justamente no fato deste tipo de áudio conter vários sons distintos ao mesmo tempo, como na Figura 17 (b), que possui uma voz humana cantada acompanhada dos sons de vários instrumentos musicais. Assim, a principal dificuldade está em isolar ou separar cada componente de áudio e a linha melódica principal ou predominante (SALAMON, 2013b).

As transformadas discretas de Fourier (similar à versão simplificada da Equação 2.1) nos permite analisar o espectro de um sinal de áudio complexo (isto é, formado por várias frequências de som), ou seja, o espectro do áudio revela as diferentes frequências que o compõe (SERRA, 1989).

É importante ressaltar que as *features* de áudio utilizadas por Salamon (2013b) na tarefa de extração de melodias, por exemplo, *Spectral Peaks*, *Harmonic Pitch-Class Profile* e *f0* também aparecem no trabalho de Mitrović, Zeppelzauer e Breiteneder (2010) que discutem o processo de seleção das *features* de áudio mais comuns na literatura.

Em um dos testes realizados por Salamon (2013b) e no estudo feito por Poliner e Ellis (2005), O algoritmo *Support Vector Machine (SVM)* foi utilizado como classificador das notas musicais extraídas da melodia principal de um som e comparados com outros experimentos que utilizaram algoritmos baseados em *Árvores de Decisão*. Alguns conceitos básicos sobre SVM e *Árvores de Decisão* podem ser consultados no Apêndice B.

Salamon (2013b) sumarizou os principais aspectos de 16 algoritmos desenvolvidos entre os anos de 2005 a 2012 que representavam o estado da arte à época para a tarefa de Extração de Melodia. Os algoritmos analisados podem ser consultados na Figura 79 do Anexo B (o idioma original foi mantido para evitar problemas de tradução do inglês para português).

A contribuição mais importante de Salamon (2013b) foi uma nova teoria para extração

automática de melodias em sons polifônicos, chamada de *Salience Function*, que foi considerada inovadora e devido à sua alta acurácia ainda nos dias atuais é considerada como estado da arte em relação à abordagem utilizada mesmo que já existam outros algoritmos competitivos com resultados de acurácia mais alta, como por exemplo, empregando novas técnicas baseadas em *Deep Learning*.

2.8.2 Solfejo

Em música, uma técnica tradicional para aprender a cantar uma nota na altura correta é conhecida como *solfejo*. Na sua forma mais simples, o *solfejo* permite que uma pessoa consiga usar sua voz para criar ou reproduzir melodias de acordo com ritmos, durações dos sons e compassos musicais específicos. No entanto, mesmo em práticas de *solfejo* mais simples, a classificação automática da altura das notas cantadas é uma tarefa considerada complexa para um computador realizar (SCHRAMM, 2015a).

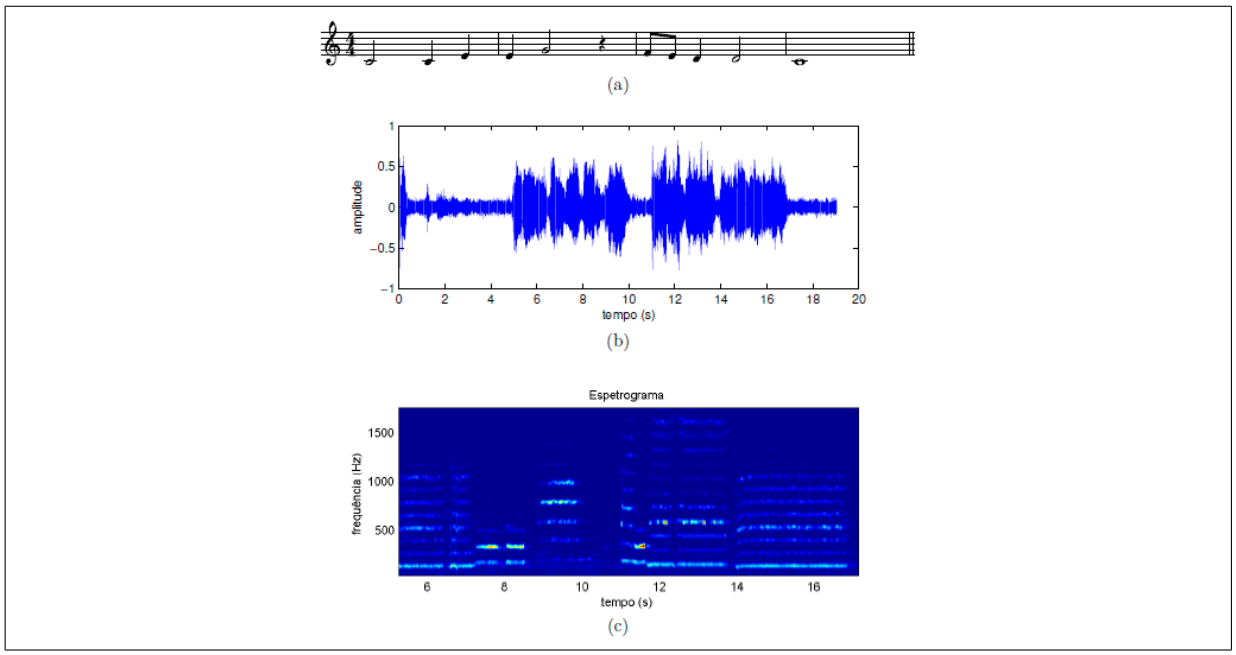
Segundo Gerhard (2002), o solfejo é um caso particular de um sinal homofônico onde a principal característica é a presença de uma voz humana cantada que pode ou não estar acompanhada de instrumentos musicais (a expressão *solo a acappella* também é utilizada para indicar o canto sem acompanhamento de instrumentos musicais). Em seu estudo, Gerhard (2002) demonstrou que *features* de áudio baseadas no *domínio perceptivo* tais como *pitch* e *chroma* são importantes discriminantes para detectar as diferenças entre a voz humana falada e a voz humana cantada.

Considerando apenas os aspectos melódicos do *solfejo*, conforme tratados em Schramm (2015a), a Figura 18 (a) mostra uma partitura resultante do exercício de *solfejo* com as respectivas amplitudes registradas no *Domínio Temporal* à partir dos dados brutos do áudio que aparece na Figura 18 (b) e o espectrograma equivalente na Figura 18 (c).

As *features* de áudio e técnicas utilizadas por Schramm (2015a) também compartilham os mesmos princípios vistos em Mitrović, Zeppelzauer e Breiteneder (2010) e Salamon (2013b) citando, por exemplo, *pitch*, *chroma*, *f0*, aplicação de *filter banks* e outras técnicas baseadas em Transformadas Discretas de Fourier.

Na Figura 19, pode-se verificar os resultados da análise de um trecho de áudio contendo um exercício de *solfejo*. A análise foi feita à partir do uso de *chroma features* (SCHRAMM, 2015a).

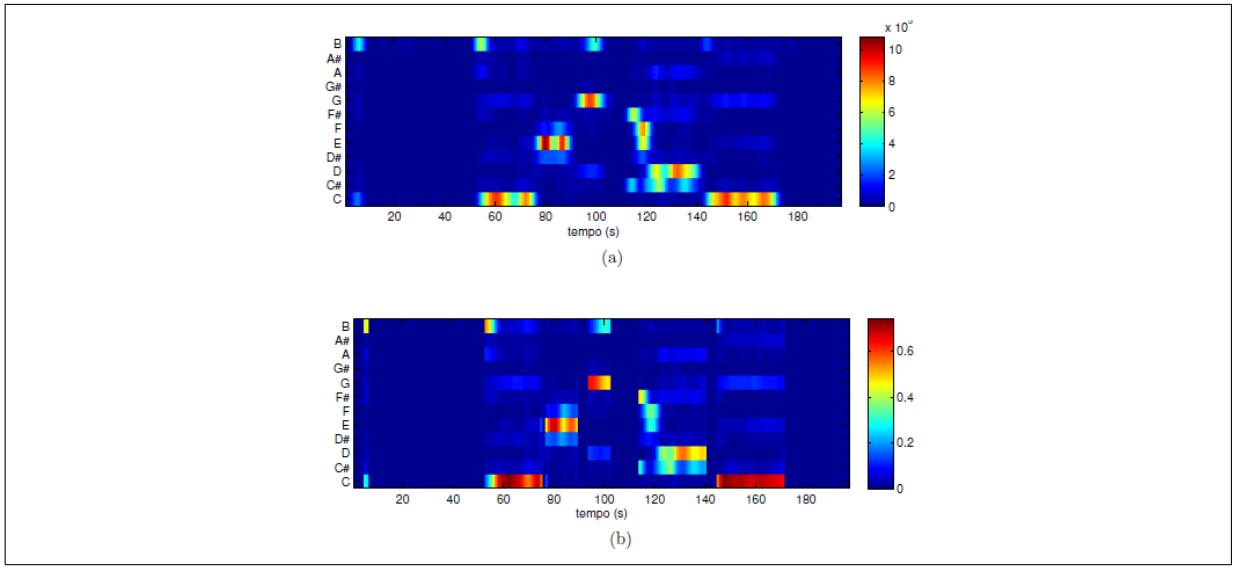
Figura 18 – Partitura, Sinal de Áudio e Espectrograma de um Solfejo



Fonte: Schramm (2015a).

O conjunto de *features* de áudio representadas por *chromas* que aparecem na Figura 19 é conhecido como *Chromagrama* e mostra as alturas e intensidades das notas detectadas. A Figura 19 (a) mostra o *Chromagrama* sem normalização enquanto a Figura 19 (b) exibe o mesmo *Chromagrama* com normalização do sinal de áudio (SCHRAMM, 2015a).

Figura 19 – Chromagrama sem Normalização e com Normalização



Fonte: Schramm (2015a).

O *Sistema Audiovisual para Solfejo* criado por Schramm (2015a) e defendido em sua

tese representa o estado da arte em atividades de avaliação automática de exercícios de *solfejo*, não sendo encontrado outros trabalhos na literatura que utilizem uma abordagem similar, isto é, que considere simultaneamente as imagens captadas em vídeos sincronizados com o som da voz para a correta avaliação do exercício de solfejo realizado por um aluno de canto.

2.8.3 Algoritmo pYIN para estimativa da Frequência Fundamental (*F0*)

Um fator importante observado nos trabalhos citados na subseção 2.8.1 e na subseção 2.8.2 é que ambos os pesquisadores utilizaram técnicas baseadas em versões adaptadas do algoritmo YIN desenvolvido por Cheveigné e Kawahara (2002) para a extrair a *feature f0* que representa a frequência fundamental do som.

Sendo assim, uma variante do algoritmo original YIN desenvolvida por Mauch e Dixon (2014), chamada de **pYIN** é frequentemente utilizada como uma alternativa moderna do algoritmo YIN original por utilizar múltiplos parâmetros de detecção dos *pitches* candidatos da frequência fundamental enquanto o YIN utiliza apenas um parâmetro fixo.

Existe uma implementação do algoritmo pYIN na forma de *plugin* VAMP que está disponível no *software* livre *Sonic Visualiser* (MARY, 2012) e que é comparável ao *plugin* MELODIA desenvolvido por Salamon (2013b), disponível no mesmo formato VAMP.

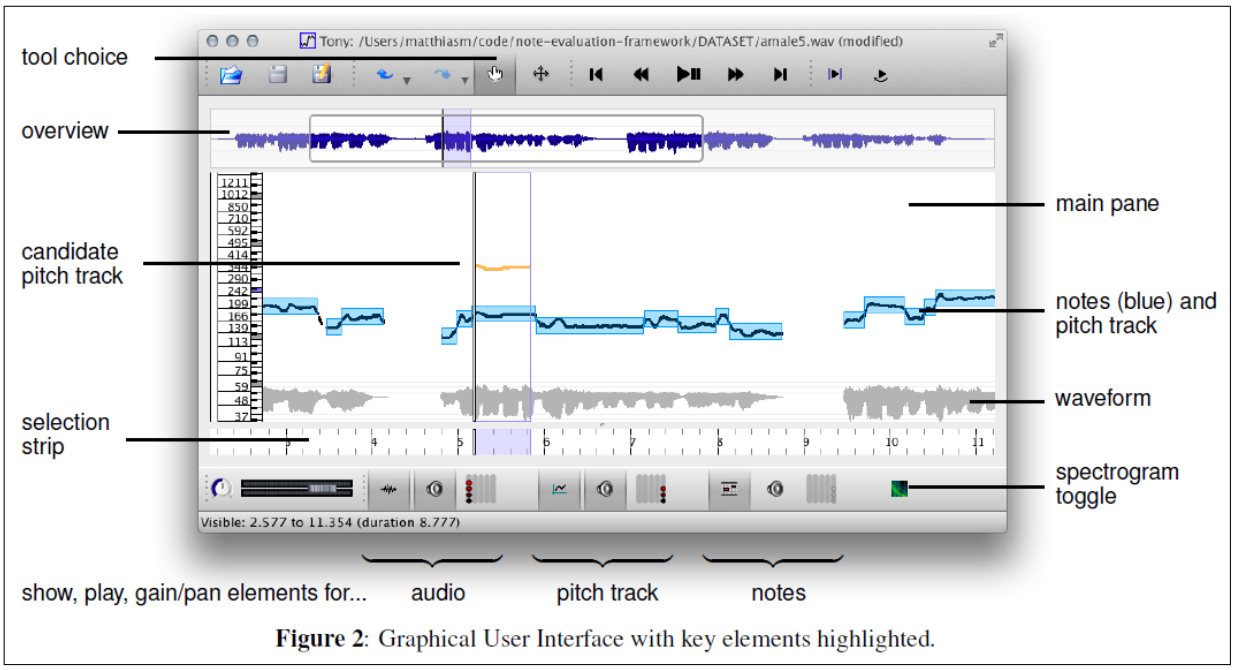
Esta mesma implementação do algoritmo pYIN foi incorporado na ferramenta *Tony* que também é um *software* específico para análise de áudio no contexto de *Music Information Retrieval* e Processamento de Sinais de Áudio (MAUCH; DIXON, 2014). Um exemplo da interface gráfica da ferramenta científica *Tony* é mostrado na Figura 20.

É possível observar que a frequência fundamental ou *pitch* candidato é destacado nas *curvas* (cor preta) com as respectivas notas musicais associadas (retângulos na cor azul). Facilitando a interpretação sobre quais sequências de *pitches* candidatos compõe uma determinada nota musical.

Segundo Mauch e Dixon (2014), o pYIN possui $85,00\%$ de acurácia, que é variável dependendo dos parâmetros (heurística) escolhidos para configuração do algoritmo que é especializado em tarefas de *transcrição melódica* de forma automática para gravações de *áudios monofônicos* (como os solfejos analisados nesta dissertação). Ainda assim, em alguns casos é necessário realizar ajustes manuais das notas musicais identificadas.

O pYIN foi avaliado, em sua maior parte, em experimentos realizados com o *dataset*

Figura 20 – Ferramenta *Tony* para Análise de Áudio



Fonte: Mauch e Dixon (2014).

ISMIR2014 criado por Molina et al. (2014) e por experimentos não publicados utilizando o *dataset MedleyDB* criado por Bittner et al. (2014).

Nesta dissertação, o algoritmo pYIN será utilizado como referência na tarefa de transcrição melódica e será utilizado para gerar os *labels* (ou rótulos) das melodias extraídas à partir dos exercícios de solfejo que serão utilizados para treinar as redes LSTM desenvolvidas no experimento.

2.8.4 Extração automática de *features* em música com *Deep Learning*

A extração de *features* de música utilizando técnicas de *Deep Learning* é um tema recente na literatura. Segundo LeCun, Bengio e Hinton (2015), o uso de *Deep Learning* através de Redes Neurais Profundas do tipo *FeedForward* recebeu mais atenção por volta do ano de 2006 devido ao esforço de um grupo de pesquisa canadense conhecido como *Canadian Institute for Advanced Research (CIFAR)*.

Os pesquisadores do CIFAR introduziram o uso de Aprendizado Não-Supervisionado na criação de camadas de redes neurais capazes de atuar como detectores automáticos de *features*, dispensando o uso de dados previamente rotulados. O principal objetivo foi permitir que essa nova arquitetura de redes neurais pudesse extrair *features* diretamente

de dados brutos (LECUN; BENGIO; HINTON, 2015).

Posteriormente, as camadas de redes neurais especializadas na detecção automática de *features* passaram a ser utilizadas para realizar o *pré-treinamento* de redes neurais profundas mais complexas. Esta foi a técnica predominante, por exemplo, para tarefas de reconhecimento de caracteres escritos à mão ou detecção de pedestres passando em uma faixa de trânsito (LECUN; BENGIO; HINTON, 2015).

Mas o uso de Redes Neurais Profundas mais complexas, ou seja, com várias camadas e milhares de parâmetros para serem treinados exigem um alto pode computacional que pode ser impeditivo em alguns casos. No entanto, com o advento das Placas de Aceleração Gráficas (GPUs) que ajudam a acelerar o treinamento destas redes, foi possível a utilização desta abordagem de *pré-treinamento* para resolver problemas ligados ao reconhecimento de fala (campo de pesquisa que possui algumas similidades com as tarefas de processamento de áudio em músicas) (LECUN; BENGIO; HINTON, 2015).

Segundo LeCun, Bengio e Hinton (2015) o uso de GPUs permitiu que os pesquisadores alcançassem um tempo de treinamento de redes neurais cerca de 10 a 20 vezes mais rápido em comparação aos métodos que não utilizavam GPUs. No ano de 2009, esta abordagem de treinamento de redes neurais com o uso de GPUs permitiu criar um modelo probabilístico capaz de prever fragmentos de fala à partir da análise de quadros (*frames*) de som extraídos usando uma janela de tempo (com uma dada duração em milissegundos) para extrair os coeficientes de Fourier associados a uma onda sonora.

À partir de então, os resultados obtidos pelo pesquisadores Mohamed et al (2012) e Dahl et al (2012), citados por LeCun, Bengio e Hinton (2015), estabeleceram um novo marco de desempenho (*benchmark*) na área de pesquisas sobre reconhecimento de fala.

Em 2012, a técnica de *pré-treinamento* com as Redes Neurais Profundas começou a ser utilizada para ajudar a reduzir os riscos de *overfitting* em *datasets* considerados pequenos, melhorando significativamente a capacidade de generalização dos modelos quando a quantidade de dados rotulados disponíveis eram insuficientes para uma dada tarefa (LECUN; BENGIO; HINTON, 2015).

Passado este período histórico do *Deep Learning*, principalmente em relação às Redes Neurais Profundas do tipo *FeedForward*, outras arquiteturas de Redes Neurais Profundas ganharam notoriedade. Este é o caso das Redes Neurais Convolucionais.

Em estudos mais recentes desenvolvidos por Bittner et al. (2017), cujos autores também integram a mesma equipe de pesquisadores de Salamon (2013b), foi demonstrado

um experimento que usa Redes Neurais Convolucionais para extrair *features* de música automaticamente.

De maneira similar, Costa, Oliveira e Silla Jr (2017) também utilizaram Redes Neurais Convolucionais em seu experimento para realização de classificação automática de música à partir de espectrogramas.

Ambos os trabalhos de Bittner et al. (2017) e Costa, Oliveira e Silla Jr (2017) evidenciaram resultados considerados como estado da arte quando da extração automática de *features* em música utilizando Redes Neurais Convolucionais.

Outro exemplo de uso de Redes Neurais Convolucionais é a extração de *features* musicais para a realização de tarefas de classificação de música de acordo com o gênero musical conforme o estudo conduzido por Senac et al. (2017).

No caso da equipe *Brain Team* da Google (GOOGLE, 2019a), os experimentos diretamente relacionados ao tema desta dissertação existem como parte do Projeto de Pesquisa *open source* conhecido como *Magenta* (GOOGLE, 2019b).

Destacam-se os experimentos denominados *Onsets and Frames* (HAWTHORNE et al., 2018) que realiza a transcrição de áudios de piano para arquivos de música em formato *MIDI* (Musical Instrumental Digital Interface), *MusicVAE* (GOOGLE, 2019d) que provê um modelo capaz de representar as características musicais de um som em uma dimensionalidade reduzida (conhecida como espaço latente) e permitir o controle criativo dos parâmetros aprendidos para a criação de novos sons e, por último, o *NSynth* (GOOGLE, 2019f), que é um modelo de Rede Neural Profunda criada para sintetizar sons a partir de um subconjunto de amostras individuais de áudio, trazendo como principal inovação a capacidade de manipular os timbres sonoros extraídos com base em uma Rede Neural preexistente chamada *WaveNet* (OORD et al., 2016) que é um modelo generativo voltado para criação de sons em sua forma pura (*waveforms*). Estes modelos são tão complexos que, por exemplo, o treinamento de uma rede do tipo *NSynth* pode consumir até 10 dias de processamento distribuído em computadores de alto desempenho equipados com até 32 Placas de Aceleração Gráfica (GPUs) especiais para treinamento de Redes Neurais Profundas (GOOGLE, 2019e).

Seguindo a tendência de uso de *Deep Learning* para processamento de áudio, outra equipe da Google formada por Hasim Sak, Andrew Senior e Françoise Beaufays, propuseram uma abordagem para realizar modelagem acústica utilizando Redes Neurais Recorrentes. Nesta abordagem, eles demonstraram que Redes Neurais Recorrentes do tipo

Long Short-Term Memory (LSTM) são mais eficientes do que Redes Neurais Profundas do tipo *Feedforward* convencionais (SAK; SENIOR; BEAUFAYS, 2014).

O principal propósito da equipe foi avaliar o desempenho do modelo treinado com Redes do tipo LSTM para tarefas de reconhecimento de fala. Porém, o projeto *Magenta* da Google possui vários experimentos musicais práticos que empregam Redes Neurais Recorrentes, incluindo redes do tipo LSTM, mostrando que é possível adaptar estes modelos para tarefas relacionadas a música.

Outro experimento do projeto *Magenta* da Google que possui afinidade com o tema desta dissertação é o *Melody RNN*. Este modelo foi originalmente construído para a área linguística, mas foi adaptado para gerar melodias utilizando uma Rede LSTM (GOOGLE, 2019c).

Existem alguns estudos como os de Hipke et al. (2014) e Stowell (2010), que abordam o uso de *Machine Learning* e *Deep Learning* para tarefas de transcrição melódica nas quais a voz humana é mapeada para o som de um instrumento musical real.

Não foram encontrados experimentos específicos de transcrição melódica de solfejos para sons com timbres de instrumentos musicais reais, apesar das novas técnicas de *Deep Learning* indicar benefícios para o campo de pesquisa que abrange o tema de transcrição melódica e extração automática de *features* em música. No entanto, devido a algumas similaridades e constatações encontradas no estudo de Rigaud e Radenen (2016), seu trabalho sobre transcrição de melodias para *voz humana cantada* será discutido brevemente na subseção 2.8.5.

2.8.5 *Voice Activity Detection* (VAD) e *F0 Estimation* com *Deep Learning*

O algoritmo pYIN apresentado na subseção 2.8.3 e usado por Schramm (2015a), ou ainda, o algoritmo MELODIA desenvolvido por Salamon (2013b) dependem da configuração de um conjunto de parâmetros que determinam quais heurísticas devem ser aplicadas durante a análise de um áudio.

Para se chegar às configurações ideais, estes tipos de algoritmos requerem conhecimento especializado em processamento de sinais de áudio e a principal vantagem que poderá ser alcançada com a aplicação de *Deep Learning* é a diminuição de parte desta complexidade.

No estudo desenvolvido por Rigaud e Radenen (2016), é feita a proposta de duas

Redes Neurais Profundas (ou *Deep Neural Networks* - DNN) operando em paralelo nas tarefas de *detecção de atividade de voz* (ou *Voice Activity Detection* - VAD) e *estimativa da frequência fundamental* (ou *F0 Estimation*).

Em MIR, a tarefa VAD é capaz de determinar se existe a presença de voz humana em um trecho de áudio que contenha melodia. Por sua vez, a atividade de *F0 Estimation* determina quais são as frequências sonoras ou *pitches* presentes em cada trecho do áudio analisado.

Uma rede neural recorrente do tipo *Bidirectional Long-Short Term Memory* foi utilizada para implementar a tarefa VAD. Esta rede continha 3 camadas BiLSTM com 50 neurônios cada uma e 1 camada de Rede Neural do tipo *FeedForward* com 1 neurônio na saída e ativação *sigmóide*, sendo que esta configuração foi escolhida arbitrariamente após tentativas e erros (RIGAUD; RADENEN, 2016). A Figura 21 mostra a configuração resultante e os dados de entrada e saída da rede adotada para a tarefa VAD.

Os dados de entrada são formados pela concatenação de 3 tipos diferentes de representação do mesmo sinal de áudio que foi decomposto em espectrogramas (STFT) na escala logarítmica sendo que p_1 contém os componentes melódicos e percussivos do áudio e h_1 e h_2 contém os componentes harmônicos e restante dos instrumentos estáveis do áudio analisado.

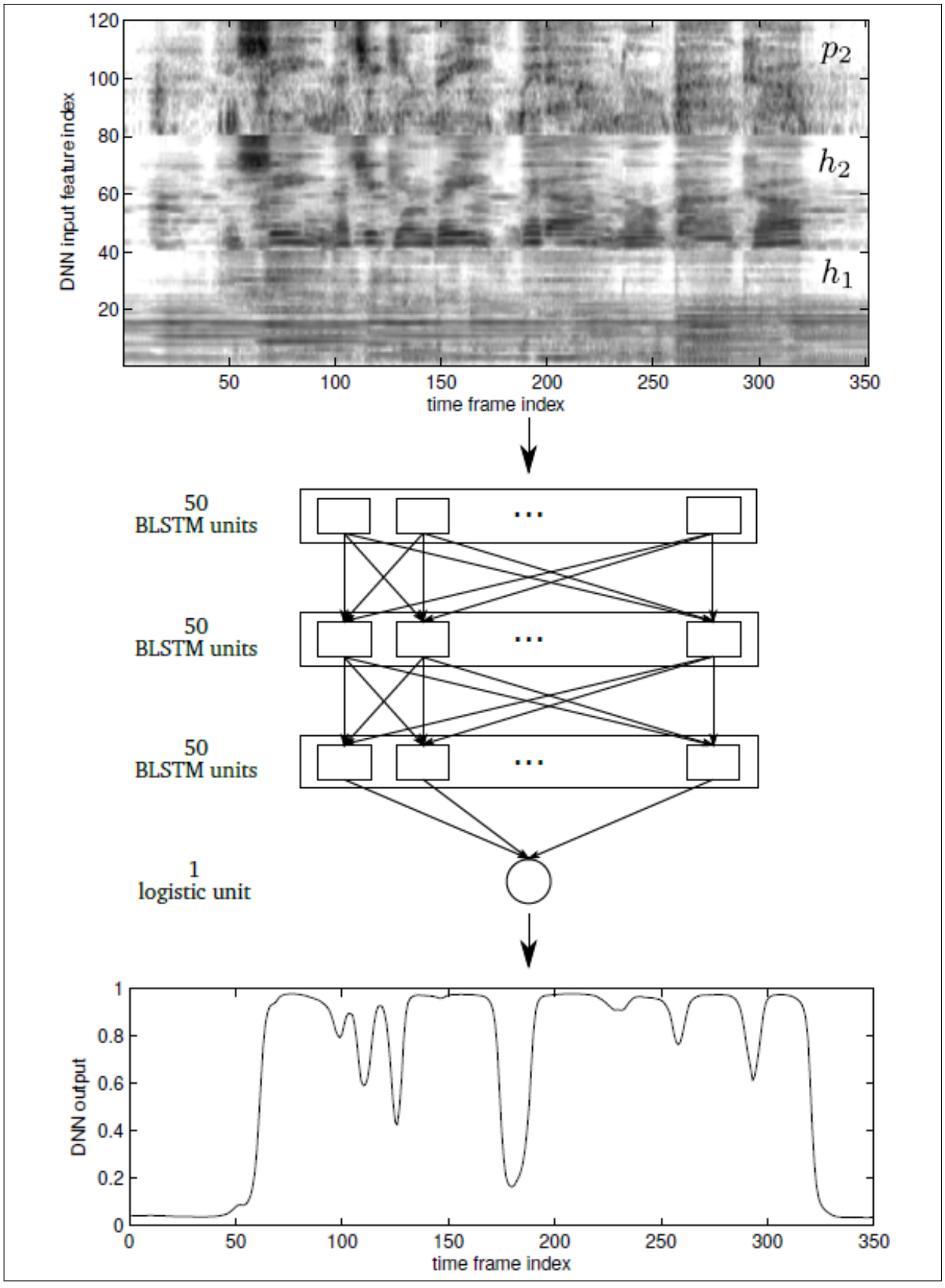
A saída da rede é um sinal contendo os picos de energia em cada *frame* de áudio, o qual é pós-processado com um *threshold* configurado com o valor 0,5 para classificar se existe ou não atividade de voz humana em um dado intervalo de tempo.

Para a tarefa de *F0 Estimation* foi utilizada a DNN que aparece na Figura 22. Nessa rede também foram utilizados espectrogramas (STFT) em escala logarítimica padronizada para valores no intervalo entre [0..1]. Como esta rede é independente da rede VAD, todos os áudios que não possuíam trechos com atividade de voz foram removidos do conjunto de treino e validação. As frequências *f0* detectadas estão na faixa das notas musicais $C\#2$ a $C\#6$.

Segundo Rigaud e Radenen (2016), após vários experimentos realizados foi possível estabelecer uma arquitetura de rede funcional. Foram avaliadas DNN contendo neurônios logísticos típicos de uma Rede Neural do tipo *FeedForward* e neurônios recorrentes utilizados em Redes Neurais do tipo *Bidirectional Long Short-Term Memory* (BLSTM).

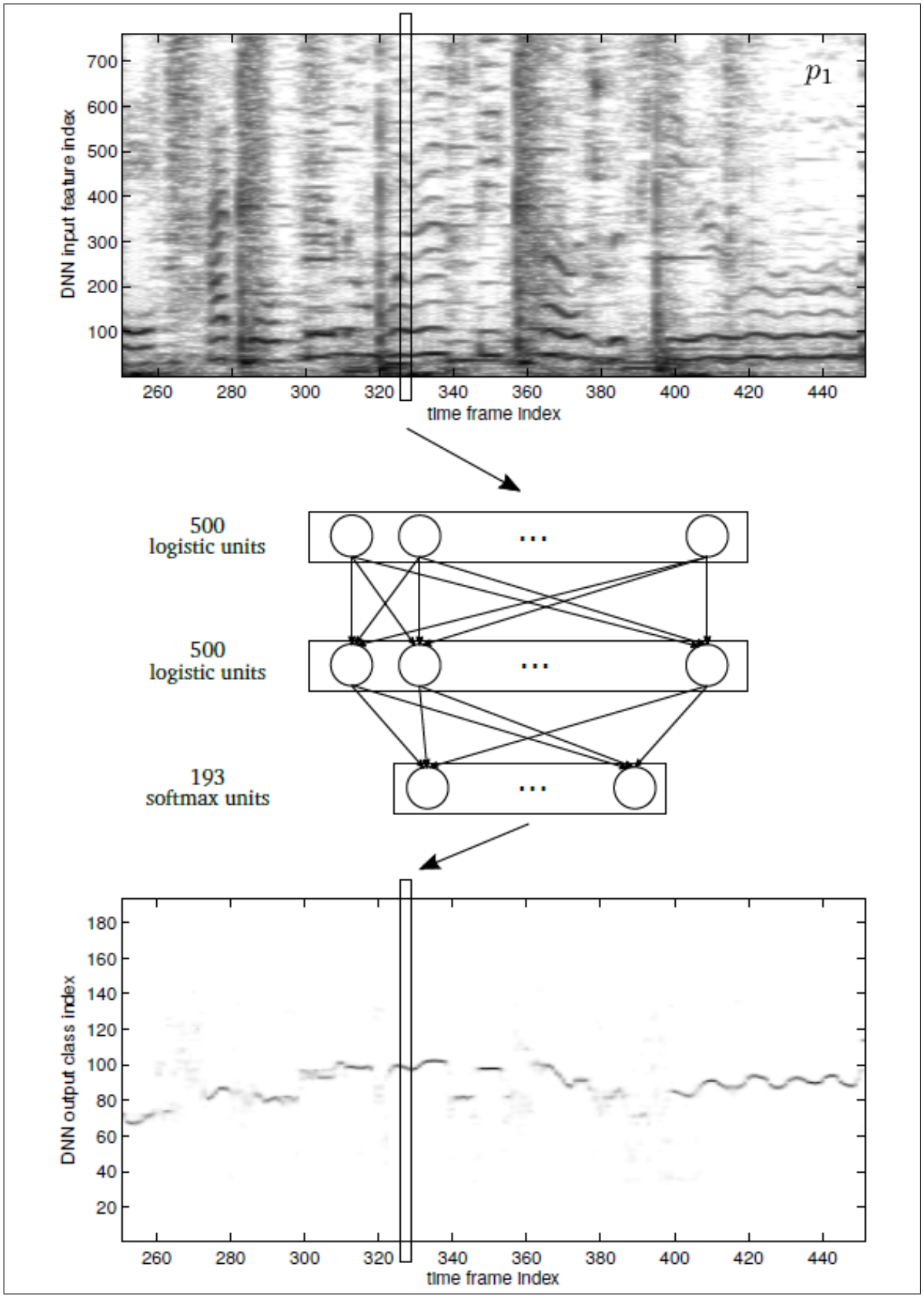
Nos testes realizados, os pesos da rede foram inicializados de forma aleatória de acordo com uma distribuição Gaussiana com média 0 (zero) e desvio padrão de 0,1 usando *loss*

Figura 21 – DNN para *Voice Activity Detection*



Fonte: Rigaud e Radenen (2016).

Figura 22 – DNN para *F0 Estimation*



Fonte: Rigaud e Radenen (2016).

function do tipo *cross-entropy* otimizada com *stochastic gradient descent*. Para evitar *overfitting* durante o treino, foi utilizado *mini-batches* aleatoriamente embaralhados juntamente com o uso da técnica de *early stopping*.

A técnica denominada *early stopping* consiste em parar o treino após um limite pré-estabelecido de épocas quando nenhum aprendizado de pesos é verificado após um número de épocas consecutivas, que neste caso foi estabelecido o limite de 100 épocas, considerando um total de 10.000 épocas para o treino completo.

Dos sinais de áudio utilizados como entrada, o p_1 foi o que obteve melhor desempenho nos testes com uma DNN do tipo *FeedForward* de 2 camadas cada uma com 500 neurônios configurados com *ativação sigmóide*. A camada de saída da rede utilizou uma função softmax para multiclassificação de 193 categorias de frequências sonoras compreendidas na faixa das notas musicais $C\#2$ a $C\#6$.

Apesar da continuidade temporal da *feature f0* ser melhor representada pelos *times-teps* da rede BiLSTM utilizada, esta opção foi descartada por apresentar problemas de *overfitting*. Então, os autores do estudo optaram por manter a rede do tipo *FeedForward* mesmo perdendo a característica de interpretabilidade da rede BiLSTM.

No final dos testes realizados por Rigaud e Radenen (2016), concluiu-se que a DNN elaborada alcançou acurácia de $82,48\%$ em comparação ao algoritmo MELODIA desenvolvido por Salamon (2013b) que registrou $62,13\%$ em relação à métrica *Raw Pitch Accuracy* (conforme definida na seção 4.3) no *dataset MedleyDB*. A biblioteca **mir_eval** (RAFFEL et al., 2014) foi utilizada para a extração das métricas.

Dentre os *datasets* usados para o treinamento das redes que realizam as tarefas de VAD e *F0 Estimation*, vale destacar que o *dataset MedleyDB* criado por Bittner et al. (2014) é o mesmo utilizado no experimento desta dissertação.

Considerando que o estudo de Rigaud e Radenen (2016) também fez uso de outro *dataset* denominado *iKala*, a acurácia geral medida para a tarefa de *F0 Estimation* foi de $85,06\%$ para o *iKala* e $75,03\%$ para o *MedleyDB*. Como o sistema desenvolvido é restrito à transcrição melódica da voz, alguns aspectos e métricas não puderam ser comparados com outros estudos do *estado da arte* porque abordam a transcrição de melodia de músicas do ponto de vista genérico.

Esta subseção encerra a revisão da literatura sobre o tema, onde também foi discutido um exemplo de aplicação de *Deep Learning* para transcrição de melodia comparável a outros algoritmos encontrados na área de MIR que são considerados *estado da arte*.

A presente dissertação utilizou um total de 79 *Referências Bibliográficas* para ser elaborada, dentre as quais, os artigos e demais trabalhos científicos que possuem maior relevância para a proposta do experimento e seus vínculos com o tema são resumidos no Quadro 3.

Quadro 3 – Principais trabalhos científicos e contribuição para esta Dissertação

Referência Bibliográfica	**Contribuição**
Bittner et al. (2017)	Fornece o modelo de CNN usado como ***Encoder***
Brownlee (2017)	Fornece a teoria do modelo CNN-LSTM
Géron (2017)	Descreve como implementar e treinar redes CNN e LSTM
Wilkins et al. (2018)	Fornece o *dataset* denominado *VocalSet* que é especializado em exercícios de solfejo
Hawthorne et al. (2018)	Apresenta o modelo CNN-BiLSTM para transcrição de Piano, usando a LSTM como ***Decoder***
Rigaud e Radenen (2016)	Apresenta modelos de Redes Neurais Profundas para transcrição de melodias em voz humana cantada
Mauch et al. (2015)	Apresenta o algoritmo pYIN e a ferramenta *Tony* usados como base para comparação do experimento
Tsiporkova (2006)	Descreve o funcionamento do *Dynamic Time Warping*
Meert et al. (2019)	Fornece uma implementação do DTW
Raffel et al. (2014)	Fornece métricas para avaliação de algoritmos em MIR
McFee et al. (2015)	Fornece implementações de STFT e CQT

Fonte: Elaborado pelo autor.

3 PROPOSTA DE EXTRAÇÃO AUTOMÁTICA DE *FEATURES* DE ÁUDIO COM REDES CNN E LSTM

Nesta seção será apresentada uma visão geral da proposta de uso combinado de redes CNN e LSTM como ***encoder*** e ***decoder***, respectivamente, em tarefas de extração de melodia e sintetização de sons de instrumentos musicais a partir de exercícios de solfejo.

A entrada de dados para a rede CNN–LSTM é feita a partir de arquivos de áudio bruto em formato WAVE (16 *bits* de resolução com 22.050 *Hz* de taxa de amostragem) que contém exercícios de solfejo, conforme mostrado na Figura 23 (d).

O áudio com solfejo, ilustrado na Figura 23 (d), está no *domínio temporal* e é necessário converter esta representação para o *domínio da frequência* (*time-frequency*) usando uma transformação baseada em *Constant-Q Transform* (CQT) definida por Bittner et al. (2017) como *Harmonic* CQT, conforme Figura 23 (c).

Com o áudio representado no *domínio da frequência*, os dados são repassados para a camada de entrada da rede CNN–Melody, ilustrada na Figura 23 (b), que atuará simultaneamente como *salience function* (para extrair a melodia) e camada de ***encoder*** para a rede LSTM–CQT.

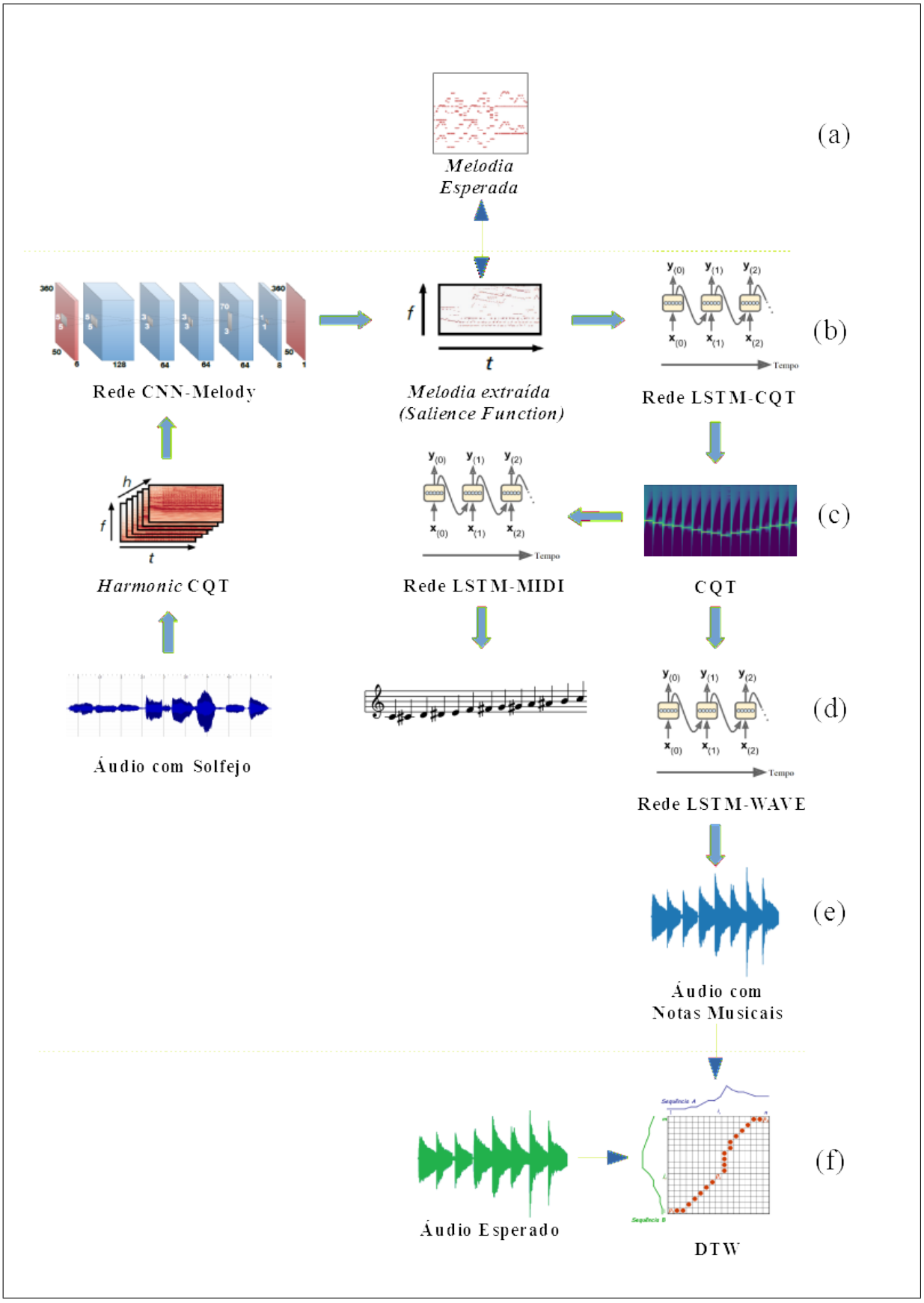
Durante o retreinamento supervisionado da rede CNN–Melody (que desconsidera qualquer aprendizado feito anteriormente), a *feature f0* extraída do áudio representa a melodia que será comparada com a *feature f0* da "*melodia esperada*", mostrada na Figura 23 (a).

A *feature f0* (frequência fundamental) da melodia e estrutura harmônica do áudio contendo solfejo será extraída pela rede CNN–Melody e repassada para a rede LSTM–CQT, ilustrada na Figura 23 (b).

A rede LSTM–CQT tem o papel de ***decoder*** e é responsável pela representação de sons com timbres de instrumentos musicais reais (ou similares) no *domínio da frequência* (com CQT) que são sintetizados para áudio bruto (*domínio temporal*) pela rede LSTM–WAVE exibida na Figura 23 (d), além de possibilitar a classificação das notas musicais (no intervalo *C4* a *D5*) pela rede LSTM–MIDI mostrada na Figura 23 (c).

O áudio sintetizado é ilustrado na Figura 23 (e) e contém as notas musicais com timbres de instrumento musical real (ou similar). Este áudio é comparado com o *"áudio esperado"* através do algoritmo *Dynamic Time Warping*, ilustrado na Figura 23 (f), que funciona como medida de similaridade entre as sequências de sons. Isto permite realizar uma avaliação complementar do desempenho da rede LSTM-WAVE.

Figura 23 – Proposta de uso combinado de redes CNN e LSTM



Fonte: Elaborado pelo autor com figuras próprias e outras adaptadas de Bittner et al. (2017), Géron (2017) e Tsiporkova (2006).

3.1 Uso de CNN como *Salience Function*

A motivação em usar a rede CNN para tarefas de extração automática de *features* em música surgiu devido ao sucesso de modelos amplamente conhecidos na literatura que utilizam CNN, como é o exemplo do modelo *Wavenet* desenvolvido por Oord et al. (2016) que é capaz de modelar a voz humana e músicas.

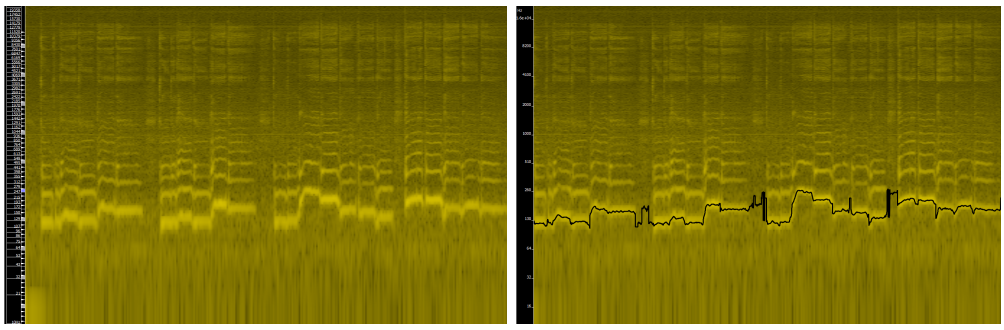
Outros estudos relevantes que incluem o uso de CNN para tarefas relacionadas a música são aqueles discutidos nos trabalhos de Senac et al. (2017) e Costa, Oliveira e Silla Jr (2017), ambos para classificação de gênero musical.

A partir daí, foi realizada uma pesquisa na literatura para identificar estudos que utilizassem CNN especificamente em tarefas de extração de *features* de melodia em solfejos.

Neste caso, o estudo mais relevante para esta dissertação é o que foi conduzido por Bittner et al. (2017), pois permite sustentar a proposta de adaptar uma rede CNN para extração automática de melodias em exercícios de solfejo.

Então, para entender como uma rede CNN pode ser aplicada no caso do solfejo, é útil analisar o exemplo mostrado na Figura 24 que é a representação do som de um solfejo no *domínio da frequência* obtida pela transformação STFT (também conhecida como *espectrograma*).

Figura 24 – Espectrograma de uma canção popular



(a) (b)

Fonte: Elaborado pelo autor utilizando o plugin *Melodia* criado por Salamon (2013b).

A canção *"Happy Birthday to You"*[1] inspirou o exercício de solfejo do espectrograma exibido na Figura 24 (a). A respectiva saliência com a melodia principal (linha sólida na

[1] Com o propósito acadêmico de analisar um espectrograma de uma canção popular conhecida sem infringir nenhum direito autoral, uma versão adaptada da melodia americana *"Happy Birthday to You"* é exibida na Figura 24. De acordo com King (2015), a melodia da canção

cor preta) é mostrada na Figura 24 (b) usando o método de *salience function* criado por Salamon (2013b) com as técnicas usuais de *feature engineering* e algoritmos de *Machine Learning*.

Considerando o estudo desenvolvido por Bittner et al. (2017), a principal função da rede CNN–Melody é atuar como *salience function*, extraindo a melodia de solfejos, além de servir como ***encoder*** para a rede LSTM.

O experimento proposto nesta dissertação apresenta o uso combinado de uma rede CNN com uma rede LSTM como alternativa em relação aos métodos tradicionais que empregam *feature engineering* e algoritmos de *Machine Learning* clássicos, pois esta rede é capaz de realizar a extração automática de *features* de melodias em músicas e, por isso, tem chances de apresentar resultados satisfatórios quando aplicada em exercícios de solfejo.

Sendo assim, as duas representações de áudio escolhidas para o caso do solfejo e a utilização de redes do tipo CNN são a CQT e a STFT. Ambas serão testadas separadamente para fornecer uma base de comparação com os resultados dos modelos elaborados por Schramm (2015a) e Salamon (2013b), respectivamente.

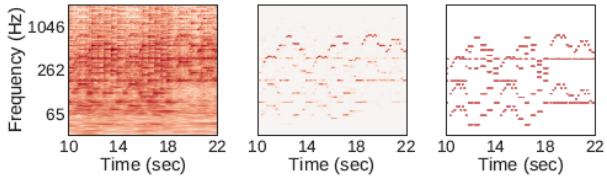
A Figura 25 (a) mostra a entrada de uma amostra de áudio, *à esquerda*, usando a representação denominada *harmonic*-CQT. *No centro* está a saída da predição da rede CNN–Melody após o aprendizado dos pesos dos filtros nas camadas convolucionais que irão produzir uma representação similar à saliência original que aparece *à direita* desta mesma figura.

No experimento proposto nesta dissertação, a arquitetura mostrada na Figura 25 (b) será utilizada como modelo de referência (BITTNER et al., 2017). Destaca-se o fato de que a rede CNN–Melody utilizada não possui camadas de *pooling*. A finalidade é evitar que a Rede CNN–Melody ignore pequenas variações ou deslocamentos nas frequências dos sons no domínio do *tempo-frequência* (BITTNER et al., 2017).

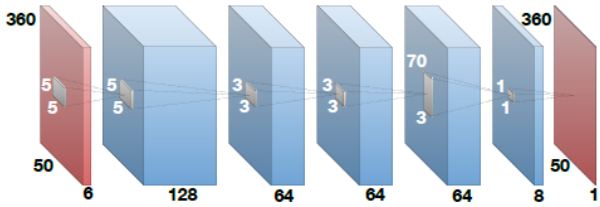
O modelo de referência para a CNN–Melody segue uma arquitetura de rede *fully convolutional* que possui 6 camadas de convolução. As duas primeiras camadas têm 128 e 64 filtros respectivamente, cada uma com *filter kernel* de tamanho 5×5. A terceira e quarta camada de convolução possuem 64 filtros cada uma com tamanho 3×3. A quinta camada de convolução tem 8 filtros de tamanho 70×3 e a última camada possui 1 filtro

"Happy Birthday to You" foi confirmada como sendo de *domínio público* em 22 de setembro de 2015 pelo juíz **George H. King** FJC (2015).

Figura 25 – Rede CNN–Melody usada como *Salience Function*



(a)



(b)

Fonte: Adaptado de Bittner et al. (2017).

com tamanho 1×1. Em cada camada, as operações de convolução ocorrem com *zero-padding* para manter tamanhos equivalentes entre o *shape* de entrada e o *shape* de saída da rede CNN (BITTNER et al., 2017).

Cada camada recebe dados normalizados via *batch normalization* e as saídas de cada camada passam por uma unidade de ativação baseada na função RELU (BITTNER et al., 2017) conforme exibido na Figura 39 da seção 4.4.

A última camada do modelo de referência utiliza uma ativação baseada na *função logística* e a saída da rede pode ser interpretada como uma *feature* de áudio contendo o contorno de *f0* estimado, isto é, similar à nota musical correspondente a cada *quadro* do *domínio de frequência* (BITTNER et al., 2017).

3.2 Uso de LSTM para Sintetização de Sons

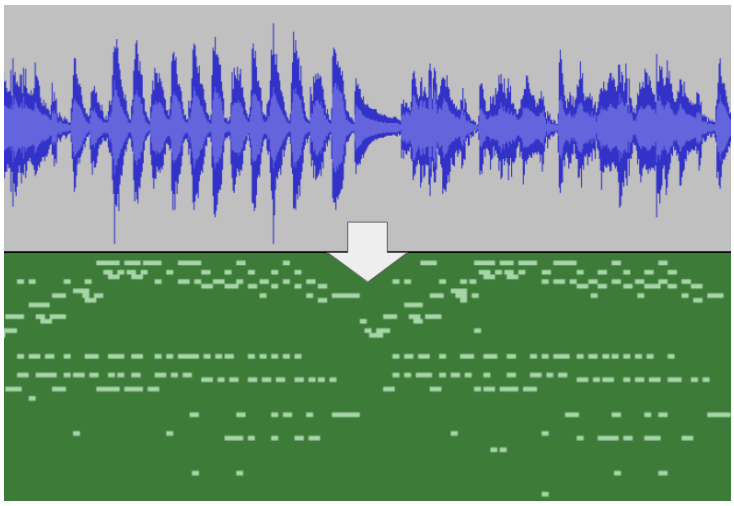
O projeto *Magenta* da Google oferece vários modelos baseados em redes neurais recorrentes do tipo LSTM que são capazes de auxiliar em tarefas de composição musical. Dentre os vários sub-projetos que compõe o *Magenta*, aquele que tem especial afinidade com a proposta desta dissertação é chamado de *Onsets and Frames*.

O modelo *Onsets and Frames* desenvolvido por Hawthorne et al. (2018) é especializado

na tarefa de transcrição automática de sons musicais polifônicos, o que pode ser uma vantagem no caso de solfejos, mesmo que os exercícios de solfejo sejam predominantemente monofônicos.

Os resultados satisfatórios do modelo *Onsets and Frames* servem de motivação para investigar o uso de redes LSTM na modelagem de música. A Figura 26 apresenta uma imagem ilustrativa da capacidade do modelo *Onsets and Frames* em converter um áudio gravado com música de piano para o formato MIDI (*piano roll*).

Figura 26 – Transcrição automática de música de Piano com LSTM



Fonte: Adaptado de Hawthorne et al. (2018).

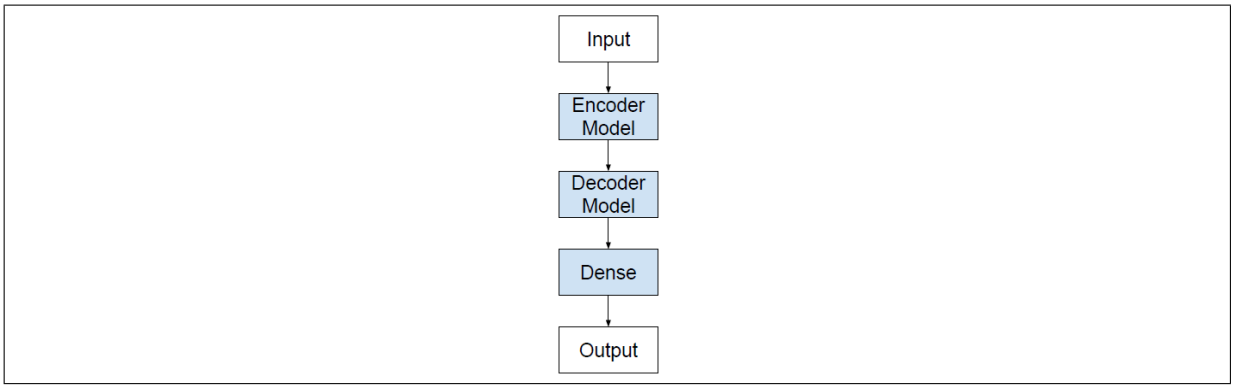
Na Figura 27 temos um modelo de Rede LSTM mais simples de 3 camadas baseada na arquitetura chamada LSTM Encoder-Decoder sugerida por Brownlee (2017, p. 108) que foi utilizado para os testes iniciais e durante o experimento poderá evoluir gradualmente para uma arquitetura mais complexa similar à apresentada por Hawthorne et al. (2018).

A rede LSTM Encoder-Decoder faz uma redução de dimensionalidade dos dados e descobre uma representação intermediária para os dados de áudio analisados. Após isto, reconstrói esta representação compacta (também chamada de *espaço latente*) através da camada de *decoder* da rede LSTM Encoder-Decoder, que aparece *no centro* da Figura 27, logo abaixo da camada de *encoder* da rede LSTM Encoder-Decoder.

Os dados reconstruídos pela camada de *decoder* da rede LSTM Encoder-Decoder são repassados para uma Rede do tipo *Feed Forward fully connected*, que aparece na Figura 27 como *"Dense"*.

Nesta dissertação, o áudio sintetizado é obtido pela camada de saída (*Output*) da rede

Figura 27 – Rede LSTM com Arquitetura Encoder-Decoder



Fonte: Brownlee (2017).

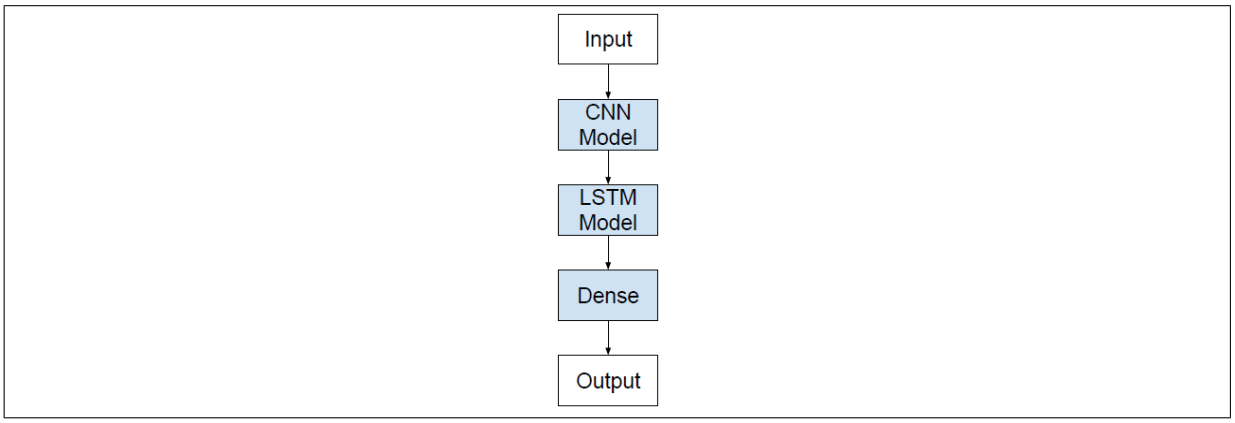
LSTM Encoder-Decoder e possui o timbre sonoro de um instrumento musical real similar ao piano.

3.3 Combinação das Redes CNN e LSTM

Conforme exemplificado na Figura 23, as *features* de áudio extraídas automaticamente à partir da rede CNN–Melody usada como *salience function* e ***encoder*** são repassadas para uma rede LSTM que atuará como um ***decoder*** responsável por representar as notas musicais com timbres correspondentes a um instrumento musical real similar ao piano.

O diagrama esquemático mostrado na Figura 28 ilustra uma das possibilidades de combinação das Redes CNN e LSTM conforme apresentada por Brownlee (2017, p. 93).

Figura 28 – Rede LSTM com Arquitetura CNN–LSTM



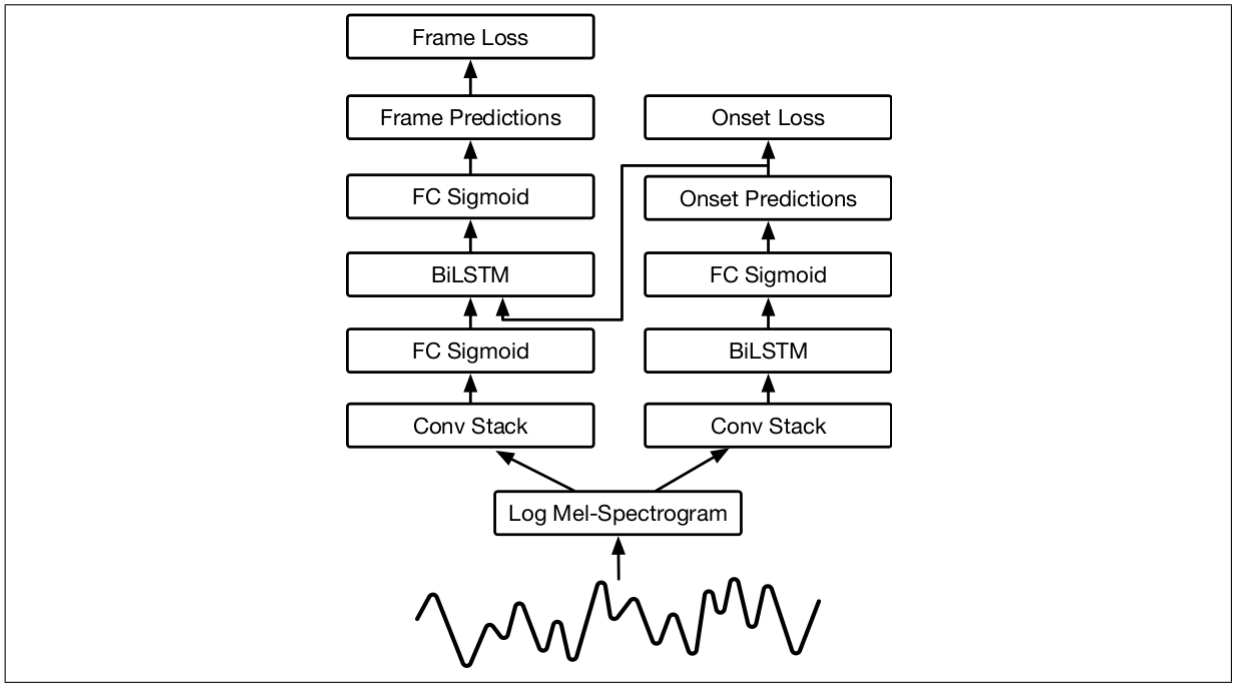
Fonte: Brownlee (2017).

Segundo Brownlee (2017, p. 93), o funcionamento básico da Arquitetura CNN–LSTM

consiste em utilizar uma rede CNN para extração de *features*, reduzir as dimensões do *feature vector* obtido para um vetor unidimensional e combinar o resultado obtido com uma rede LSTM. No final da rede CNN–LSTM existe uma camada de Rede Neural *Feedforward* (*Dense*) que pode ser utilizada para tarefas de classificação ou regressão.

No experimento desta dissertação, é criado um modelo influenciado pelo trabalho de Hawthorne et al. (2018), ilustrado na Figura 29, combinado com a rede CNN–Melody proposta por Bittner et al. (2017) (representada como *"Conv Stack"* na Figura 29). Desse modo, é feito a extração automática da melodia seguida da transcrição automática das notas musicais identificadas no exercício de solfejo para notas com timbres correspondentes a um instrumento musical real.

Figura 29 – Rede LSTM do Modelo *Onset and Frames*



Fonte: Hawthorne et al. (2018).

O principal diferencial do Modelo *Onset and Frames* é a capacidade de detectar os eventos de início (*onsets*) e fim de uma nota musical (*offsets*) em concordância com a predição de *frames* de áudio. Na Figura 29 é possível verificar que existem duas Redes CNN–LSTM em paralelo que são interligadas de modo que as predições realizadas pela sub-rede denominada *Frames* está condicionada pela sub-rede denominada *Onsets*.

3.4 *Datasets* para Solfejo

O *dataset* denominado *Solfège30* é formado por 7 arquivos de áudio/vídeo contendo práticas de exercícios de solfejo realizados por estudantes de música em diferentes velocidades e modalidades de canto. Os exercícios foram baseados em trechos de partituras próprias criadas com o *software* livre *MuseScore* (MUSESCORE, 2020) e possuem cerca de 110 amostras rotuladas, embora possam conter erros de anotação (SCHRAMM, 2015b).

Não foi possível obter o *dataset* denominado *DATASET_1* com 3276 exemplos de notas rotuladas (SCHRAMM, 2015a). Por isso, foi utilizado o *Solfège30* e as principais *features* de áudio encontradas no *Solfège30* são listadas no Quadro 4.

Quadro 4 – Principais *features* de áudio do *dataset Solfège30*

Features	**Descrição**
Pitch	Contendo a nota musical aproximada
Onset	Intervalo de tempo com o ataque da nota musical
Offset	Intervalo de tempo que a nota deixa de soar

Fonte: Elaborado pelo autor.

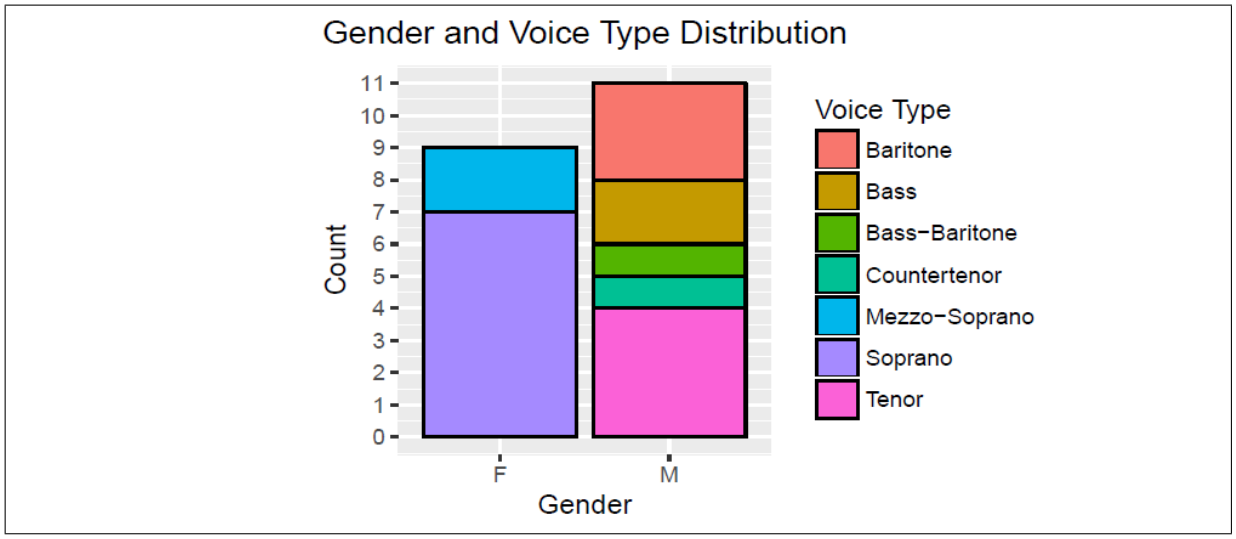
O *dataset* musical conhecido como *MedleyDB* (versão com 122 músicas, total aproximado de 7,5 horas de áudio) foi disponibilizado por Bittner et al. (2014) e será utilizado para retreinar o modelo CNN–Melody demonstrado em Bittner et al. (2017) para ajustá-lo à presente proposta de experimento.

Por fim, o *dataset* musical *VocalSet* criado por Wilkins et al. (2018) é especializado em exercícios de Solfejo e contém 3.560 arquivos WAVE, totalizando 10,1 horas de áudios gravados por 20 cantores profissionais ao todo (com formação acadêmica em canto ou música), sendo 9 vozes femininas (identificadas com os códigos *f1* a *f9*) e 11 vozes masculinas (identificadas com os códigos *m1* a *m11*), conforme a Figura 30 que apresenta a distribuição dos cantores e respectivos tipos de voz.

Dentre os três *datasets* citados (*Solfège30*, *MedleyDB* e *VocalSet*), o *VocalSet* é o único estruturado hierarquicamente de acordo com variadas técnicas de canto e que oferece exercícios de solfejo com progressão de notas musicais em escala natural ascendente (*C3* a *D5*) e descendente (*D5* a *C3*) com vocalizações distintas para cada vogal (*a*, *e*, *i*, *o*, *u*), além de alguns exercícios com escalas musicais contendo sustenidos e bemóis (por exemplo, solfejos da cantora f2 com notas de *C#4* a *D#5*).

Para comprovar a utilidade do *VocalSet*, Wilkins et al. (2018) apresenta dois sistemas

Figura 30 – Classificação de Tipos de Voz presentes no *VocalSet*

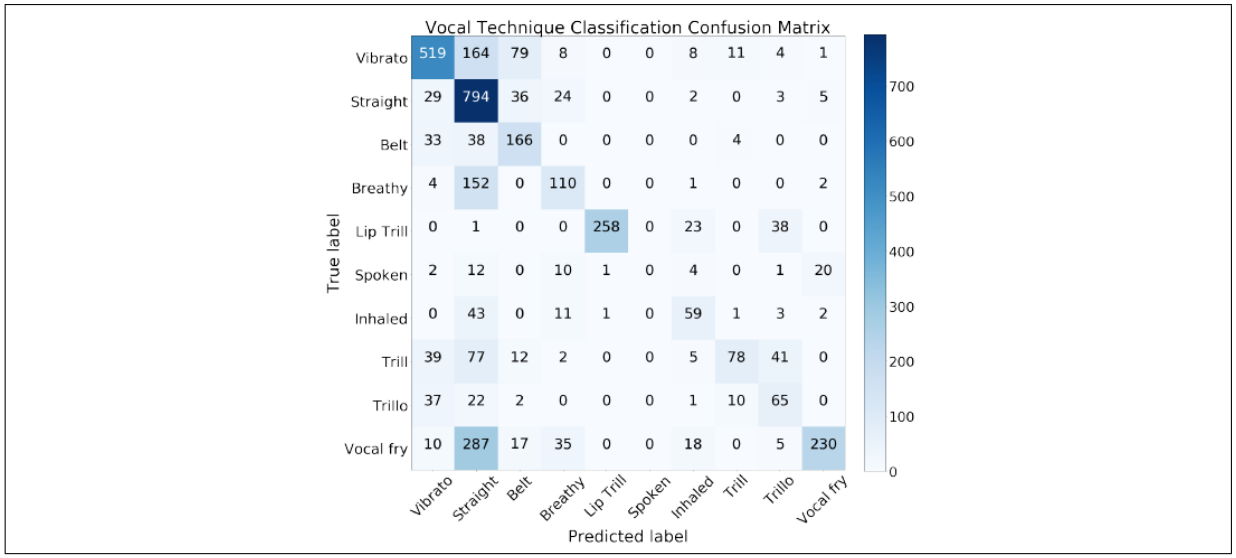


Fonte: Wilkins et al. (2018).

de classificação baseados em *Deep Learning*. Ambos utilizam uma rede CNN para as tarefas de *identificação de cantores* e *identificação de técnicas vocais*.

É importante observar que as técnicas conhecidas como *Vibrato* e *Straight* aparecem na Figura 31 como as classes mais frequentes na tarefa de *identificação de técnicas vocais*.

Figura 31 – Matriz de Confusão para Classificação de Técnicas Vocais de Canto

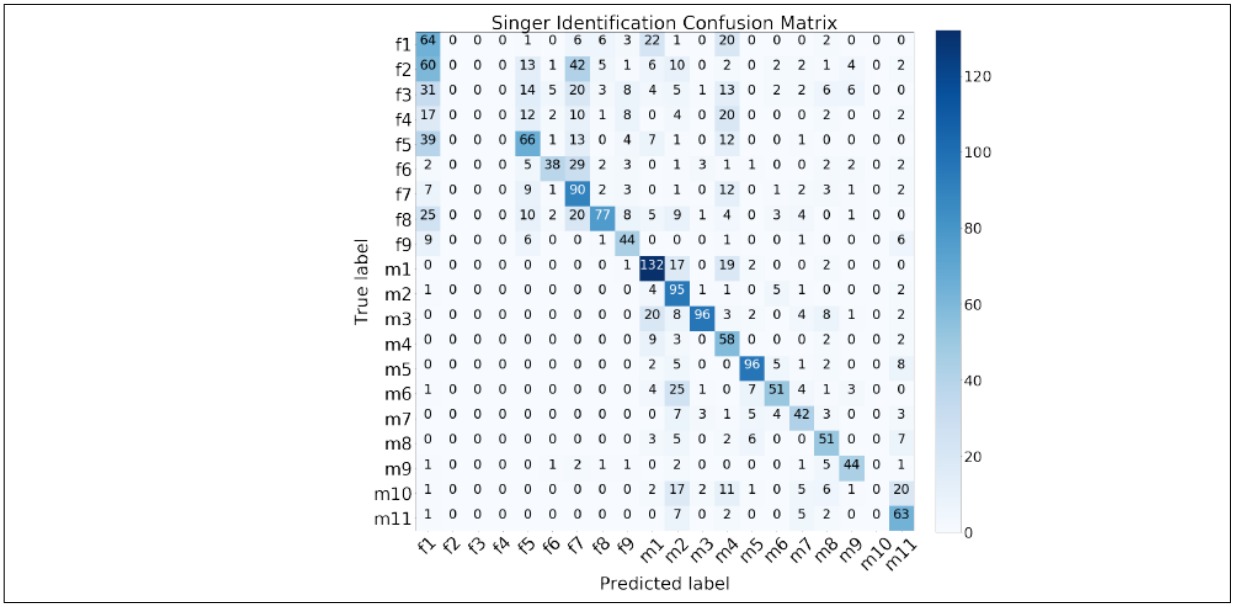


Fonte: Wilkins et al. (2018).

Este *desbalanceamento de classes* ocorre porque o *VocalSet* possui mais exemplos de áudio que utilizam as técnicas *Vibrato* (255 exemplos, total de 57,79 minutos) e *Straight* (361 exemplos, total de 71,65 minutos) em relação às demais técnicas vocais de canto.

O mesmo fenômeno de *desbalanceamento de classes* pode ser visto na tarefa de *identificação de cantores*, onde o número de classificações incorretas das vozes femininas superam os erros referentes às vozes masculinas que são maioria no *VocalSet*, conforme a Figura 32.

Figura 32 – Matriz de Confusão para Identificação de Cantores



Fonte: Wilkins et al. (2018).

Analisando-se o *desbalanceamento de classes* na tarefa de *identificação de cantores*, evidencia-se que existem baixas ocorrências de erros em relação à identificação de vozes femininas sendo confundidas com vozes masculinas, enquanto que o contrário é observado. Exemplificando, as vozes dos cantores *m1*, *m2* e *m4* foram confundidas com vozes femininas com maior frequência (44, 32 e 85 vezes, respectivamente) do que as vozes femininas *f1*, *f6* e *f7* (5, 1 e 2 vezes, respectivamente).

Considerando-se as peculiaridades do *VocalSet* e a complexidade envolvida para distinguir corretamente as vozes e as técnicas vocais utilizadas, acredita-se que também poderá existir um impacto na identificação das notas musicais executadas nos solfejos de cada cantor ou cantora. Por isso, apenas um subconjunto dos dados foram utilizados, compreendendo as 9 vozes femininas executando a técnica *Straight* para a vogal *a* (apenas uma amostra da vogal *u* foi utilizada para validação de solfejo da cantora *f2*).

Mais detalhes sobre o *VocalSet* com exemplos de espectrogramas e *técnicas vocais* citadas podem ser consultadas no Anexo C e no artigo escrito por Wilkins et al. (2018).

4 EXPERIMENTO

O treinamento das redes CNN e LSTM apresentadas na seção 3, ilustrado na Figura 23, é realizado em diferentes etapas para cada rede, aplicando-se o conceito de *transfer learning* que consiste em reutilizar os pesos previamente aprendidos por uma rede neural (ou camadas específicas desta) (GÉRON, 2017, p.287).

O processo se inicia à partir da rede CNN–Melody que, após treinada, tem seus pesos congelados para possibilitar o *transfer learning* sem que seja afetada pelo treinamento da rede LSTM–CQT que está acoplada em sua última camada.

O mesmo método de *transfer learning* é aplicado nas redes LSTM–CQT, LSTM–MIDI e LSTM–Wave. Portanto, as subseções seguintes apresentam os detalhes para implementação das redes que irão compor a Arquitetura CNN–LSTM conforme a proposta da seção 3, Figura 23.

4.1 Configuração do Ambiente Computacional para o Experimento

A linguagem de programação *Python* (versão 3.7) foi escolhida para a implementação do experimento em conjunto com bibliotecas científicas para *Machine Learning* e *Deep Learning* tais como o *scikit-learn* e *Keras+Tensorflow* (versão específica para rodar em GPU), respectivamente.

Para complementar os pacotes de *softwares* necessários para a parte de manipulação de áudio, matrizes e plotagem de gráficos foi utilizado bibliotecas como *Librosa* e *SciPy*, *Numpy+Pandas* e *Matplotlib*, respectivamente, em conjunto com *softwares* para edição de áudio, análise de espectrogramas e composição musical tais como *Audacity*, *Sonic Visualiser* (com o *plugin Melodia*) e *MuseScore*.

Os experimentos realizados podem ser executados em Sistemas Operacionais da família *Linux* (à partir do *kernel* com versão 4.15) e *Microsoft Windows 10* (qualquer versão) ambos com arquitetura de 64bits.

Em questões de *hardware*, um processador da família *Intel Core i5* de oitava geração com 2.90GHz de velocidade foi utilizado para o processamento do código em *Python*, 16GB de memória RAM DDR4 e uma placa aceleradora de vídeo da *Nvidia*, modelo *Geforce RTX 2070 Super* com *8GB* de memória dedicada (do tipo GDDR 6) que foi usada para treinamento das redes neurais profundas implementadas em *Keras+Tensorflow*.

4.2 Preparação dos Dados para Treinamento

O conjunto de amostras do *dataset MedleyDB*[2] disponibilizado por Bittner et al. (2014) foi pré-processado usando uma transformação para o domínio da frequência baseada em *Constant-Q Transform* (CQT) com reamostragem de 22.050 *Hz* e 16 *bits* de precisão antes de ser usado no retreinamento da rede CNN–Melody criada por Bittner et al. (2017).

O *dataset Solfège30* disponibilizado por Schramm (2015a) não foi utilizado durante o treinamento da rede CNN–Melody nem das redes LSTM, sendo mantido à parte como dados de teste. No caso da rede LSTM–CQT, foi necessário ajustar os arquivos-texto contendo as anotações dos *onsets* para que ficassem corretamente alinhados aos exercícios de solfejo gravados.

Os arquivos de áudio do *VocalSet* contendo as vozes femininas foram editados com o *software Audacity* para remover os silêncios e ruídos existentes no início e final das gravações de áudio, além de equalizar os volumes para +15dB. Em seguida, foi feito o pré-processamento dos áudios com a biblioteca *Librosa* para normalizar os dados entre os intervalos $[-1..1]$, realizar *downsampling* para 22.050 *Hz* com 16 *bits* de precisão e converter a representação do áudio do *domínio temporal* para o *domínio da frequência* (CQT) antes de serem fornecidos para as redes CNN–Melody e LSTM–CQT. Foram utilizados 360 *bins* de frequência, 6 oitavas, 60 *bins* para cada oitava, resolução de 20 *cents*, *hop length* de 256 *frames* e frequência mínima de 32.7 Hz (equivalente à nota C1).

Os sons com timbres de piano utilizados para o aprendizado da rede LSTM–WAVE foram sintetizados à partir de partituras elaboradas pelo autor com a ajuda dos *softwares MuseScore* e *Sonic Visualiser* que também são utilizados para exportar arquivos de áudio em formato MIDI à partir da referida partitura.

A maioria dos arquivos MIDI e WAVE exportados contém a escala natural de *C4* a *D5* gravadas em diferentes velocidades[3] (utilizando a *semínima* como unidade de tempo e fórmula de compasso 3/4): 92bpm, 111bpm, 191bpm, 129bpm, 112bpm, 97bpm e 157bpm pelas cantoras *f2*, *f3*, *f4*, *f5*, *f6*, *f7* e *f9*, respectivamente, que fazem parte do conjunto de treino. Os solfejos da cantora *f1* foram utilizados como conjunto de validação durante o treinamento das redes LSTM (quando não é usado *cross-validation*) e sua gravação possui

[2] O acesso ao *MedleyDB* foi concedido mediante solicitação aos autores originais em 18/12/2019 por meio de formulário eletrônico disponível em <https://zenodo.org/record/1649325>.

[3] A velocidade de execução de uma peça musical é medida em *Batidas por Minuto*, abreviado como **bpm**.

velocidade de 103bpm. Os solfejos da cantora *f8* possuem 130bpm e foram utilizados como conjunto de testes finais para avaliação dos modelos. O Quadro 5 relaciona as cantoras selecionadas, a respectiva partição no *dataset* e velocidade aproximada de execução dos solfejos.

Quadro 5 – Preparação dos conjuntos de Treino, Validação e Testes do *VocalSet*

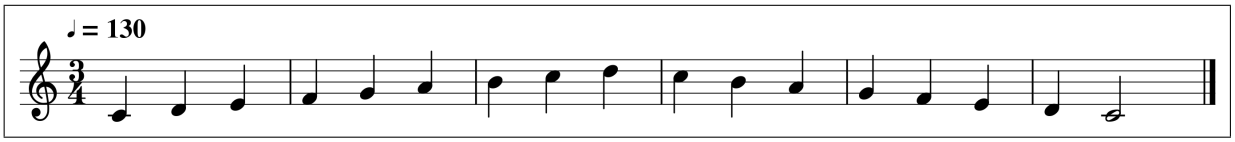
Cantora	**Partição de Dados**	**Velocidade do Solfejo**
f2	Treino	92bpm
f3	Treino	111bpm
f4	Treino	191bpm
f5	Treino	129bpm
f6	Treino	112bpm
f7	Treino	97bpm
f9	Treino	157bpm
f1	Validação	103bpm
f8	Teste	130bpm
f2 (“o”)	Teste	94bpm

Fonte: Elaborado pelo autor.

Os arquivos MIDI exportados do *Sonic Visualiser* (criados com o algoritmo **pYIN**) e editados no *MuseScore* foram utilizados para o treinamento da rede LSTM–MIDI apenas para classificar as notas musicais, ou seja, não foram gerados arquivos MIDI de saída. Isto foi necessário porque o *VocalSet* não fornece as anotações de áudio com *features* como *f0*, *Pitches*, *Onsets* e *Offsets*.

Segundo Raffel (2016), utilizar arquivos MIDI como *Ground Truth* é uma abordagem aceitável na área de *Music Information Retrieval* (MIR). Os *Onsets* foram utilizados para segmentar o áudio de entrada e saída das redes CNN e LSTM em *frames* que variam entre 30 a 55 *timesteps*, dependendo do solfejo que está sendo processado. Um exemplo de partitura criado no *MuseScore* para a cantora *f8* aparece na Figura 33.

Figura 33 – Exemplo de Partitura para a cantora *f8*



Fonte: Elaborado pelo autor com o *software MuseScore*.

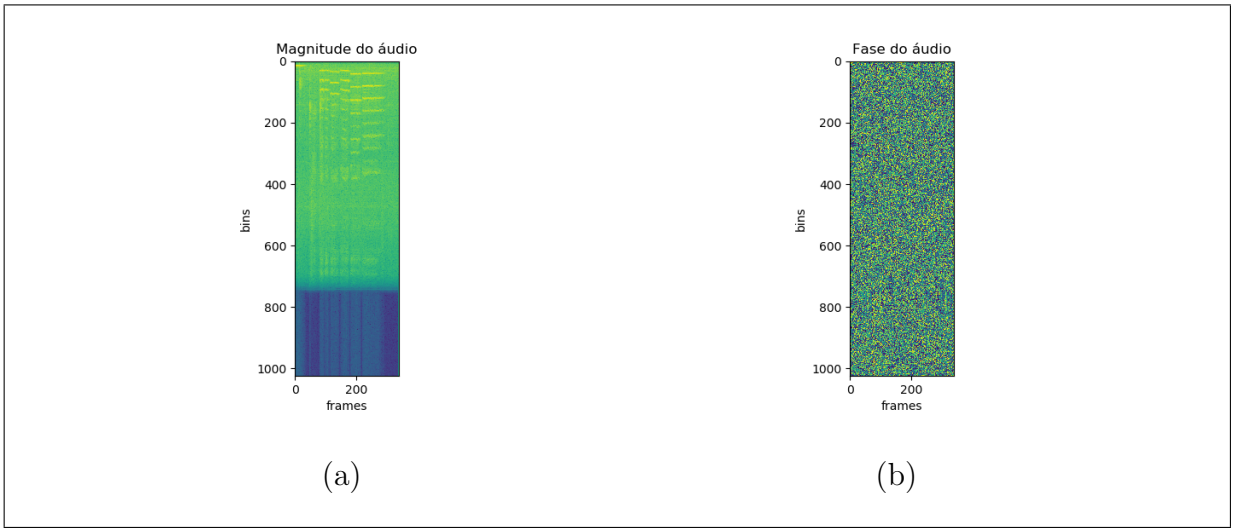
Os *bytes* dos áudios utilizados nas redes CNN–Melody e CNN–LSTM são armazenados em estruturas de dados conhecidas como *Matrizes* (ou *arrays* no *NumPy*) e *Tensores* (um

tipo especial de Matriz com n-dimensões no *Tensorflow*) quando manipulados pela CPU e GPU, respectivamente. Os dados de áudio possuem dimensões 2D (*bins*[4], *frames*[5]) após o pré-processamento.

Considerando o fato da transformada discreta de *Fourier* empregada nos métodos CQT e STFT devolver números complexos (conjunto $\mathbb{C}$) e a forma como os *bytes* são armazenados internamente foi necessário converter as representações em 2D do áudio armazenado nas matrizes e tensores para o formato 3D transposto (*canal*, *frames*, *bins*) com os números complexos sendo representados por um par de números reais (conjunto $\mathbb{R}$) para preservar o máximo de informações dos sinais de áudio em relação à *magnitude* e à *fase* das ondas sonoras e possibilitar a reconstrução do sinal ou *sintetização de áudio*.

Como exemplo, a Figura 34 (a) exibe a *magnitude* do áudio contido no exercício 01 do *dataset Solfège30* ao lado da respectiva *fase* do áudio que é exibida na Figura 34 (b). No entanto, devido à limitação existente nos *frameworks* de *Deep Learning* atuais para lidar diretamente com Números Complexos, principalmente a parte imaginária, outra estratégia foi adotada no experimento desta dissertação.

Figura 34 – Magnitude e Fase do áudio contido no exercício 01 do *dataset Solfège30*



Fonte: Elaborado pelo autor utilizando a biblioteca *Matplotlib*.

A preparação alternativa dos dados consistiu em substituir a representação em STFT para CQT e descartar a parte imaginária sem prejuízo para o experimento. Isto foi possível

[4] Os *bins* são frequências discretizadas obtidas pela transformada de *Fourier* que representam uma faixa de frequências contínuas.

[5] Os *frames* ou quadros de tempo são uma janela de tempo em milisegundos cujo eventos de áudio foram capturados.

porque a parte real da representação em CQT foi utilizada para análise das frequências, enquanto o áudio bruto em formato WAVE foi usado para a sintetização dos sons diretamente pela rede LSTM-Wave.

4.3 Critério de Validação do Experimento

As métricas de avaliação em *Machine Learning* são importantes para determinar a capacidade de generalização dos modelos utilizados na execução de uma dada tarefa quando novos dados são apresentados. É esperado que modelos treinados corretamente e com dados suficientes não sofram de problemas como *overfitting* ou *underfitting* conforme estudado na subseção 2.6.1.

Os critérios de validação serão baseados em métricas tradicionais utilizadas em algoritmos de *Machine Learning* e compreendem ***Acurácia***, ***Precision***, ***Recall*** e ***F1-Score*** dentre outras métricas específicas para o processamento de música em MIR que são extraídas com o apoio da biblioteca *mir_eval* desenvolvida por Raffel et al. (2014). As principais métricas usadas em tarefas de *extração de melodias* em MIR são:

- *Voice Recall Rate* - Computa a proporção de *frames* de áudio classificados corretamente por um algoritmo como sendo realmente *frames* que contém trechos de melodia.

- *Voicing False Alarm Rate* - Computa a proporção de *frames* de áudio identificados como trechos de áudio que contém melodias, quando na verdade não há melodias presentes naqueles *frames* de áudio analisados por um algoritmo.

- *Raw Pitch Accuracy* - Computa a proporção de *frames* de áudio contendo melodias que tiveram as frequências (em *Hz*) de cada nota musical identificadas corretamente por um algoritmo.

- *Raw Chroma Accuracy* - Estima qual é a família de notas musicais que a *feature* de áudio *f0* pertence, considerando o intervalo de uma oitava. Esta estimativa precede o cálculo da métrica *Raw Pitch Accuracy*.

- *Overall Accuracy* - Computa a proporção de todos os *frames* de áudio que foram estimados corretamento considerando o resultado das outras métricas específicas para a tarefa de *extração de melodias*.

A avaliação dos modelos é feita em duas etapas separadas para a camada de *Encoder* e *Decoder* do modelo proposto no experimento. A razão desta avaliação em duas etapas é permitir a medição individual de cada tipo de Rede Neural Profunda, CNN e LSTM, além da avaliação geral.

Na primeira etapa, usou-se as principais métricas específicas para *extração de melodias* que se aplicam à rede CNN–Melody usada como *Encoder* e os resultados já consolidados podem ser consultados no artigo escrito por Bittner et al. (2017).

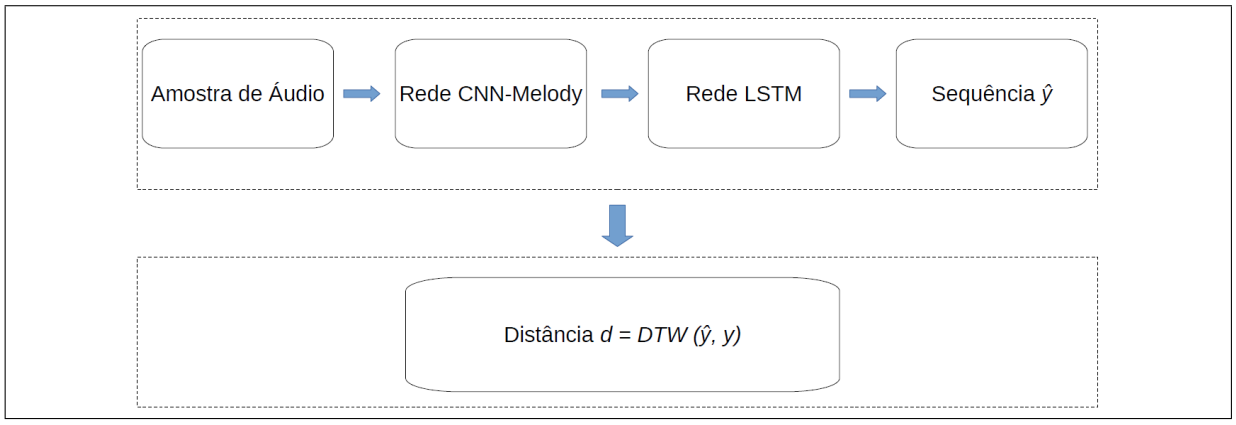
Na segunda etapa, a métrica MSE é usada para avaliar o modelo LSTM-CQT e LSTM-Wave usado como *Decoders* durante o treinamento, enquanto que as métricas de Acurácia são aplicadas ao modelo LSTM-MIDI para avaliar a classificação das notas musicais.

Após a fase de treinamento, o uso do algoritmo DTW criado por Shou, Mamoulis e Cheung (2005) permitiu uma avaliação complementar das predições realizadas pelo modelo LSTM-Wave ao permitir comparar as sequências de áudio geradas pelo modelo proposto no experimento com as sequências de sons gravadas previamente como gabaritos dos áudios esperados.

Considerando sequências de áudio como *séries temporais*, o uso do DTW foi possível, pois é um algoritmo específico para comparação de séries temporais e sequências, conforme explicado na seção 2.5, servindo como métrica de similaridade baseada em distância euclidiana.

A implementação do DTW a ser utilizada no experimento proposto foi elaborada por Meert et al. (2019). Um fluxo com as principais etapas do processo de avaliação do áudio gerado é ilustrado na Figura 35:

Figura 35 – Fluxo de análise de áudio com DTW



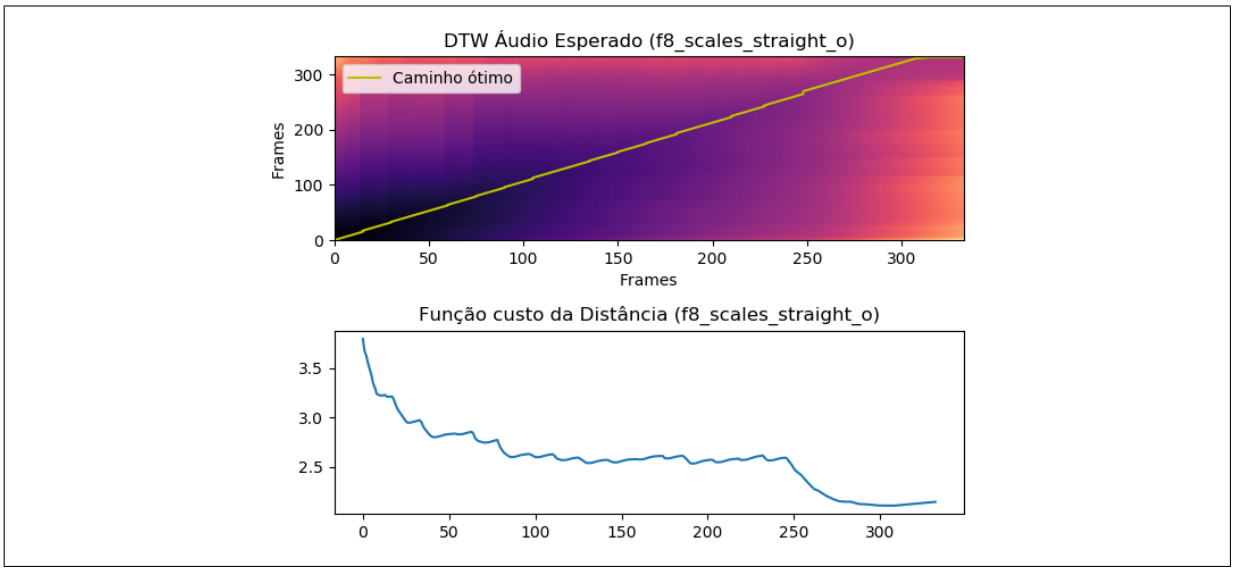
Fonte: Elaborado pelo autor.

Na Figura 35, considere $\hat{y}$ a predição da sequência e y a sequência escolhida como gabarito (previamente preparada). Quanto mais próximo de zero a distância d estiver, mais similares são as sequências.

No caso da rede LSTM-Wave, o algoritmo DTW foi usado para verificar se o áudio produzido é similar ao áudio esperado para cada exercício de solfejo. A comparação é feita no *domínio da frequência* utilizando a sequência de *frames* contidas nas representações em formato CQT das Notas Musicais com timbres de piano. Quando os sons são similares, o *Caminho Ótimo* traçado pelo DTW é representado por uma linha diagonal cuja trajetória torna-se mais retilínea na proporção que as sequências dos áudios coincidem.

A Figura 36 mostra o resultado da análise feita pelo DTW e a *Função Custo* de *Distância Euclidiana* que avalia quantitativamente a dissimilaridade entre as sequências de áudio. No experimento, um custo abaixo de 2,5 significa que os áudios são similares, enquanto valores mais altos que 7,0 revelam uma disparidade entre os sons.

Figura 36 – Exemplo de análise de áudio com DTW



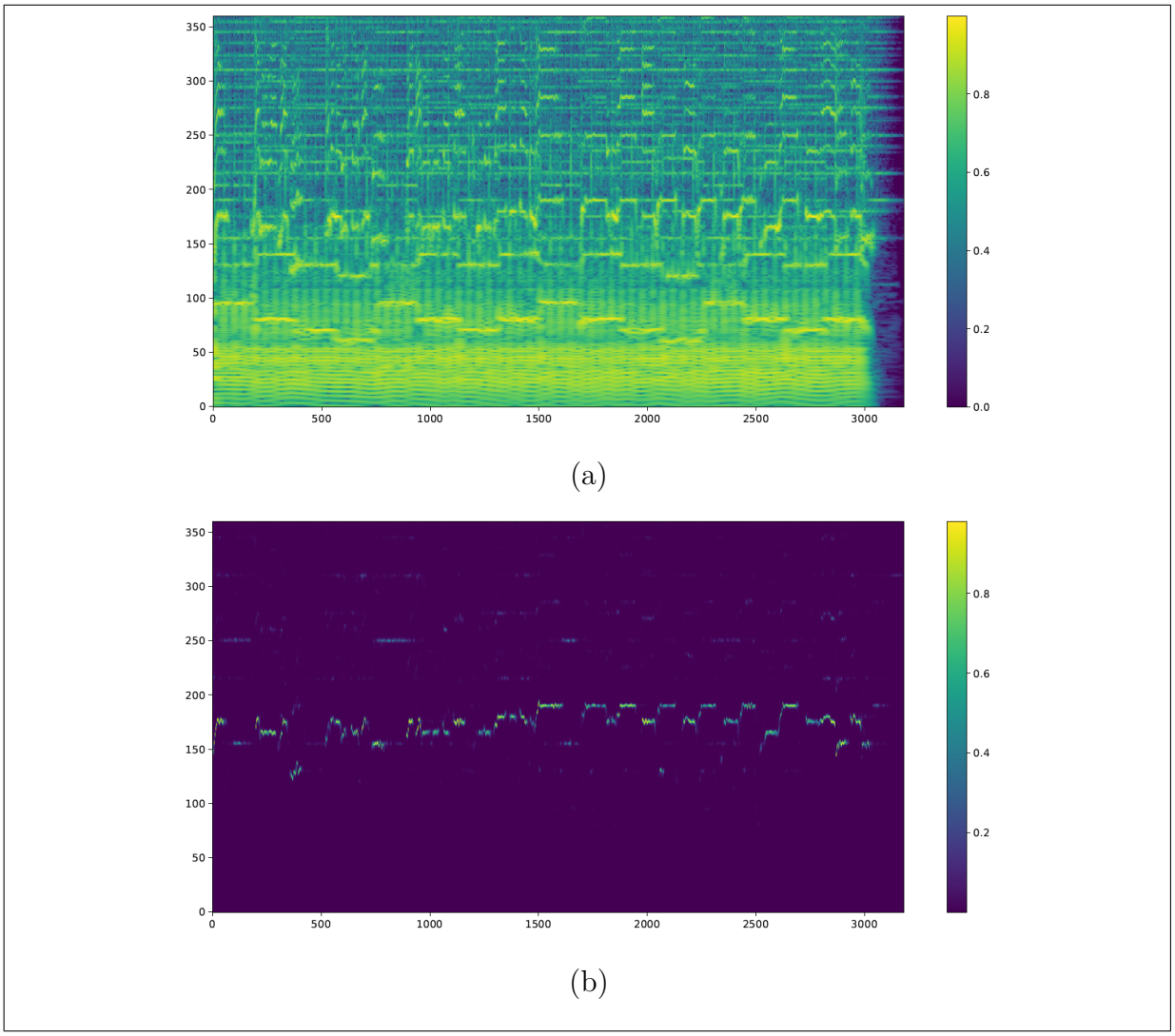
Fonte: Elaborado pelo autor.

4.4 Retreinamento do Modelo CNN–Melody para Extração de Melodias

Para garantir a validade do experimento com solfejos, o modelo CNN–Melody foi completamente retreinado com base no *dataset MedleyDB* (desconsiderando qualquer aprendizado anterior), juntamente com uma versão adaptada do código-fonte criado por Bittner et al. (2017).

Um exemplo de áudio utilizado no retreinamento da rede CNN–Melody, identificado como *MusicDelta_80sRock* (contido no *MedleyDB*), aparece na Figura 37 (a) sendo representado no *domínio da frequência* com CQT e a respectiva *saliência* com a melodia principal detectada é mostrada na Figura 37 (b). De maneira similar, a Figura 38 mostra a predição realizada pela rede CNN–Melody (após conclusão do treinamento) para o áudio do *exercício 01* contido no *dataset Solfège30* disponibilizado por Schramm (2015b).

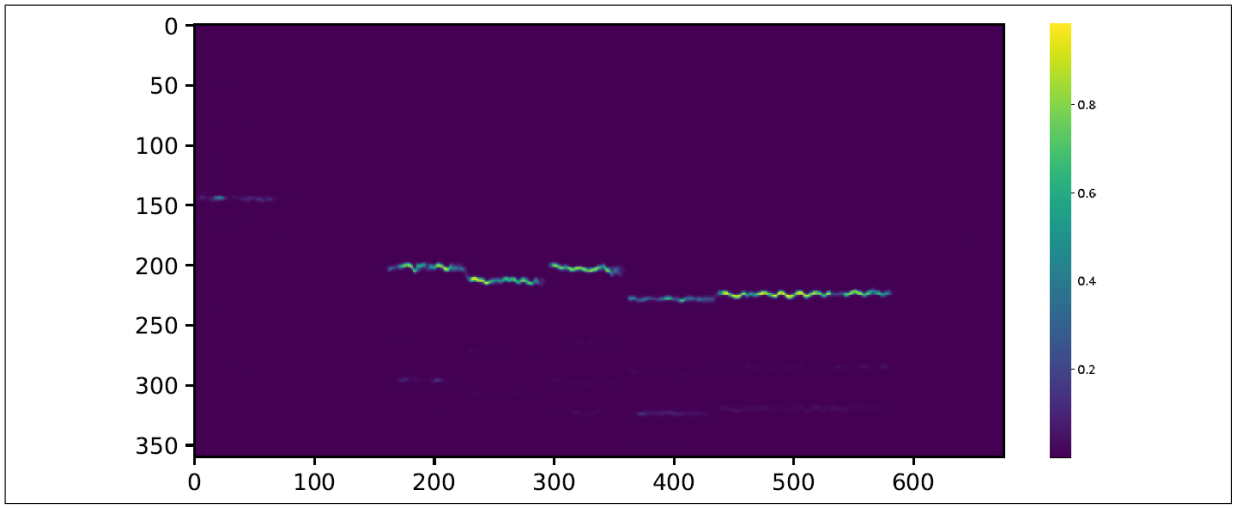
Figura 37 – Exemplo de áudio processado com a rede CNN–Melody retreinada



Fonte: Elaborado pelo autor com base no código-fonte criado por Bittner et al. (2017).

O treino da CNN–Melody foi realizado em 30 épocas e utilizou a métrica *Acurácia* (*Soft binary accuracy*) para avaliar as frequências em *Hz* de cada quadro de áudio (*frames*). A *loss function* utilizada foi a divergência de *Kullback-Leibler* (ou entropia relativa) com otimizador *Adam* fornecido pelo *framework Keras*.

Figura 38 – Exemplo de predição (*CQT*) da rede CNN–Melody retreinada



Fonte: Elaborado pelo autor com base no código-fonte criado por Bittner et al. (2017).

A arquitetura implementada para a rede CNN–Melody usando o *framework Keras* pode ser visualizado na Figura 39 conforme detalhado na seção 3.1. Ao todo, foram utilizadas seis camadas convolucionais intercaladas com camadas de normalização de dados na saída de cada operação de convolução.

O *Mean Squared Error* (MSE) foi uma métrica adicional utilizada e também aparece no gráfico de treinamento do modelo CNN–Melody ao longo das épocas conforme exibido na Figura 40.

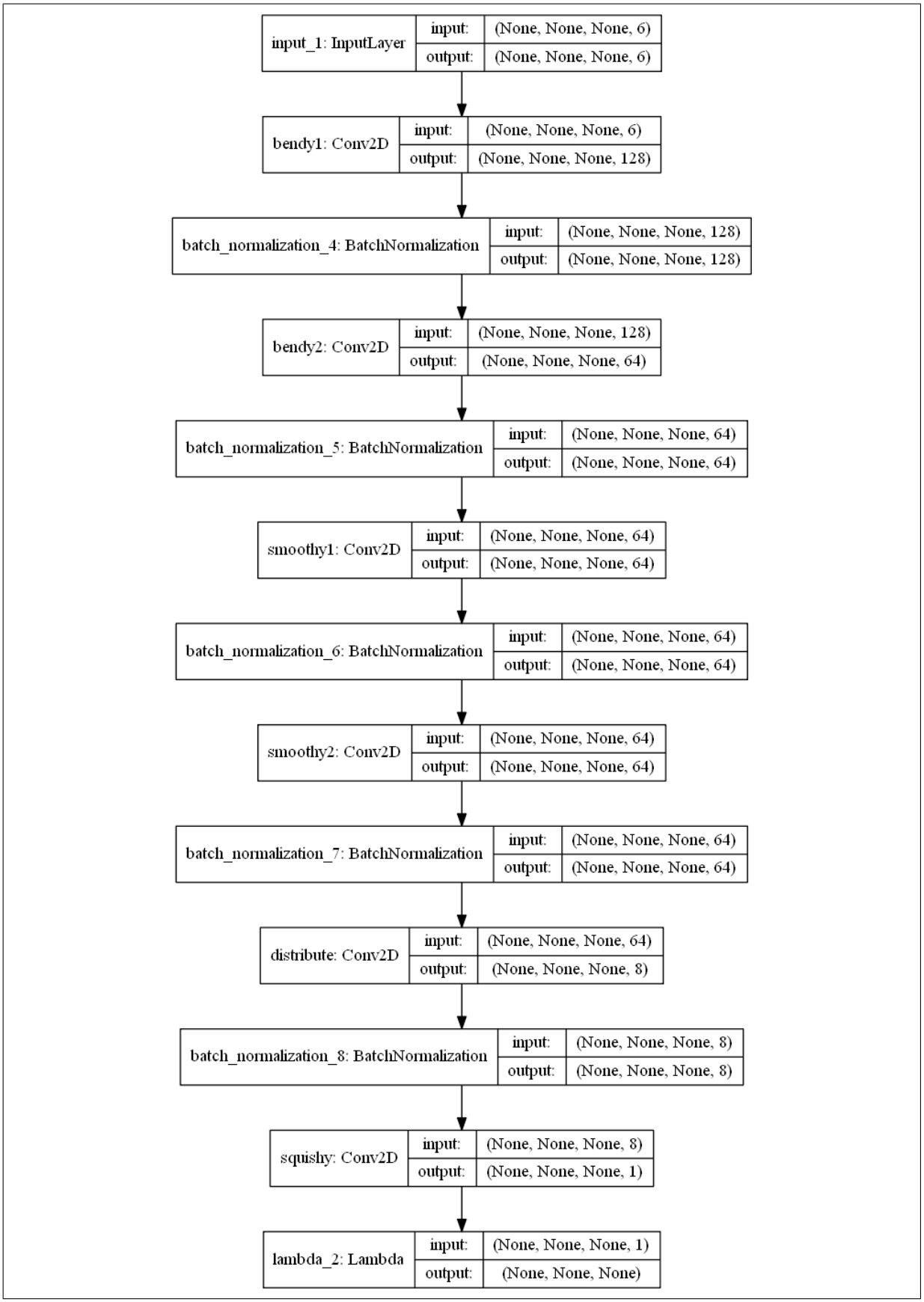
4.5 Estratégias para Implementação da Arquitetura CNN–LSTM

Em relação à arquitetura CNN–LSTM, duas estratégias foram testadas para realizar a implementação das Redes Neurais Profundas apresentadas na proposta da seção 3, ilustrada na Figura 23.

Na primeira estratégia, foram realizados testes baseados na arquitetura *CNN–LSTM* que foi descrita por Brownlee (2017), além da proposta apresentada em Hawthorne et al. (2018) para o modelo *Onsets and Frames*. Em seguida, a segunda estratégia foi avaliada através de uma Rede Convolucional combinada com LSTM conhecida como *ConvLSTM* que foi criada por Xingjian et al. (2015).

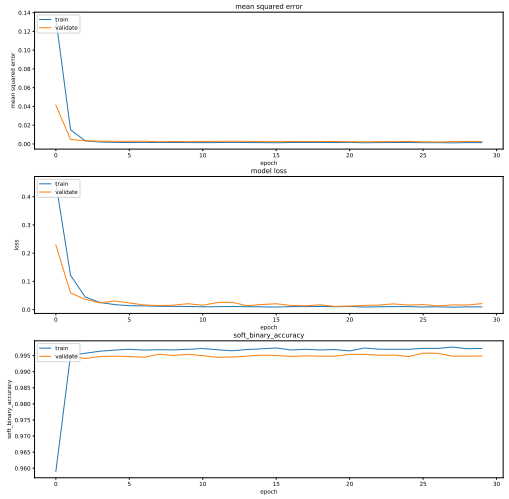
Segundo Xingjian et al. (2015), a principal diferença entre as duas arquiteturas de redes citadas, *CNN–LSTM* e *ConvLSTM*, está no fato da **CNN–LSTM** utilizar uma representação interna do espaço latente com um tensor (ou vetor) de 1D enquanto que

Figura 39 – Arquitetura da rede CNN–Melody criada por Bittner et al. (2017)



Fonte: Elaborado pelo autor com base no código-fonte criado por Bittner et al. (2017).

Figura 40 – Retreinamento da rede CNN–Melody criada por Bittner et al. (2017)



Fonte: Elaborado pelo autor com base no código-fonte criado por Bittner et al. (2017).

a rede **ConvLSTM** substitui as multiplicações internas de matrizes por operações de convolução e utiliza um tensor 3D para representar os mesmos dados no espaço latente (*encoding*). O uso interno de um tensor 3D ajuda a preservar a informação espaço-temporal contida nos dados de áudio representados no *domínio da frequência* conforme explicado na seção 4.2.

O modelo *ConvLSTM* que foi avaliado inicialmente para o experimento é uma variante da proposta original descrita por Xingjian et al. (2015) usando uma implementação específica que é oferecida pela classe *keras.layers.**ConvLSTM2D*** do *framework Keras*.

Por restrições de *hardware* relacionadas à questões de memória total da GPU e capacidade de processamento, as camadas CNN que compõe o modelo *ConvLSTM* teve que ser limitada à quantidade máxima de 16 filtros, da mesma maneira, a quantidade de camadas empilhadas não pode ultrapassar o limite de 10 *layers*.

A classe *keras.layers.**Conv2DTranspose*** do *framework Keras* foi utilizada para implementar a operação de *deconvolução* nas camadas finais do modelo *ConvLSTM* e foi adicionada com o intuito de reverter o processo de convolução e retornar imagens com tamanhos compatíveis à entrada. Esta abordagem mostrou-se inadequada porque modificou a quantidade de *timesteps* contidas nos áudios, portanto, entrando em conflito com os *shapes* dos modelos LSTM subsequentes (LSTM–CQT, LSTM–MIDI e LSTM–Wave).

Outro aspecto de incompatibilidade da arquitetura *ConvLSTM* identificado durante o experimento foi que este tipo de arquitetura é mais adequado para prever múltiplos *quadros* de imagens em vídeos, enquanto que os áudios representados no *domínio da frequência* são imagens estáticas. Portanto, a estratégia que prevaleceu para a implementação do experimento final desta dissertação foi aquela que utiliza a arquitetura *CNN–LSTM* explicada em Brownlee (2017) e proposta por Hawthorne et al. (2018) com adaptações e variações específicas para o caso dos solfejos.

4.6 Implementação das redes LSTM–CQT, LSTM–MIDI, LSTM–Wave e CNN–CQT

As redes LSTM que constituem a proposta apresentada na seção 3, Figura 23, foram projetadas com base nas características do *dataset VocalSet* que contém exercícios de solfejos gravados individualmente para cada cantor ou cantora, sendo exclusivamente monofônico, ou seja, sem acompanhamento de instrumentos musicais ou várias pessoas cantando simultaneamente (que é uma característica de áudios polifônicos).

A rede CNN–CQT foi uma implementação extra criada para explorar a detecção de *Onsets* de maneira automática e será explicada brevemente na subseção 4.6.1, embora não tenha sido incorporada no experimento principal desta dissertação.

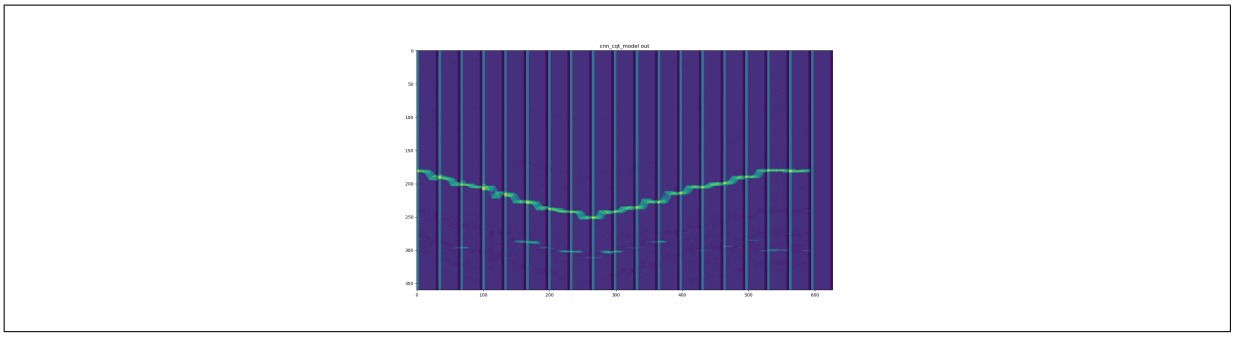
4.6.1 Definição da CNN–CQT

Onsets são eventos musicais que caracterizam o início de uma nova nota musical, geralmente, contidos em *bins* de frequência com maior intensidade de energia (HAWTHORNE et al., 2018).

A rede CNN-CQT implementada foi uma breve tentativa de realizar a detecção automática dos *onsets* presentes nos exercícios de solfejo, mas os resultados obtidos não foram integrados aos demais componentes da arquitetura CNN-LSTM proposta na seção 3, Figura 23, devido à complexidade inerente a esta tarefa específica em MIR que, dentre outras coisas, permite realizar a segmentação automática de áudio.

Os resultados preliminares aparecem na Figura 41. É possível perceber longos retângulos verticais destacando a ocorrência de *onsets* no início de cada *frame* (com coloração amarelada). Uma possível explicação para o efeito observado na Figura 41 é a forma (dimensões) dos filtros utilizados.

Figura 41 – Exemplo de *onsets* destacados pela rede CNN-CQT



Fonte: Elaborado pelo autor.

Segundo Pons et al. (2017) os filtros verticais com dimensões $1 \times n$ capturam as frequências presentes no áudio, revelando o conteúdo harmônico enquanto os filtros horizontais com dimensões $m \times 1$ capturam os aspectos temporais como os eventos de *onsets*. No entanto, vale ressaltar que no experimento descrito nesta dissertação os *shapes* e tensores foram transpostos para representar o áudio na forma $bins \times frames$ que é exigida pela rede LSTM-CQT. Assim, ao invés de se utilizar os *filtros temporais* da forma indicada por Pons et al. (2017), foi necessário realizar a mesma operação de transposição nos filtros, mas este fato não invalida o efeito previsto.

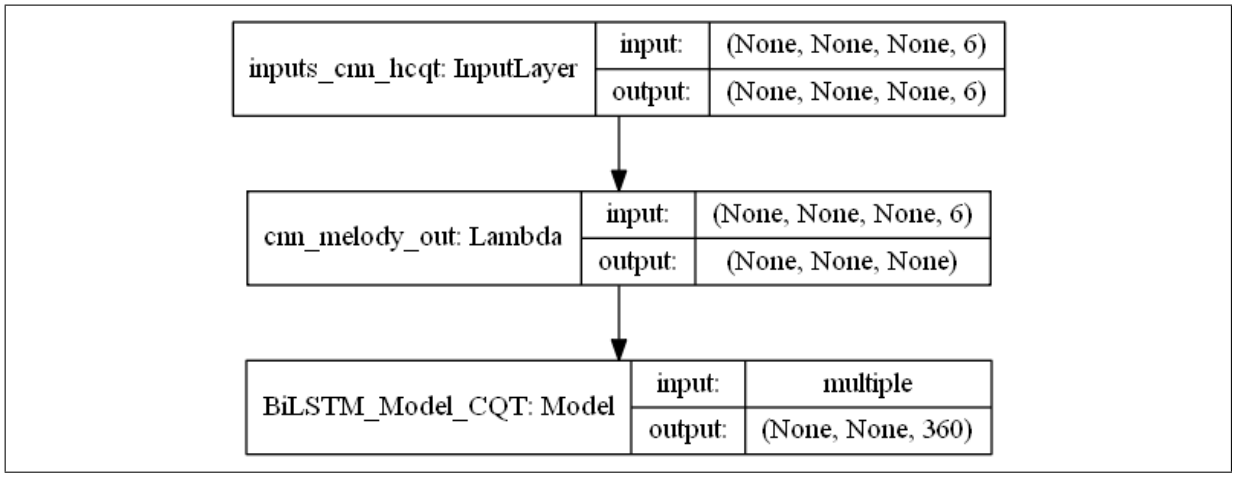
4.6.2 Definição da LSTM–CQT

A rede LSTM–CQT é o componente responsável por converter a melodia extraída do solfejo para uma representação no *domínio da frequência* que é equivalente ao timbre de instrumento musical do piano. Para isto, é realizada uma *tarefa de regressão* na qual o sinal obtido é convertido para o formato CQT. A Figura 42 mostra a arquitetura híbrida da rede LSTM–CQT. É possível observar que a rede LSTM–CQT é concatenada à saída da rede CNN–Melody.

A rede LSTM–CQT é detalhada na Figura 43. Na primeira camada (concatenada na rede CNN–Melody) foi avaliado o uso de uma rede LSTM normal e outra Bidirecional, não apresentando diferenças muito significativas quanto ao treinamento. Porém, a LSTM Bidirecional apresentou mais problemas em relação à *overfitting*, por isso, optou-se pelo uso da rede LSTM normal.

A última camada da rede LSTM–CQT utiliza uma Rede Neural do tipo *FeedForward* para a *tarefa de regressão* considerando cada *timestep* produzido pela camada LSTM

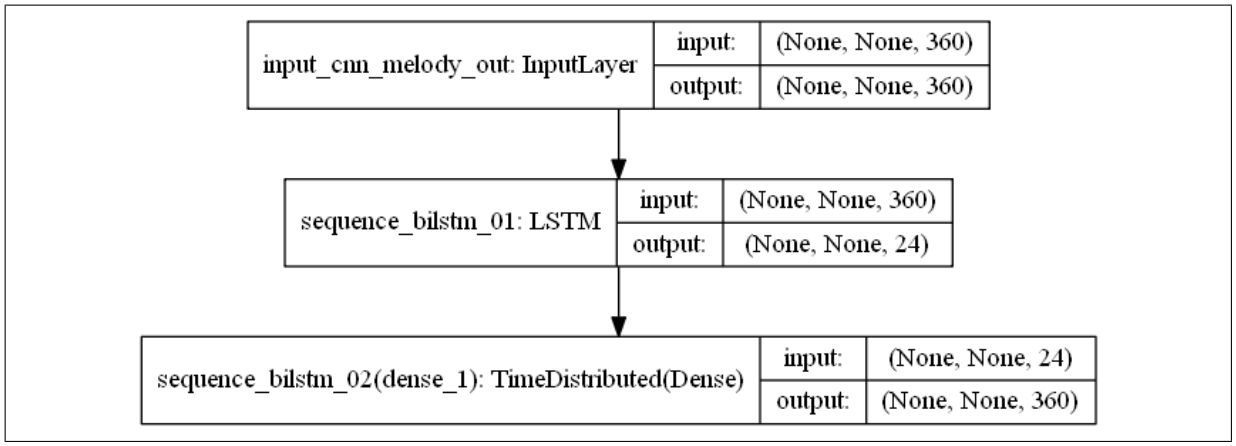
Figura 42 – Arquitetura da rede LSTM–CQT



Fonte: Elaborado pelo autor.

anterior que equivale aos *frames* existentes no áudio do solfejo. Os detalhes sobre as funções de ativação utilizadas e quantidade de neurônios podem ser vistos na subseção C.2.1 e os dados de entrada e saída podem ser encontrados no Apêndice D.

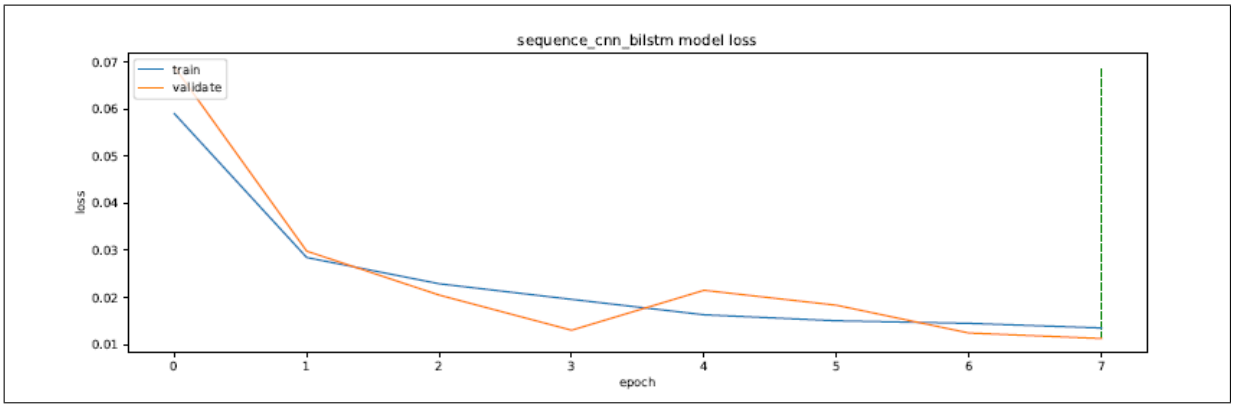
Figura 43 – Camadas da rede LSTM–CQT



Fonte: Elaborado pelo autor.

A *loss function* utilizada para treinamento da rede LSTM-CQT em 7 épocas foi o *Mean Squared Error* (MSE) com otimizador *Adam*. A Figura 44 mostra o gráfico de aprendizado da rede LSTM-CQT e a linha vertical tracejada destaca qual época obteve o melhor desempenho.

Figura 44 – Gráfico de treinamento da rede LSTM–CQT



Fonte: Elaborado pelo autor.

4.6.3 Definição da LSTM–MIDI

A rede LSTM–MIDI foi implementada considerando duas abordagens para avaliação das notas musicais presentes no solfejo. Na primeira abordagem, é realizada uma *tarefa de classificação* de cada *bloco* de *frames* contidos no solfejo, atribuindo uma nota musical a cada *bloco* ou *agrupamento de frames*.

Esta forma de classificação das notas musicais não considera aspectos como duração da nota musical, mas reconhece a amplitude do sinal presente naquele *frame* e respectivo *bin* da frequência (*Hz*) mais ativa. Nesta dissertação, esta abordagem será chamada de classificação *note-by-note* (ou em tradução livre do inglês, "nota por nota") e o objetivo é identificar a sequência de notas musicais cantadas.

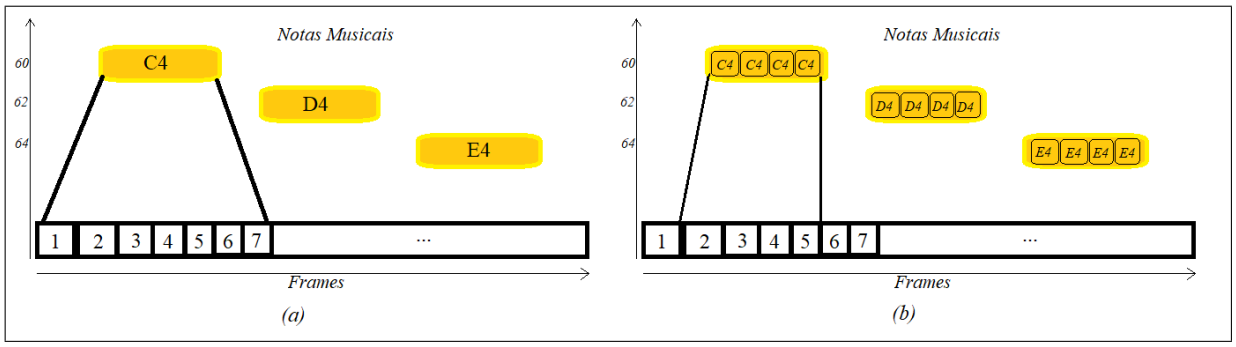
A outra abordagem consiste em classificar as notas musicais presentes em cada *frame* individualmente, funcionando como uma aproximação mais simples de técnicas mais complexas como ***pitch tracking*** que identifica a *feature f0* (frequência fundamental) presente em cada instante de um sinal de áudio para a *sinusóide* analisada (SERRA, 1989).

Como resultado, é obtido um *piano-roll*, ou seja, uma sequência de *pitches* (ou frequências de notas musicais) dispostas horizontalmente a cada instante de tempo de acordo com cada classe de nota musical, que está no eixo vertical do gráfico gerado pelo *piano-roll* e é expresso em escala logarítmica através de códigos MIDI. Nesta dissertação, esta abordagem de classificação será chamada de *frame-by-frame* (ou "quadro a quadro").

Para ilustrar as classificações *note-by-note* e *frame-by-frame*, a Figura 45 (a) mostra o relacionamento entre as notas musicais atribuídas a um *bloco de frames* enquanto a

Figura 45 (b) destaca a classificação individual de cada *frame* para a nota musical correspondente.

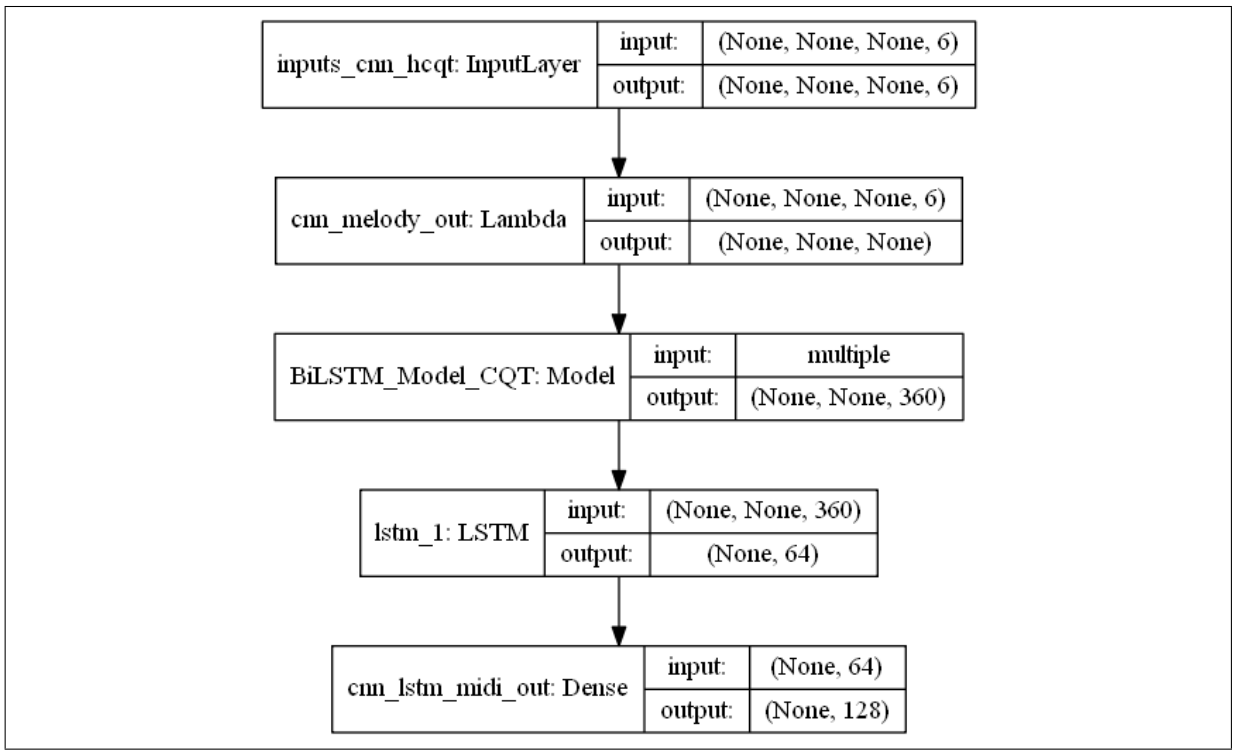
Figura 45 – Relação entre Notas Musicais e *Frames*



Fonte: Elaborado pelo autor.

A arquitetura da rede LSTM–MIDI usada na classificação *note-by-note* é mostrada na Figura 46. Na saída da última camada é usada uma Rede Neural do tipo *FeedForward* com ativação *softmax* para realizar a *tarefa de classificação* das Notas Musicais, atribuindo o resultado da análise dos blocos de *frames* de áudio para o código MIDI equivalente, contido no intervalo [0..127] (vide Apêndice D).

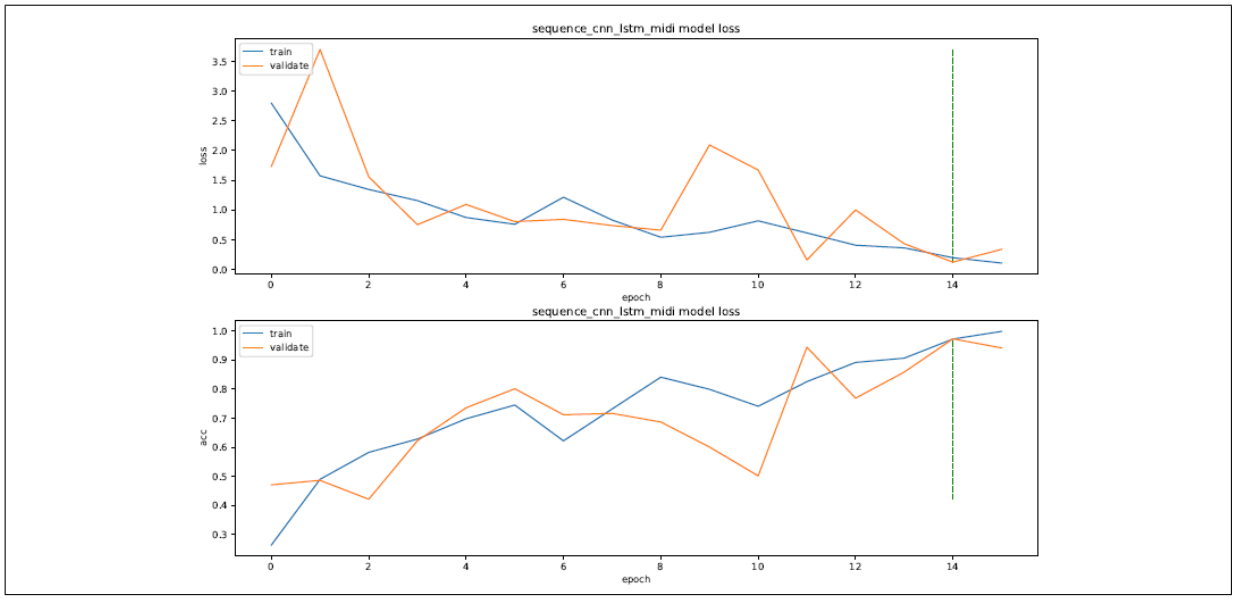
Figura 46 – Arquitetura da rede LSTM–MIDI *Notes*



Fonte: Elaborado pelo autor.

O treinamento da LSTM–MIDI *Notes* foi feito com a *loss function* chamada *Categorical Cross Entropy* para multiclassificação e otimizador *Adam*. O gráfico do treinamento é mostrado na Figura 47.

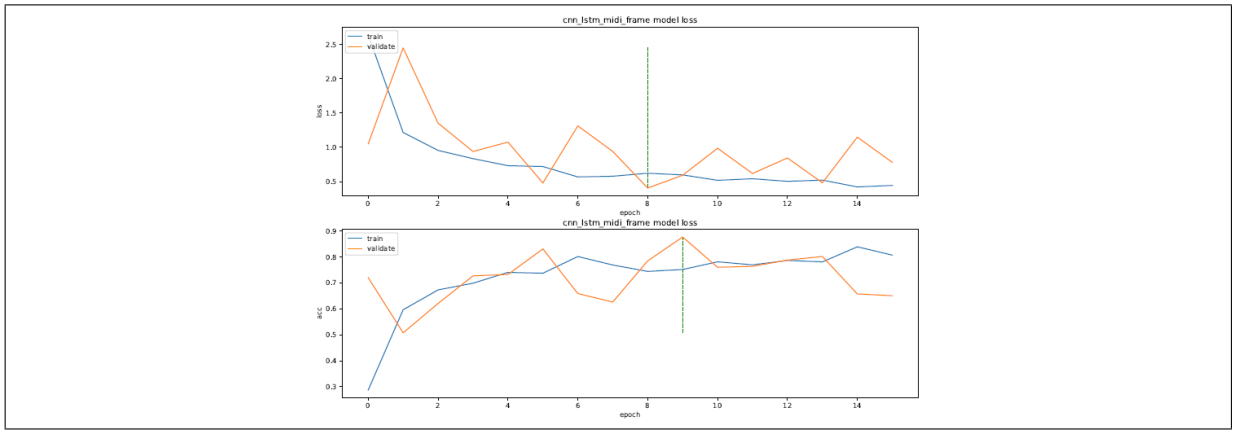
Figura 47 – Gráfico de treinamento da rede LSTM–MIDI *Notes*



Fonte: Elaborado pelo autor.

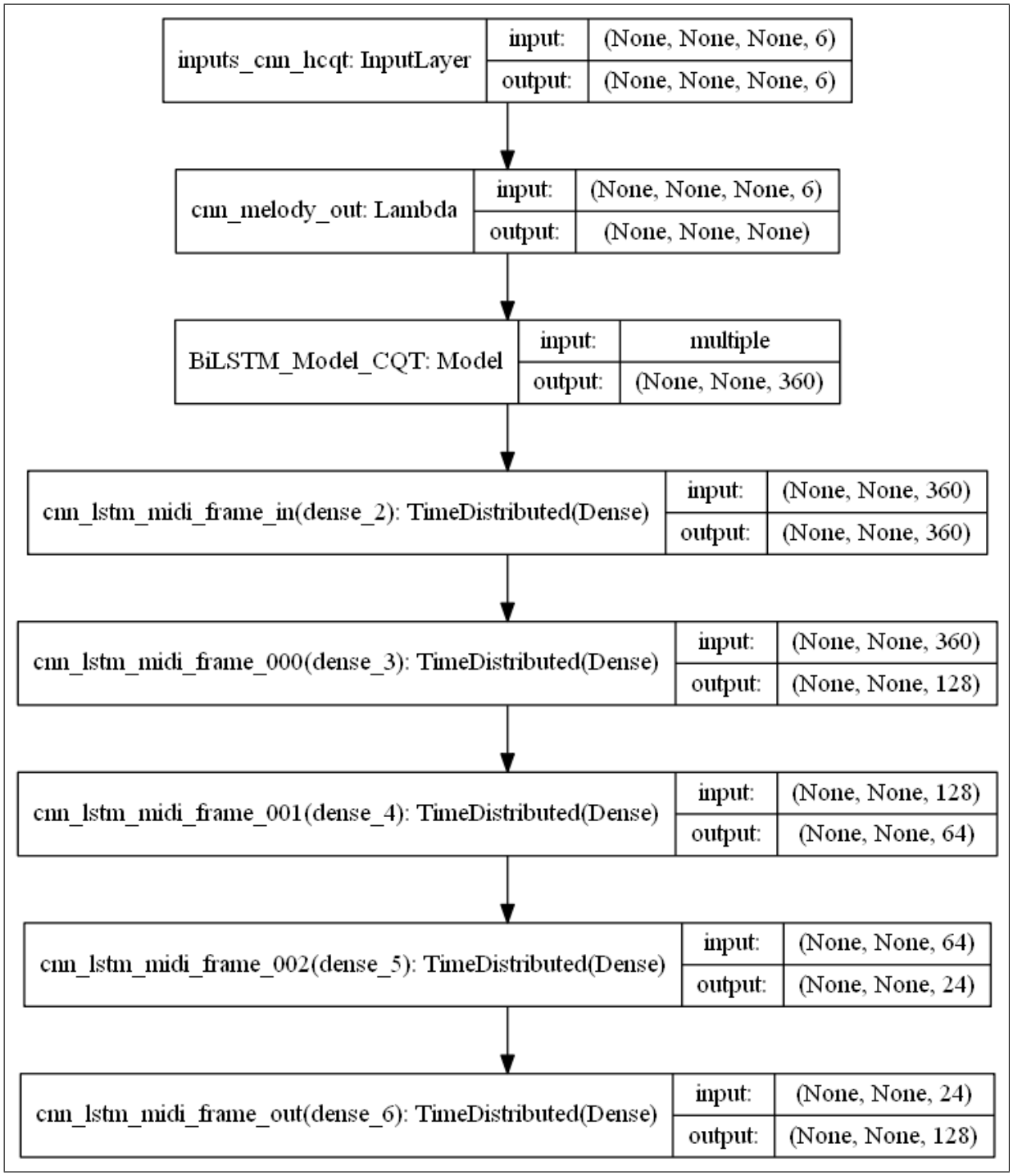
Para a classificação *frame-by-frame*, a Arquitetura da rede LSTM-MIDI é modificada para que as camadas finais considerem os *timesteps* produzidos pela camada LSTM com os *frames* de áudio que serão classificados individualmente. O gráfico de treinamento da rede LSTM-MIDI *Frames* é mostrado na Figura 48 e todas as camadas de Redes Neurais do tipo *Feedforward* que foram utilizadas são apresentadas na Figura 49.

Figura 48 – Gráfico de treinamento da rede LSTM–MIDI *Frames*



Fonte: Elaborado pelo autor.

Figura 49 – Arquitetura da rede LSTM–MIDI *Frames*



Fonte: Elaborado pelo autor.

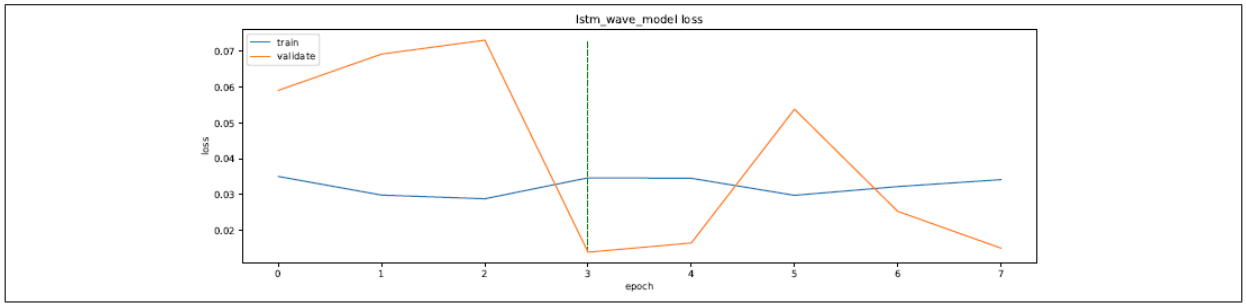
Mais detalhes sobre as funções de ativação utilizadas e quantidades de neurônios podem ser encontradas na seção subseção C.3.1 e os respectivos dados utilizados para o treinamento e resultados obtidos são encontrados no Apêndice D.

4.6.4 Definição da LSTM–Wave

A rede LSTM–Wave realiza uma *tarefa de regressão* para sintetizar o áudio final com timbre de instrumento musical similar a um piano. Isto é feito convertendo a saída da rede LSTM–CQT para o equivalente em formato WAVE após a classificação das Notas Musicais pela rede LSTM-MIDI *Notes*.

O treinamento da rede LSTM-Wave utilizou o*Mean Squared Root* como *loss function* e otimizador *Adam*. O gráfico de treinamento é mostrado na Figura 50. A linha vertical tracejada destaca a época que obteve melhor resultado durante o treinamento.

Figura 50 – Gráfico de treinamento da Rede LSTM-Wave

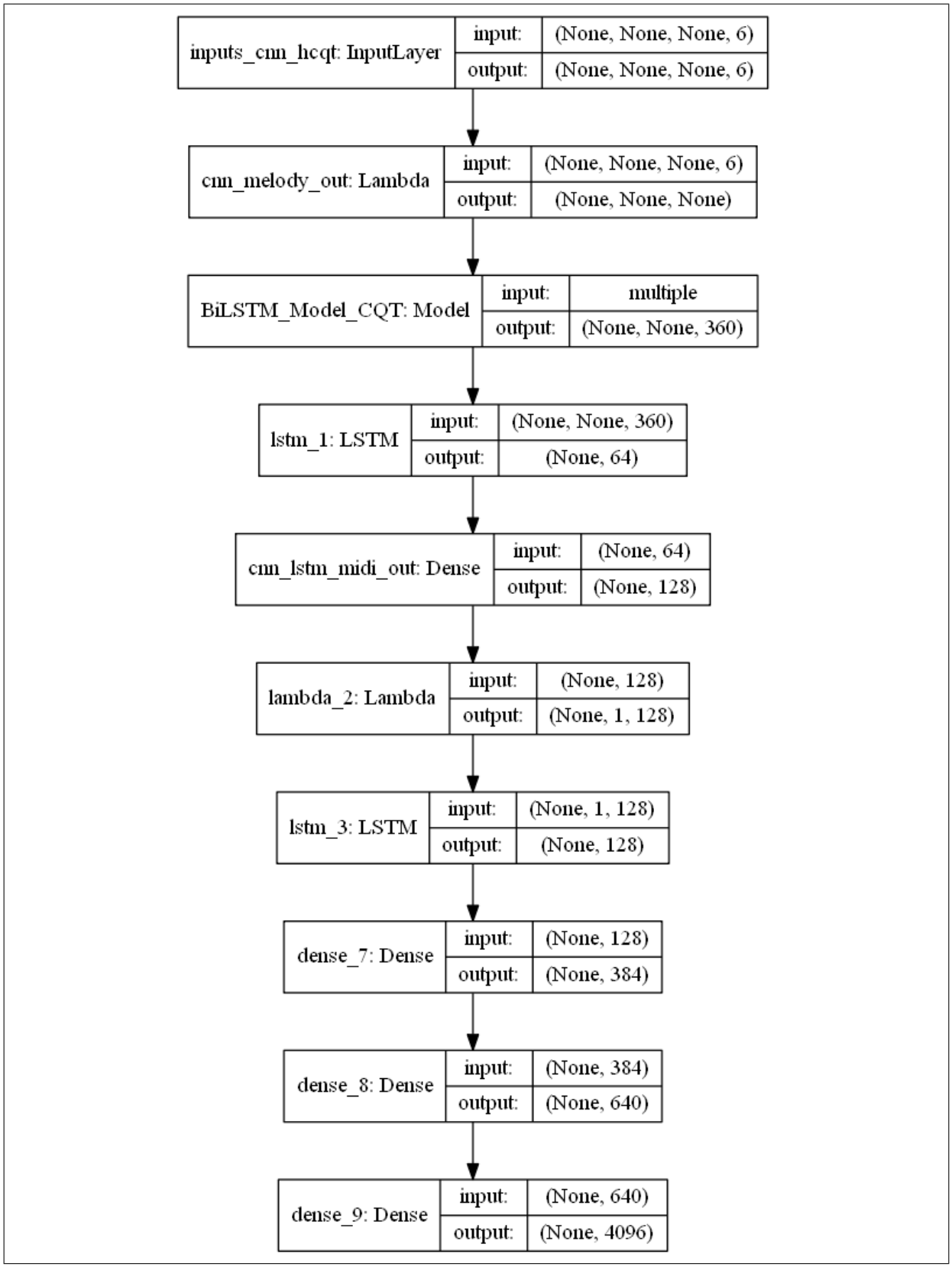


Fonte: Elaborado pelo autor.

No experimento, foram realizadas diversas configurações na arquitetura desta rede que foi conectada diretamente à rede LSTM-CQT e, depois, conectada à rede LSTM-MIDI. Dentre as diferenças percebidas, destaca-se uma melhor qualidade na sintetização de áudio e reconhecimento de *pausas* e *silêncios* quando a rede é conectada na saída da LSTM-MIDI. A Arquitetura de Camadas utilizada na rede LSTM-Wave é mostrada na Figura 51.

É possível observar a rede LSTM-Wave agrega os modelos CNN-Melody, LSTM-CQT e LSTM-MIDI *Notes* em sua composição. Os detalhes de implementação estão na subseção C.4.1 e respectivos dados de treinamento e resultados obtidos podem ser verificados na Apêndice D.

Figura 51 – Arquitetura da Rede LSTM-Wave



Fonte: Elaborado pelo autor.

4.7 Análise dos Resultados

De acordo com os objetivos definidos na seção 1.3, o uso de redes CNN e LSTM combinadas para realizar a transcrição melódica das notas musicais presentes em exercícios de solfejo mostrou-se viável.

No experimento realizado, as Redes Neurais do tipo LSTM que foram utilizadas realizaram *tarefas de regressão* e *tarefas de classificação* para obter a transcrição automática de notas musicais à partir dos solfejos contidos no *dataset VocalSet*.

Embora a rede CNN–Melody utilizada para *extração de melodias* tenha sido incorporada como parte do experimento, é válido ressaltar que sua elaboração e contribuições originais são creditadas a Bittner et al. (2017). No entanto, foi necessário realizar diversas adaptações no código-fonte original disponibilizado para que se tornasse possível realizar o retreinamento da rede CNN–Melody desde o estágio inicial (desconsiderando treinamentos realizados anteriormente) utilizando o *dataset MedleyDB* na sua versão atual que é diferente da versão utilizada no trabalho de Bittner et al. (2014). Os principais motivos para se realizar o retreinamento foi permitir a verificação da reprodutibilidade do estudo desenvolvido por Bittner et al. (2017) e reduzir a quantidade de dados necessárias para treinar esta rede devido à restrições de *hardware* que representavam um fator limitante para o experimento desta dissertação.

Sobre o retreinamento da rede CNN–Melody, o resultado das métricas mais relevantes que foram extraídas de acordo com as definições da seção 4.3 usando a biblioteca de *software* **mir_eval** (RAFFEL et al., 2014) são apresentadas na Tabela 1.

Tabela 1 – Métricas da rede CNN–Melody

Precision	**Recall**	**Accuracy**	**Raw Chroma Accuracy**
$82,28\%$	$82,33\%$	$77,71\%$	$87,23\%$

Considerando as redes LSTM–CQT, LSTM–MIDI *Notes*, LSTM–MIDI *Frames* e LSTM–Wave desenvolvidas durante o experimento, apenas a rede LSTM–MIDI foi utilizada para efeitos de comparação e desempenho geral do protótipo executado porque não há contraparte para as outras redes LSTM citadas. Neste caso, o algoritmo **pYIN** incorporado na ferramenta de análise de áudio *Tony* (discutida na subseção 2.8.3) foi utilizado como referência na comparação de transcrição automática de notas musicais identificadas em melodias.

A arquitetura das redes LSTM e *FeedForward* apresentadas no estudo de Rigaud e Radenen (2016) na subseção 2.8.5, além de alguns aspectos técnicos da arquitetura proposta por Hawthorne et al. (2018), também foram utilizados como base de comparação para alguns desafios encontrados durante o experimento.

As métricas da rede LSTM-MIDI que constam na Tabela 2 são *médias ponderadas* extraídas com a biblioteca de *software* **scikit-learn** (PEDREGOSA et al., 2011). Esta forma de cálculo foi utilizada porque o *dataset VocalSet* possui um *desbalanceamento de classes* em relação às notas musicais presentes nos exercícios de solfejo (por exemplo, acidentes musicais como *sustenidos* e *bemóis* ocorrem com menor frequência).

Tabela 2 – Métricas da rede LSTM-MIDI

Item	**Prec.**	**Recall**	**Acc.**	**F-Meas.**	**Técnica**
LSTM-MIDI-*Notes*: Vogal *a*	94,00%	88,00%	88,00%	88,00%	*Arpeggio*
LSTM-MIDI-*Notes*: Vogal *e*	44,00%	50,00%	50,00%	46,00%	*Arpeggio*
LSTM-MIDI-*Notes*: Vogal *i*	81,00%	50,00%	50,00%	58,00%	*Arpeggio*
LSTM-MIDI-*Notes*: Vogal *o*	54,00%	62,00%	62,00%	56,00%	*Arpeggio*
LSTM-MIDI-*Notes*: Vogal *u*	62,00%	62,00%	62,00%	62,00%	*Arpeggio*
LSTM-MIDI-*Notes*: Vogal *a*	81,00%	78,00%	78,00%	78,00%	*Escala*
LSTM-MIDI-*Notes*: Vogal *e*	67,00%	61,00%	61,00%	60,00%	*Escala*
LSTM-MIDI-*Notes*: Vogal *i*	72,00%	72,00%	72,00%	70,00%	*Escala*
LSTM-MIDI-*Notes*: Vogal *o*	91,00%	89,00%	89,00%	89,00%	*Escala*
LSTM-MIDI-*Notes*: Vogal *u*	87,00%	89,00%	86,00%	86,00%	*Escala*
LSTM-MIDI-*Notes*	–	–	71,90%	–	**Todas**
LSTM-MIDI-*Frames*	–	–	–	57,24%	**Todas**
Item	**Prec.**	**Recall**	**Acc.**	**F-Meas.**	***Dataset***
Tony: **pYIN***(s=0.0,prn=0.0)*	–	–	83,00%	61,00%	**ISMIR2014**
Rigaud e Radenen (2016)	–	–	85,06%	93,15%	**iKala**
Rigaud e Radenen (2016)	–	–	75,03%	79,19%	**MedleyDB**

No caso específico da rede LSTM-MIDI-*Frames*, foi utilizado uma comparação parcial com a métrica *F COnP* que é baseada em *F-Measure* e mede quando a frequência de uma nota musical (*pitch*) e o *onset* foram detectados corretamente (MAUCH et al., 2015).

4.7.1 Análise dos desafios encontrados durante o experimento

A escolha da representação de áudio para utilização nas redes LSTM-CQT e LSTM-Wave foi uma questão técnica que demandou vários testes de arquiteturas alternativas até atingir uma configuração funcional.

Considerando o *domínio temporal*, os dados brutos em formato WAVE costumam gerar

um alto volume de *samples* a cada segundo de áudio, dependendo da taxa de amostragem escolhida (por exemplo, $44.000Hz$). Isto, pode ser um fator limitante quando a sequência de áudio a ser aprendida por uma rede LSTM ultrapassa $4s$ de duração, por exemplo, sendo necessário reduzir a qualidade do áudio comprimindo sua taxa de amostragem (*downsampling*) para $16.000Hz$, $11.000Hz$ ou $8.000Hz$ e reduzindo a quantidade de canais de estéro para mono.

Quando se explora a representação do áudio no *domínio da frequência* com STFT ou CQT surgem outros desafios relacionados à necessidade de se preservar as características do áudio como *magnitude*, *amplitude* e *fase do som* (vide seção 4.2).

No experimento, o uso do STFT foi preterido em favor da representação construída com CQT que dificulta a reconstrução da fase do áudio em comparação ao STFT que é invertível na maioria das vezes. Mas, para a aplicação específica do experimento, as redes BiLSTM e LSTM tiveram dificuldade para reconstruir o sinal de áudio. Por isso, o CQT foi utilizado para representar a parte Real do som, enquanto a parte Imaginária foi descartada. Devido a este fato, o CQT foi usado para representar o timbre de piano na rede LSTM-CQT e o formato WAVE foi utilizado na rede LSTM-Wave, após *downsampling* para $8.000Hz$, para representar o áudio final que seria sintetizado.

Por causa destas dificuldades da representação do áudio, o experimento realizado contemplou apenas a geração de notas musicais com duração fixa. Uma possibilidade de gerar áudio com tamanho variável, seria substituir as camadas *Dense* (vide Figura 51) da rede LSTM-Wave por camadas LSTM configuradas para retornar sequências de *frames* de áudio.

Em comparação ao estudo desenvolvido por Rigaud e Radenen (2016), não há problemas em se descartar a parte Imaginária do STFT porque não é realizada a sintetização do áudio na saída da rede *FeedForward* utilizada para a tarefa de *F0 Estimation*, cuja tarefa é de *classificação* e não de *regressão* como a realizada pelas redes LSTM-CQT e LSTM-Wave.

4.7.2 Análise dos principais resultados obtidos

Sobre a segmentação automática de áudio contendo melodias, os exercícios de solfejo analisados no *dataset VocalSet* possuem várias técnicas vocais e níveis de complexidade diferentes, além de timbres e tipos específicos de vozes como ilustrados na Figura 30.

Embora o *VocalSet* tenha gravações de solfejos feitas por cantores profissionais e o *dataset Solfège30* (descrito na seção 3.4) tenha sido composto por estudantes de música, em sua maioria, existem fatores em comum que tornam a tarefa de *detecção de onsets* mais difíceis de serem realizadas por DNN (e a detecção de *onsets* é fundamental para a segmentação automática de áudio).

No experimento, o fator comum, identificado nestes *datasets*, que dificulta o aprendizado das Redes Neurais e a complexidade na detecção automática de *onsets* é a variação da duração ou prolongamento de uma nota musical cantada. Outra característica identificada é que a voz humana cantada, normalmente, não possui os *elementos percussivos* existentes, por exemplo, em instrumentos musicais de corda como o piano.

O ataque inicial (em média, nos primeiros $200ms$ de áudio) de uma nota musical de piano, como a ilustrada na Figura 2 (b), é a parte do áudio que, normalmente, apresenta *elementos percussivos* que são originados por causa da força (intensidade) com a qual o pianista toca uma tecla do piano. Este fato constitue a principal diferença entre o experimento realizado com solfejos nesta dissertação e o trabalho desenvolvido por Hawthorne et al. (2018) para transcrição de áudio de piano. Desta forma, a idéia da rede BiLSTM usada para detecção de *onsets* do trabalho de Hawthorne et al. (2018) não pôde ser implementada no experimento com solfejos.

No entanto, a arquitetura proposta por Hawthorne et al. (2018) para a detecção de *frames* de áudio contendo melodias, serviu como influência no momento da elaboração das arquiteturas das redes LSTM-MIDI usadas no experimento que são responsáveis pela classificação das notas musicais à partir dos exercícios de solfejo. Então, ao mesmo tempo que as redes LSTM-MIDI foram influenciadas pelo trabalho de Hawthorne et al. (2018) para a detecção de notas musicais, esta abordagem mostrou-se uma alternativa para a extração indireta da *feature f0* por meio das técnicas de identificação de notas musicais definidas como *note-by-note* e *frame-by-frame* que foram explicadas na subseção 4.6.3.

5 CONCLUSÃO

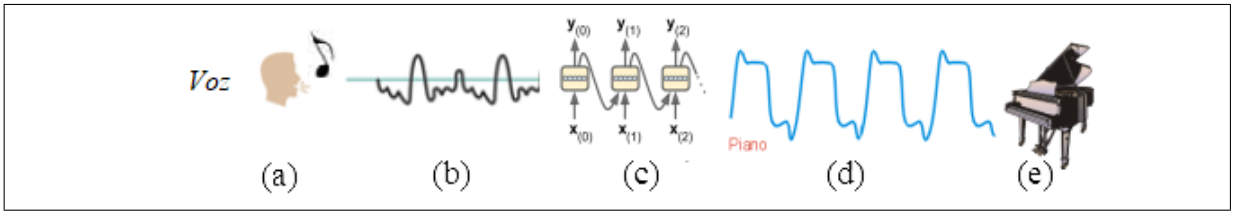
A área de *Music Information Retrieval* (MIR) é ampla e possui diversas oportunidades de pesquisa em assuntos relacionados a extração de melodias em músicas, separação de vozes, detecção de *onsets* em áudio polifônicos, análise da voz humana cantada em áudios monofônicos, modelagem acústica e tantos outros problemas que ainda estão em aberto e são tópicos ativos de pesquisa dentro da comunidade internacional de MIR.

Nessa dissertação foi explorado o uso de algoritmos de *Deep Learning* para modelagem de notas musicais à partir da transcrição de melodias cantadas em exercícios de solfejo trazendo algumas perspectivas que podem contribuir para futuros estudos nesta área. As subseções seguintes destacam os desafios encontrados, as contribuições dadas ao tema e finaliza com sugestões de continuidade da pesquisa.

5.1 Perspectivas do uso de *Deep Learning* para o Processamento de Sinais de Áudio

O experimento realizado nesta dissertação pode ser exemplificado pela Figura 52. A idéia simplificada do experimento foi permitir que uma pessoa, Figura 52 (a), cante uma melodia de solfejo, por exemplo, utilizando a entonação da vogal *a* e o som produzido pela voz humana cantada, Figura 52 (b), seja processado pelas Redes Neurais Profundas, Figura 52 (c), que irão converter o sinal de áudio obtido para um timbre de instrumento musical real, Figura 52 (d), como o piano mostrado na Figura 52 (e).

Figura 52 – Ilustração simplificada do Experimento



Fonte: Elaborado pelo autor com figuras adaptadas de Fischer e Jubilut (2019), Géron (2017) e Hollis (2017).

Em relação às principais questões de pesquisa levantadas na seção 1.2, o uso de Redes Neurais Convolucionais e Redes Neurais Recorrentes demonstrou aplicabilidade no contexto de processamento de sinais de áudio, notadamente para as tarefas executadas na área de *Music Information Retrieval* (MIR).

A Arquitetura de rede CNN–LSTM desenvolvida no experimento superou as expectativas iniciais em relação à capacidade de generalização do modelo devido às diferentes modalidades de canto e técnicas diversas que são executadas no solfejo, forçando o modelo a lidar com diferentes velocidades (BPM), timbres da voz humana cantada e quantidade limitada de amostras de notas musicais para o aprendizado. Ou seja, o modelo foi capaz de fazer predições corretas de notas musicais à partir de áudios contendo solfejos cujas as características não estavam presentes originalmente no conjunto de treino selecionado. Por exemplo, a técnica de canto vocal que foi selecionada para treinamento do modelo denomina-se *straight* e as entonações dos exercícios continham apenas a vogal *a*. Apesar deste cenário inicial restrito, o modelo foi capaz de prever corretamente as notas musicais entoadas com outras entonações de vogais, tais como *e*, *i*, *o* e *u*, utilizando-se a técnica vocal de *arpeggios* que não foi incluída no treinamento das redes LSTM (ver Anexo C).

Dessa forma, algumas situações de baixa acurácia já eram previstas dado à escassez de amostras ou *desbalanceamento de classes* em relação à quantidade de notas musicais suficientes para cada oitava da escala musical ocidental (ver Anexo D). Portanto, os resultados encontrados no experimento sugerem que à medida que novas categorias de notas musicais, técnicas vocais, ritmos e timbres de vozes são apresentados ao modelo, este aumentaria a sua capacidade de generalização.

Uma das principais dificuldades encontradas no decorrer desta dissertação está relacionada à falta de padronização no uso de *datasets* específicos para solfejo, dificultando a realização de testes e comparações (*benchmarks*). Embora o *dataset VocalSet* elaborado por Wilkins et al. (2018) tenha uma alta qualidade de organização e tenha sido criado com gravações de solfejos realizados por cantores profissionais, a aderência a este *dataset* ainda é baixa principalmente por ser muito recente (foi apresentado na *19th International Society for Music Information Retrieval Conference*, Paris, França, em 2018). Neste contexto, excetuando-se os autores originais, pode-se destacar que uma contribuição adicional desta dissertação é ter servido como primeiro trabalho acadêmico a utilizar formalmente o *dataset VocalSet* (até o presente momento[6]).

[6] De acordo com pesquisa realizada no sítio <https://search.datacite.org/works/10.5281/ZENODO.1492453> em 19/04/2020.

5.2 Contribuições

Antes de discorrer sobre as contribuições é necessário revisitar os problemas na extração de *features* de áudio em músicas que motivaram esta dissertação, por isso, é oportuno destacar algumas conclusões referentes aos itens P1 a P7 listados no Quadro 1 da seção 1.2.

No decorrer do experimento, para o problema P1 (Seleção de *Features* de Áudio), confirmou-se o fato de que o uso de transformações espectrais como STFT e CQT são adequados para a representação das estruturas harmônicas e temporais presentes no áudio de forma que os algoritmos de *Machine Learning* e *Deep Learning* possam ser beneficiados.

Dentre as diversas *features* de áudio que são apresentadas na Figura 77, a principal *feature* que influenciou os modelos testados foi a *frequência fundamental (f0)* identificada como *Pitch* (*dominant frequency*) na seção 5.4.4 do *survey* realizado por Mitrović, Zeppelzauer e Breiteneder (2010). O CQT foi mais compatível com a tarefa de transcrição melódica de solfejos.

Os problemas P2 (Extração de *Features* de Áudio) e P3 (Dificuldades com *Feature Engineering*) foram abordados com o uso de algoritmos de *Deep Learning* capazes de auxiliar na resolução de ambos os problemas de forma automática.

No entanto, o problema P4 (Hiperparâmetros) foi difícil de lidar durante o experimento, requerendo várias tentativas de calibração manual para se obter um modelo funcional, isto envolveu o teste de diferentes combinações de funções de ativação, algoritmos de otimização, quantidade de épocas e tamanho de *batches* para o treinamento dos algoritmos, dentre outras questões ligadas à restrição de tamanho de memória da GPU.

Quando se trata do problema P5 (Variedades de Timbres) o que se verificou é que o ideal seriam utilizar modelos distintos para representar os diferentes timbres de instrumentos musicais, por exemplo, não foi possível (durante os testes realizados no experimento) sintetizar o timbre de um Piano e o timbre de uma Guitarra utilizando a mesma DNN. Também observou-se que para os modelos alcançarem uma generalização satisfatória é necessário possuir amostras de tipos de vozes humanas (ver Figura 30) que não estejam sub-representadas no conjunto de treino e validação.

Duas contribuições desta dissertação referente à implementação da rede LSTM-MIDI, ajudam a responder o problema P6 (Classificação de Notas Musicais) que foi o uso da rede LSTM-MIDI *Notes* para realizar a *tarefa de classificação* das notas musicais em relação a um *bloco* de *frames* de áudio enquanto a rede LSTM-MIDI *Frames* foi utilizada para

realizar a classificação individual de cada *frame de áudio*, semelhante ao resultado obtido na tarefa de *Pitch Tracking* (SALAMON, 2013b, p.26).

Quanto ao problema P7 (Extração de Melodia e *Pitch Tracking*), esta dissertação utilizou a abordagem proposta por Bittner et al. (2017) para realizar a extração de melodias dos solfejos com uma rede CNN (discutida na seção 4.4) ao invés de implementar o próprio modelo ou seguir a proposta de Rigaud e Radenen (2016), visto que a escassez de *datasets* para solfejo é uma importante limitação para o treinamento de novos modelos específicos para a tarefa de extração de melodia e *Pitch Tracking*.

Outro fator priorizado no experimento foi a interpretabilidade do *espaço latente* das redes neurais utilizadas, tanto a CNN quanto a LSTM. No caso da CNN, não foram utilizadas camadas de *pooling* (evitando redução de dimensionalidade). Para as redes LSTM, optou-se em utilizar a mesma quantidade de *bins* e *timesteps* da rede CNN, permitindo-se manter o significado contextual e visual das notas musicais analisadas.

Em comparação ao algoritmo pYIN (MAUCH; DIXON, 2014) e MELODIA (SALAMON, 2013b), a limitação do modelo avaliado no experimento está na falta de capacidade de realizar segmentação automática do áudio com base na detecção automática de *onsets*. Sendo esta limitação uma importante motivação para estudos futuros que façam o uso de algoritmos de *Deep Learning* em linhas de pesquisas e experimentos similares como o encontrado no trabalho de Rigaud e Radenen (2016).

De toda forma, os resultados de $71,90\%$ de acurácia na detecção de notas musicais com a abordagem *note-by-note* e de $57,24\%$ com a abordagem *frame-by-frame* mostrados na Tabela 2, ambas propostas nesta dissertação, indicam que há oportunidades de estudos mais aprofundados utilizando métodos alternativos para transcrição melódica de solfejos.

5.3 Trabalhos Futuros

É importante, em estudos futuros, avaliar os impactos das técnicas discutidas nesta dissertação no âmbito das ferramentas utilizadas para Educação Musical e modalidades de Ensino à Distância via *internet* com o intuito de reunir informações sobre sua aplicabilidade nestes cenários e validação em estudos de casos que envolvam as áreas da Educação Musical e Computação Musical diretamente. Neste sentido, acredita-se que a evolução de ferramentas de *software* que auxiliem o processo de ensino-aprendizagem de música e o caso particular do solfejo poderá facilitar o compartilhamento do tempo de um único pro-

fessor de solfejo para diversos alunos que assistem as aulas à distância e ainda possuem a chance de realizar uma autoavaliação apoiada por *software*.

Seria interessante explorar outras arquiteturas de redes neurais que pudessem melhorar a acurácia da *tarefa de classificação* de notas musicais e explorar técnicas automáticas de escolha de hiperparâmetros como, por exemplo, *Grid Search*, o qual permite testar diversas combinações de hiperparâmetros de forma automatizada (GÉRON, 2017).

Por fim, a área de *Music Information Retrieval* (MIR) possui diversas tarefas que dependem umas das outras e, neste caso, a tarefa de detecção de *onsets* merece um destaque especial por ser um dos pré-requisitos quando se deseja realizar transcrição melódica, segmentação de áudio e outras tarefas afins. Existe portanto, uma oportunidade de explorar o uso de *Deep Learning* para a detecção de *onsets* para completar a tarefa de transcrição de melodia em voz humana cantada.

REFERÊNCIAS

ALPAYDIN, E. **Introduction to Machine Learning**. London, England: MIT Press, 2010. (Adaptive Computation and Machine Learning series). ISBN 9780262012430. Disponível em: <https://books.google.com.br/books?id=TtrxCwAAQBAJ>.

ANSI, Bioacoustical Terminology. Ansi s3. 20-1995 (r2003). **American National Standards Institute, New York**, 1995. Disponível em: <https://webstore.ansi.org/standards/asa/ansis3201995r2003>.

BENWARD, B.; SAKER, M. N. **Music in theory and practice**. McGraw-Hill, 2009. v. 8. Disponível em: <https://www.mheducation.com/prek-12/product/music-theory-practice-volume-1-marilyn-saker-bruce-benward/9780078025150.html>.

BITTNER, R. M. et al. Deep salience representations for f0 estimation in polyphonic music. In: **ISMIR**. [s.n.], 2017. p. 63–70. Disponível em: <https://bmcfee.github.io/papers/ismir2017_salience.pdf>.

_____. Medleydb: A multitrack dataset for annotation-intensive mir research. In: **ISMIR**. [s.n.], 2014. v. 14, p. 155–160. Disponível em: <https://www.researchgate.net/publication/265508421_MedleyDB_A_Multitrack_Dataset_for_Annotation-Intensive_MIR_Research>.

BREANDAN. **Automação Seja Louvada (versão traduzida para português)**. 2019. Disponível em: <http://softwarelivre.org/piratas/blog/automacao-seja-louvada>. Acesso em: 06 de dez. 2019.

BROWNLEE, J. **Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning**. Machine Learning Mastery, 2017. Disponível em: <https://machinelearningmastery.com/cnn-long-short-term-memory-networks/>.

CHEVEIGNÉ, A. D.; KAWAHARA, H. Yin, a fundamental frequency estimator for speech and music. **The Journal of the Acoustical Society of America**, Acoustical Society of America, v. 111, n. 4, p. 1917–1930, 2002. Disponível em: <http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf>.

COSTA, Y. M.; OLIVEIRA, L. S.; SILLA JR, C. N. An evaluation of convolutional neural networks for music classification using spectrograms. **Applied soft computing**, Elsevier, v. 52, p. 28–38, 2017. Disponível em: <https://dx.doi.org/10.1016/j.asoc.2016.12.024>.

EVERITT, B. **The Cambridge dictionary of statistics in the medical sciences**. Cambridge University Press, 1995. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1167783/>.

FABBRI, S. et al. Improvements in the start tool to better support the systematic review process. In: ACM. **Proceedings of the 20th International Conference on**

Evaluation and Assessment in Software Engineering. 2016. p. 21. Disponível em: <https://dl.acm.org/citation.cfm?id=2916013>.

_____. **StArt LAPES Tool**. 2016. 21 p. Disponível em: <http://lapes.dc.ufscar.br/tools/start_tool>. Acesso em: 08 de dez. 2019.

FAWAZ, H. I. et al. Data augmentation using synthetic data for time series classification with deep residual networks. **CoRR**, abs/1808.02455, 2018. Disponível em: <http://arxiv.org/abs/1808.02455>.

FISCHER, D.; JUBILUT, P. **Música: Ondas Mecânicas que se transformam em Arte**. 2019. Disponível em: <https://blog.biologiatotal.com.br/musica-ondas-mecanicas/>. Acesso em: 20 de abr. 2020.

FISHER, R. **Iris Dataset**. 1996. Disponível em: <https://archive.ics.uci.edu/ml/datasets/Iris>. Acesso em: 17 de nov. 2019.

FJC. George h. king mini-biography. 2015. Disponível em: <https://www.fjc.gov/history/judges/king-george-h>. Acesso em: 08 de dez. 2019.

GERHARD, D. Pitch-based acoustic feature analysis for the discrimination of speech and monophonic singing. **Canadian Acoustics**, v. 30, n. 3, p. 152–153, 2002. Disponível em: <https://jcaa.caa-aca.ca/index.php/jcaa/article/view/1500>.

GÉRON, A. **Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems**. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media, Inc., 2017. Disponível em: <https://books.google.com.br/books?id=bRpYDgAAQBAJ>.

GIANNAKOPOULOS, T. pyaudioanalysis: An open-source python library for audio signal analysis. **PloS one**, Public Library of Science, Athens, 15310, Greece, v. 10, n. 12, p. e0144610, 2015. Disponível em: <https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0144610&type=printable>.

GOODFELLOW, I. **Deep Learning/Ian Goodfellow, Yoshua Bengio, Aaron Courville**. MIT Press, 2016. Disponível em: <http://www.deeplearningbook.org/>.

GOOGLE. **Brain Team**. 2019. Disponível em: <https://ai.google/research/teams/brain/>. Acesso em: 06 de out. 2019.

_____. **Magenta**. 2019. Disponível em: <https://magenta.tensorflow.org>. Acesso em: 06 de out. 2019.

_____. **Melody RNN**. 2019. Disponível em: <https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn>. Acesso em: 04 de dez. 2019.

_____. **MusicVAE**. 2019. Disponível em: <https://magenta.tensorflow.org/music-vae>. Acesso em: 06 de out. 2019.

_____. **NSynth Trainning**. 2019. Disponível em: <https://github.com/tensorflow/magenta/tree/master/magenta/models/nsynth>. Acesso em: 19 de out. 2019.

_____. **Onsets and Frames**. 2019. Disponível em: <https://magenta.tensorflow.org/nsynth>. Acesso em: 06 de out. 2019.

HAWTHORNE, C. et al. Onsets and frames: Dual-objective piano transcription. **arXiv preprint arXiv:1710.11153**, 2018. Disponível em: <https://arxiv.org/abs/1710.11153>.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2007.

HIPKE, K. et al. Beatbox: end-user interactive definition and training of recognizers for percussive vocalizations. In: ACM. **Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces**. 2014. p. 121–124. Disponível em: <https://dl.acm.org/citation.cfm?doid=2598153.2598189>.

HOLLIS, B. **Physics of Sound**. 2017. Disponível em: <https://method-behind-the-music.com/mechanics/physics/>. Acesso em: 20 de abr. 2020.

HOPKIN, B. **Musical instrument design: Practical information for instrument making**. [S.l.]: See sharp press, 1996.

KING, G. H. Happy birthday to you copyright decision by george h. king. 2015. Disponível em: <https://ia800204.us.archive.org/32/items/gov.uscourts.cacd.564772/gov.uscourts.cacd.564772.244.0.pdf>. Acesso em: 08 de dez. 2019.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. Citeseer, Durham,UK, 2007.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–44, 05 2015. Disponível em: <https://www.researchgate.net/publication/277411157_Deep_Learning>.

LEE, D. Hornbostel-sachs classification of musical instruments. **Knowledge Organization**, Nomos Verlagsgesellschaft, v. 47, n. 1, p. 72–91, 2019. Disponível em: <https://www.isko.org/cyclo/hornbostel>.

MARY, C. f. D. M. Q. **VAMP Plugins**. 2012. Disponível em: <https://vamp-plugins.org/>. Acesso em: 17 de nov. de 2019.

MAUCH, M. et al. Computer-aided melody note transcription using the tony software: Accuracy and efficiency. In: **Proceedings of the First International Conference**

on Technologies for Music Notation and Representation. [s.n.], 2015. Accepted. Disponível em: <https://code.soundsoftware.ac.uk/attachments/download/1423/tony-paper_preprint.pdf>.

MAUCH, M.; DIXON, S. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In: **Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014)**. [s.n.], 2014. In press. Disponível em: <http://matthiasmauch.de/_pdf/mauch_pyin_2014.pdf>.

MCFEE, B. et al. librosa: Audio and music signal analysis in python. In: **Proceedings of the 14th python in science conference**. [s.n.], 2015. v. 8. Disponível em: <https://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf>.

MEERT, W. et al. **wannesm/dtaidistance v1.2.2**. Zenodo, 2019. Disponível em: <https://doi.org/10.5281/zenodo.3276100>.

MELO FILHO, E. D. N. Ensino de música a distância: Análise de softwares de edição e criação musical. **Conservatório de Artes e Música de Brasília**, 2017. Disponível em: <http://www.abed.org.br/congresso2017/trabalhos/pdf/156.pdf>.

MIR. **What is Music Information Retrieval?.** 2019. Disponível em: <https://musicinformationretrieval.com/why_mir.html>. Acesso em: 18 de ago. de 2019.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072. Disponível em: <https://dl.acm.org/citation.cfm?id=541177>.

MITROVIĆ, D.; ZEPPELZAUER, M.; BREITENEDER, C. Features for content-based audio retrieval. In: **Advances in computers**. Elsevier, 2010. v. 78, p. 71–150. Disponível em: <https://www.researchgate.net/publication/233740207_Features_for_Content-Based_Audio_Retrieval>.

MOLINA, E. et al. Evaluation framework for automatic singing transcription. 2014. Disponível em: <http://www.atic.uma.es/ismir2014singing/Singing_evaluation_Framework_ISMIR2014_Paper.pdf>.

MUSESCORE. **MuseScore**. 2020. Disponível em: <https://musescore.org/pt-br>. Acesso em: 09 de abr. 2020.

NAKAGAWA, E. Y. et al. **Revisão sistemática da literatura em Engenharia de Software: teoria e prática**. Rio de Janeiro: Elsevier Brasil, 2017.

OORD, A. v. d. et al. Wavenet: A generative model for raw audio. **arXiv preprint arXiv:1609.03499**, 2016. Disponível em: <https://www.deepmind.com/blog/article/wavenet-generative-model-raw-audio>.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Disponível em: <https://scikit-learn.org/stable/about.html#citing-scikit-learn>.

POLINER, G. E.; ELLIS, D. P. A classification approach to melody transcription. 2005. Disponível em: <https://academiccommons.columbia.edu/doi/10.7916/D81Z4DT7>.

PONS, J. et al. Timbre analysis of music audio signals with convolutional neural networks. In: IEEE. **2017 25th European Signal Processing Conference (EUSIPCO)**. [S.l.], 2017. p. 2744–2748.

RAFFEL, C. **Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching**. Tese (Doutorado) — Columbia University, 2016. Disponível em: <https://academiccommons.columbia.edu/doi/10.7916/D8N58MHV>.

RAFFEL, C. et al. mir_eval: A transparent implementation of common mir metrics. In: CITESEER. **In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR**. 2014. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.722.1267>.

RASCHKA, S. **Python machine learning**. Packt Publishing Ltd, 2015. Disponível em: <https://sebastianraschka.com/books.html>.

REPSOLD, M. Tecnologias da informação e comunicação no ensino de música na educação básica: iniciando uma revisão bibliográfica. **Anais do SIMPOM**, v. 5, n. 5, 2018. Disponível em: <http://www.seer.unirio.br/index.php/simpom/article/download/7721/6672>.

RIGAUD, F.; RADENEN, M. Singing voice melody transcription using deep neural networks. In: . [s.n.], 2016. Disponível em: <https://www.researchgate.net/publication/304011421_Singing_Voice_Melody_Transcription_Using_Deep_Neural_Networks>.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. In: RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, C. P. R. (Ed.). Cambridge, MA, USA: MIT Press, 1986. cap. Learning Internal Representations by Error Propagation, p. 318–362. ISBN 0-262-68053-X. Disponível em: <http://dl.acm.org/citation.cfm?id=104279.104293>.

SAK, H.; SENIOR, A.; BEAUFAYS, F. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. **arXiv preprint arXiv:1402.1128**, 2014. Disponível em: <https://arxiv.org/abs/1402.1128>.

SAKOE, H.; CHIBA, S. Dynamic programming algorithm optimization for spoken word recognition. **IEEE transactions on acoustics, speech, and signal processing**, IEEE, v. 26, n. 1, p. 43–49, 1978. Disponível em: <https://ieeexplore.ieee.org/abstract/document/1163055/>.

SALAMON, J. J. **MELODIA**. 2013. Disponível em: <https://www.upf.edu/web/mtg/melodia>. Acesso em: 08 de dez. de 2019.

SALAMON, J. J. **Melody extraction from polyphonic music signals**. Tese (Doutorado) — Universitat Pompeu Fabra, 2013. Disponível em: <http://www.justinsalamon.com/uploads/4/3/9/4/4394963/jsalamon_phdthesis.pdf>.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, July 1959. ISSN 0018-8646. Disponível em: <https://ieeexplore.ieee.org/document/5392560>.

SCHÖRKHUBER, C.; KLAPURI, A. Constant-q transform toolbox for music processing. In: . [s.n.], 2010. Disponível em: <https://iem.kug.ac.at/fileadmin/media/iem/projects/2010/smc10_schoerkhuber.pdf>.

SCHRAMM, R. Sistema audio visual para análise de solfejo. Porto Alegre, 2015. Disponível em: <https://www.lume.ufrgs.br/handle/10183/122533>.

_____. Solfège30 dataset, sistema audio visual para análise de solfejo. 2015. Disponível em: <http://www.inf.ufrgs.br/~rschramm/projects/music/solfege/>. Acesso em: 08 de dez. 2019.

SENAC, C. et al. Music feature maps with convolutional neural networks for music genre classification. In: . [s.n.], 2017. p. 1–5. Disponível em: <https://doi.org/10.1145/3095713.3095733>.

SERRA, X. A system for sound analysis/transformation/synthesis based on a deterministic plus stochastic decomposition. 1989. Disponível em: <http://mtg.upf.edu/content/serra-PhD-thesis>.

SHOU, Y.; MAMOULIS, N.; CHEUNG, D. W. Fast and exact warping of time series using adaptive segmental approximations. **Machine Learning**, Springer, v. 58, n. 2-3, p. 231–267, 2005. Disponível em: <https://link.springer.com/article/10.1007/s10994-005-5828-3>.

SIMSEKLI, U. **Bayesian methods for real-time pitch tracking**. Tese (Doutorado) — Master's thesis, Bogaziçi University, Istanbul, Turkey, 2010. Disponível em: <https://perso.telecom-paristech.fr/simsekli/resources/UmutSimsekliMSThesis.pdf>.

STOWELL, D. **Making music through real-time voice timbre analysis: machine learning and timbral control**. Tese (Doutorado) — Queen Mary University of London, 2010. Disponível em: <https://theses.eurasip.org/theses/358/making-music-through-real-time-voice-timbre/download/>.

TSIPORKOVA, E. Dynamic time warping (dtw). 2006. Disponível em: <https://www.psb.ugent.be/cbd/papers/gentxwarper/DTWAlgorithm.ppt>.

VAIL, M. **The synthesizer: a comprehensive guide to understanding, programming, playing, and recording the ultimate electronic music instrument**. Oxford University Press, 2014. Disponível em: <https://global.oup.com/us/companion.websites/9780195394894/>.

VILELA, I. Vem viola, vem cantando. **Estudos Avançados**, scielo, v. 24, p. 323 – 347, 00 2010. ISSN 0103-4014. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-40142010000200021&nrm=iso>.

WAIKATO, U. of. **WEKA**. 2019. Disponível em: <https://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 17 de nov. de 2019.

WIDROW, B.; KOLLÁR, I. Quantization noise. **Cambridge University Press**, v. 2, p. 5, 2008. Disponível em: <http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521886710&ss=exc>.

WIKIPEDIA. **Portal da Música**. 2019. Disponível em: <https://pt.wikipedia.org/wiki/M\%C3\%BAsica>. Acesso em: 03 de dez. 2019.

WILKINS, J. et al. Vocalset: A singing voice dataset. In: **ISMIR**. [s.n.], 2018. p. 468–474. Disponível em: <http://ismir2018.ircam.fr/doc/pdfs/114_Paper.pdf>.

WOLFE, J. Note names, midi numbers and frequencies. **Diambil tanggal**, v. 20, 2005. Disponível em: <https://newt.phys.unsw.edu.au/jw/notes.html>. Acesso em: 16 de abr. 2010.

XINGJIAN, S. et al. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In: **Advances in neural information processing systems**. [s.n.], 2015. p. 802–810. Disponível em: <http://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting>.

APÊNDICE A – REVISÃO SISTEMÁTICA DA LITERATURA

A.1 Protocolo de Revisão

O processo de revisão sistemática é iterativo e está divido em diferentes fases que são: planejamento da revisão, condução da revisão e sumarização da revisão. Mais informações sobre estes conceitos e o uso de critérios PICO (*Population*, *Intervention*, *Comparison* e *Outcome*), bem como outros aspectos mais aprofundados sobre revisão sistemática aplicada à Engenharia de Software poderão ser consultados em Kitchenham (2004, pág. 12) e Felizardo et al. (2017).

O software gratuito *StArt LAPES* (FABBRI et al., 2016b) desenvolvido por Fabbri et al. (2016a) foi utilizado para auxiliar na criação do protocolo de pesquisa e o resultado será apresentado nos parágrafos seguintes.

Na fase de Planejamento da revisão foram definidos os seguintes elementos:

• Objetivo – O objetivo desta pesquisa é identificar quais *features* são mais relevantes para a identificação de melodias, harmonias e ritmos musicais baseados em processamento automático de áudio através de algoritmos de *Machine Learning* e realizar um experimento através da criação de um protótipo de software capaz de auxiliar no processo de composição musical com base nas *features* identificadas no estudo e nas técnicas de Recuperação de Informações Musicais (MIR) disponíveis.

• Principal Questão de Pesquisa – Quais as principais características ou descritores de áudio (*features*) presentes em amostras sonoras de uma música que influenciam no desempenho dos algoritmos de Machine Learning para uma dada tarefa de classificação musical por aproximação melódica, harmônica ou rítmica? Para este estudo, os seguintes critérios PICO foram adotados:

◦ Population (ou População) - Estudos realizados na área de recuperação de informações musicais (MIR) e processamento de áudio com o uso de algoritmos de *Machine Learning*.

◦ Intervention (ou Intervenção) - Estudos e experimentação com diversas amostras de áudio para classificação de características musicais à partir do uso de algoritmos de *Machine Learning*.

◦ Comparison (ou Controle) – Os estudos escolhidos nesta dissertação para efeitos de comparação tem como base o uso da voz humana cantada ou técnicas de Solfejo assim

como descritas por Schramm (2015a), Salamon (2013b), Bittner et al. (2017), Giannakopoulos (2015), McFee et al. (2015), Stowell (2010) e outros estudos reunidos na Pesquisa Exploratória.

◦ Outcome (ou Resultados) - Estudos primários sobre o uso de algoritmos de *Machine Learning* aplicados em computação musical e sugestão de um processo ou método de engenharia de software para abordagem de processamento de áudio com o uso de *Machine Learning*.

Uma extensão adicional dos critérios PICO elencados nos parágrafos anteriores foi utilizado para compor parte das informações solicitadas pelo software *StArt LAPES* (FABBRI et al., 2016b). Esta extensão é chamada Application e está descrita a seguir:

◦ Application (ou Aplicação) - O resultado desta revisão sistemática poderá fornecer uma visão inicial sobre o assunto aos desenvolvedores de softwares que precisem projetar sistemas de processamento de áudio com base em algoritmos de *Machine Learning*, além de identificar as principais técnicas em uso na área da computação musical, especificamente, em Recuperação de Informações Musicais (do inglês Music Information Retrieval – MIR).

A.1.1 Palavras Chaves

As palavras chaves, sinônimos e principais termos de busca utilizados foram:

- *Audio Feature Extraction*
- *Content-Based Audio Retrieval*
- *Machine Learning Algorithm and Music*
- *Music Information Retrieval*
- *Music Transcription Machine Learning*
- *Music and Artificial Intelligence*
- *Music composition with Machine Learning*
- *Music concatenation*
- *Music feature extraction*
- *Musical Instrument Recognition*
- *Musical characteristics and structure*
- *Pattern Processing in Melodic Sequences*
- *Readings in Music and Artificial Intelligence*
- *Solfege (Solfége)* ou Solfejo em português

- *Unsupervised feature learning for music*
- *Deep Learning*
- *Analysis of music signals*
- *Chord-scale detection machine learning*
- *Humming music*
- *Humming music machine learning*
- *Machine learning timbral control*
- *Query by humming music*
- *Music score machine learning*
- *Speech music discrimination*

A.1.2 Critério de Seleção de Origem de Estudos Primários

- Bibliotecas digitais on-line (como IEEE Xplore, ACM, etc.)
- Bases eletrônicas indexadas (como Scopus-Mendeley, Google Scholar, etc.)
- Simpósios Nacionais e Internacionais em Computação Musical
- Teses e Dissertações de Universidades Nacionais e Internacionais que possuam grupos de pesquisas ou programas de graduação, mestrado ou doutorado na área de Computação Musical e Aprendizagem de Máquina (USP, UFBA, UPF-Barcelona, etc.)
- Buscas específicas por artigos e temas relacionados simultaneamente à Ciência da Computação, Computação Musical, *Machine Learning* e Música

A.1.3 Idioma de preferência para os estudos

Os principais estudos considerados devem estar em língua inglesa para facilitar a busca de abstracts (resumos) publicados neste idioma.

Estudos em português que estejam relacionados a Computação Musical que foram publicados em Simpósios aqui no Brasil.

A.1.4 Estratégia de busca

Em cada sessão de pesquisa, os termos de busca (strings) deverão ser informados nas respectivas bases, anais e repositórios de artigos científicos indicados como origens para a revisão sistemática.

Para os artigos que abordam experimentos e métodos específicos da área de Music Information Retrieval (MIR) com o uso de algoritmos de *Machine Learning*, as referências bibliográficas utilizadas pelo artigo em questão que forem mais utilizadas ou fornecerem maior embasamento teórico serão pesquisadas também através da técnica manual conhecida como "Pearl Growing" que consiste em analisar recursivamente as referências bibliográficas listadas em um artigo ou publicação científica.

No caso do Google Scholar (Google Search), em caráter de exceção, foi utilizada a expressão "dissertação *machine learning* + música + espectrograma".

A.1.5 Lista de Fontes de Pesquisa (Origens)

- IEEE Xplore Digital Library ACM
- Scopus
- ACM Digital Library
- ScienceDirect
- Web of Science
- Oasisbr
- Google/Google Scholar

A.1.6 Critérios de Seleção de Estudos (Inclusão)

Os Critérios de Inclusão adotados foram:

(I) Artigos, Teses e Dissertações Publicados em Bibliotecas Digitais, Revistas, Congressos, Anais ou Eventos similares na área de Computação Musical e *Machine Learning*

(II) Capítulos preliminares de livros recém-lançados (intervalo de 5 anos) ou previstos para lançamento futuro que tenham sido aceitos e publicados em Bibliotecas Digitais, Revistas, Congressos, Anais ou Eventos similares na área de Computação Musical e *Machine Learning*

(III) Estudos que indiquem o uso de análise, extração, processamento ou classificação automática de áudio/música com o auxílio de algoritmos de *Machine Learning* ou *Deep Learning*

(IV) Estudos que apresentem experimentos com sintetizadores ou samplers de instrumentos musicais e interface de controle por voz utilizando *Machine Learning*

(V) Artigos que atendam algum dos critérios de inclusão e apresentem ferramentas baseadas em Python, Scikit-Learn, C/C++ ou Tensorflow na área de *Music Information Retrieval*

Como parâmetro geral de avaliação da Qualidade dos Estudos Primários, determina-se que os estudos elencados tenham sido publicados em revistas científicas, periódicos, anais, repositórios acadêmicos ou eventos científicos reconhecidos nacionalmente ou internacionalmente no meio acadêmico.

Estes, deverão apresentar em sua completude e estrutura: início, desenvolvimento e conclusões, além de demonstrar experimentos relacionados a esta pesquisa (quando aplicáveis).

A.1.7 Critérios de Exclusão

Os Critérios de Exclusão adotados foram:

(VI) Quando os critérios de inclusão não forem suficientes, estudos com score (*Start Lapes*) inferior a 10 pontos serão descartados.

(VII) Artigos, Teses ou Dissertações que não utilizem técnicas de *Machine Learning* (ou *Deep Learning*)

(VIII) Artigos, Teses ou Dissertações que não sejam disponibilizados gratuitamente via convênios acadêmicos com Universidades e Institutos de Pesquisa ou na Internet

(IX) Artigos, Teses ou Dissertações que não utilizem tecnologias baseadas em códigos-abertos para os algoritmos de *Machine Learning* e recuperação de informações musicais

(X) Estudos que não atendam os critérios mínimos de qualidade exigidos nesta revisão sistemática

(XI) Estudos com abordagens repetidas ou similares em relação aos resultados já obtidos nesta revisão sistemática

(XII) Textos que não sejam Artigos, Teses ou Dissertações, Novos Capítulos de Livros

(XIII) Estudos anteriores ao ano de 2010 (exceto em casos de Pearl Growing)

(XIV) Os estudos que não apresentarem os termos "*Music Information Retrieval* (MIR)" ou equivalente e "*Machine Learning*" ou "*Deep Learning*" simultaneamente no corpo do texto serão excluídos

(XV) Artigo em duplicidade

(XVI) Idioma usado no artigo difere do Inglês ou Português

A.1.8 Definição dos Tipos de Estudos

A Revisão Sistemática Qualitativa foi adota como critério.

A.1.9 Estratégia para seleção dos estudos

Os critérios de Inclusão e Exclusão serão utilizados na seleção inicial.

Na primeira etapa, serão lidos os títulos e resumos (abstracts) ou primeiro parágrafo da introdução quando o resumo não estiver disponível ou for insuficiente para classificar o texto com base nos critérios de inclusão e exclusão especificados.

Na segunda etapa, aqueles artigos que apresentarem maior número de critérios de inclusão serão selecionados para leitura completa.

Finalmente, inicia-se a extração e tabulação dos dados quando a principal pergunta de pesquisa já tiver sido respondida pela leitura completa e gradual dos artigos selecionados, respeitada a ordem de leitura, na segunda etapa.

A.1.10 Campos para o Formulário de Extração de Dados

Para atender as solicitações de preenchimento do software *StArt LAPES* (FABBRI et al., 2016b), os seguintes campos foram informados:

- Área = [Inteligência Artificial, *Machine Learning*, Computação Musical e *Music Information Retrieval*]
- Problema = [Extração de *feature*s em áudios musicais, classificação, regressão]
- Modelos = [Linear Regression, Logistic Regression, SVM, Naïve Bayes, Perceptron, Redes Neurais, K-Means, kNN, Autoencoders, DNN, CNN, RNN, Decision Trees]
- Representação = [MFCC, Constant-Q, RMS, FFT]

Alguns dos itens indicados no campo "Modelos" poderão ser suprimidos do experimento final no decorrer da elaboração desta dissertação, assim como outros itens poderão ser acrescentados.

A.1.11 Resultados esperados na Sumarização

Após a seleção e extração dos dados mais relevantes obtidos a partir dos estudos primários selecionados, será realizada uma síntese com as *features* de áudio mais utilizadas.

A.1.12 Seleção e Avaliação dos Artigos

Na fase de Condução da Revisão foram identificados 14.315 artigos resultantes dos termos de busca indicados na seção anterior para todas as bases de pesquisa relacionadas. Neste conjunto inicial de artigos encontrados, 7158 resultados estavam duplicados, principalmente, em relação à base da ACM, sendo assim, efetivamente foram retornados um total de 7.157 artigos.

Deste total, depois de aplicar os critérios de inclusão e exclusão foram selecionados 9 artigos, a maioria foi descartada ao aplicar o critério de pontuação (Score) parametrizado pelo autor e atribuído pelo software *StArt LAPES* (FABBRI et al., 2016b) que rejeitou artigos com pontuação inferior a 10 pontos (vide critério de exclusão VI).

Por fim, na fase de Extração, 2 artigos foram rejeitados após leitura completa dos artigos. Assim, foram extraídos o total de 7 artigos para sumarização e uso efetivo na dissertação.

A.2 Condução da Revisão

Neste capítulo da Revisão Sistemática, em decorrência do grande número de artigos encontrados considerando-se todas as bases de pesquisa utilizadas, será apresentada apenas um piloto inicial restrito às buscas realizadas na base do IEEE Xplore e Mendeley, onde um total de 203 artigos foram identificados. Desse total, apenas as strings mais relevantes serão mostradas conforme listagens descritas nas subseções A.1.14.1 e A.1.14.2.

Para consultar o resultado completo, favor consultar o arquivo *StArt LaPES* disponibilizado em companhia desta dissertação.

A.2.1 Fonte 1 – IEEE Xplore – Busca 004

Fonte: IEEE Xplore Digital Library
Data de Busca: 07/11/2018
Strings Utilizadas – Em inglês: “Music Feature Extraction”
Campos Pesquisados: Resumo (Abstract)
Período Considerado: Janeiro de 2010 a Novembro de 2018
Filtros Utilizados: “Metadata Only”

Lista de Artigos Encontrados:

1.C. Kaur and R. Kumar, “Study and analysis of feature based automatic music genre classification using Gaussian mixture model”, 2017 International Conference on Inventive Computing and Informatics (ICICI), 2017, pp. 465-468. DOI: 10.1109/ICICI.2017.8365395, URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8365395>

2.R. H. D. Zottesso and Y. M. G. Costa and D. Bertolini, “Music genre classification using visual features with feature selection”, 2016 35th International Conference of the Chilean Computer Science Society (SCCC), 2016, pp. 1-6. DOI: 10.1109/SCCC.2016.7836004, URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7836004>

3.Z. Gao and Y. Liu and Y. Jiang, “An effective method on content based music feature extraction”, 2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2015, pp. 780-784. DOI: 10.1109/IAEAC.2015.7428662, URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7428662>

4.S. G. Tanyer, “Feature extraction for music signals: Just tuned, equal temperament and intense diatonic systems”, 2014 22nd Signal Processing and Communications Applications Conference (SIU), 2014, pp. 277-280. DOI: 10.1109/SIU.2014.6830219, URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6830219>

5.R. C. P. Kumar and D. A. Chandy, “Audio retrieval using timbral feature”, 2013 IEEE International Conference ON Emerging Trends in Computing, Communication and Nanotechnology (ICECCN), 2013, pp. 222-226. DOI: 10.1109/ICE-CCN.2013.6528497, URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6528497>

6.Da-chuan Wei, “An improved feature extraction algorithm of humming music”, Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), 2011, pp. 2500-2503. DOI: 10.1109/TMEE.2011.6199729, URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6199729>

7.Huiyu Zhou and A. Sadka and R. M. Jiang, “Feature extraction for speech and music discrimination”, 2008 International Workshop on Content-Based Multimedia Indexing, 2008, pp. 170-173. DOI: 10.1109/CBMI.2008.4564943, URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4564943>

Os artigos e seus respectivos status de inclusão ou exclusão estão listados no Quadro 6.

Quadro 6 – Protocolo de Pesquisa - IEEE Seleção Preliminar - Busca 004

Artigo	**Critérios de Inclusão**	**Critérios de Exclusão**	**Estado**
1	(III)	n/a	Incluído
2	(III)	(VI)(XIV)	Excluído
3	(III)	n/a	Incluído
4	n/a	(VI)(XVI)	Excluído
5	(III)	(VI)(XI)	Excluído
6	(III)(IV)	n/a	Incluído
7	n/a	(VI)(VII)	Excluído

Fonte: Elaborado pelo autor.

A.2.2 Fonte 5 (Indexador) – Mendeley – Busca 017

Indexador de Pesquisa: Mendeley

Data de Busca: 20/11/2018

Strings Utilizadas – Em inglês: “Audio Features Survey”

Campos Pesquisados: Resumo (Abstract)

Período Considerado: Janeiro de 2010 a Novembro de 2018

Filtros Utilizados: “Default”

Lista de Artigos Encontrados:

8. Mitrovic, Dalibor and Zeppelzauer, Matthias and Breiteneder, Christian, “Features for Content-Based Audio Retrieval”, Advances in Computers, November 2012, 2012, Volume 78. DOI: 10.1016/S0065-2458(10)78003-7, URL: <https://www.ims.tuwien.ac.at/publications/tuw-186351>

Os artigos e seus respectivos status de inclusão ou exclusão estão listados no Quadro 7.

Quadro 7 – Protocolo de Pesquisa - Mendeley Seleção Preliminar - Busca 017

Artigo	**Critérios de Inclusão**	**Critérios de Exclusão**	**Estado**
8	(I)(II)(III)	n/a	Incluído

Fonte: Elaborado pelo autor.

A.3 Seleção Final de Artigos e Análise dos Resultados

Após a extração dos artigos listados no Quadro 8, que foram selecionados para o presente estudo, foi necessário rever alguns critérios de pesquisar e atualizar a lista de artigos finais selecionados com artigos complementares. Analisando estes artigos é possível

verificar que diferentes tipos de *features* podem ser extraídas de um arquivo de áudio levando em conta a tarefa que se deseja realizar.

Até este ponto da dissertação, é possível avaliar que o processo de revisão sistemática contribuiu para que artigos e estudos relevantes na área de *Machine Learning* aplicado a problemas de *Music Information Retrieval* pudessem ser encontrados de forma metódica utilizando critérios definidos de inclusão e exclusão, respondendo à principal pergunta de pesquisa deste estudo, além de possibilitar o embasamento teórico-científico necessário para a realização dos experimentos que serão propostos nesta dissertação.

Assim, respondendo à pergunta de pesquisa, observou-se que os diferentes descritores de áudio (ou *features*) disponíveis em uma amostra de som poderiam ser agrupados de acordo com os *domínios* da aplicação, seguindo uma taxonomia própria para a descrição de áudio como a abordada em Mitrović, Zeppelzauer e Breiteneder (2010), que separa as *features* em *domínios* como: *Temporal*, *Frequência*, *Correlação*, *Cepstral*, *Modulação de Frequência*, *Reconstrução de Fase* e *Autodomínio*.

Quadro 8 – Protocolo de Pesquisa - Seleção de Artigos Preliminares

Artigo	**Título do Artigo**	**Autor(es)**	**Fonte**
1	Study and analysis of feature based automatic music genre classification using Gaussian mixture model	C. Kaur and R. Kumar	IEEE
3	An effective method on content based music feature extraction	Z. Gao and Y. Liu and Y. Jiang	IEEE
6	An improved feature extraction algorithm of humming music	Da-chuan Wei	IEEE
8	Features for Content-Based Audio Retrieval	Mitrovic, Dalibor and Zeppelzauer, Matthias and Breiteneder, Christian	Mendeley

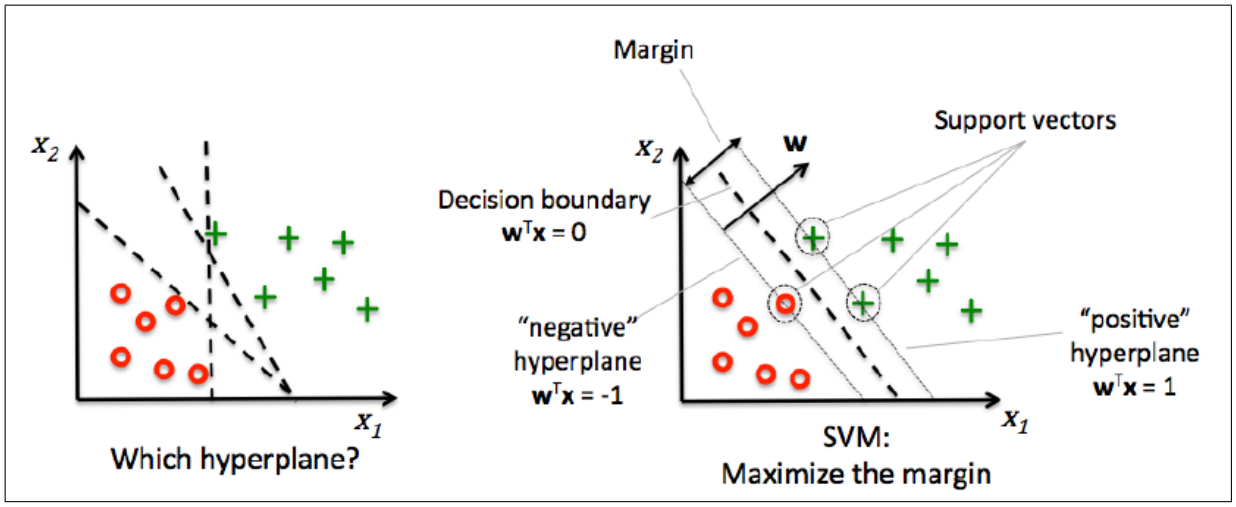
Fonte: Elaborado pelo autor.

APÊNDICE B – OUTROS ALGORITMOS DE *MACHINE LEARNING*

B.1 *Support Vector Machine*

Os algoritmos do tipo *Support Vector Machine (SVM)* pertencem a uma família de modelos baseados na Teoria de Aprendizado Estatístico desenvolvido por Vladimir Vapnik (RASCHKA, 2015, p. 71). Em tarefas de classificação, o principal objetivo desta família de algoritmos é encontrar hiperplanos no espaço *n-dimensional* que permitam separar e classificar corretamente os dados de acordo com a classe a que pertencem, mantendo a máxima distância possível entre os hiperplanos e as amostras. Esta distância máxima entre os hiperplanos é conhecida como *margem* e as amostras mais próximas à *margem* são chamadas de *vetores de suporte* (RASCHKA, 2015, p. 69).

Figura 53 – Hiperplanos no *Support Vector Machine* com *kernel* linear



Fonte: Raschka (2015, p. 69).

A Figura 53 apresenta um exemplo de hiperplano no SVM com *kernel* linear. Os símbolos com circunferências (*em vermelho*) e cruzes (*em verde*) que aparecem na ilustração denotam as amostras de dados pertencentes a classes distintas de um problema de classificação binário. As linhas retas tracejadas representam os hiperplanos que se quer achar para separar as amostras de acordo com as classes corretas, formando a chamada fronteira de decisão (*Decision Boundary*).

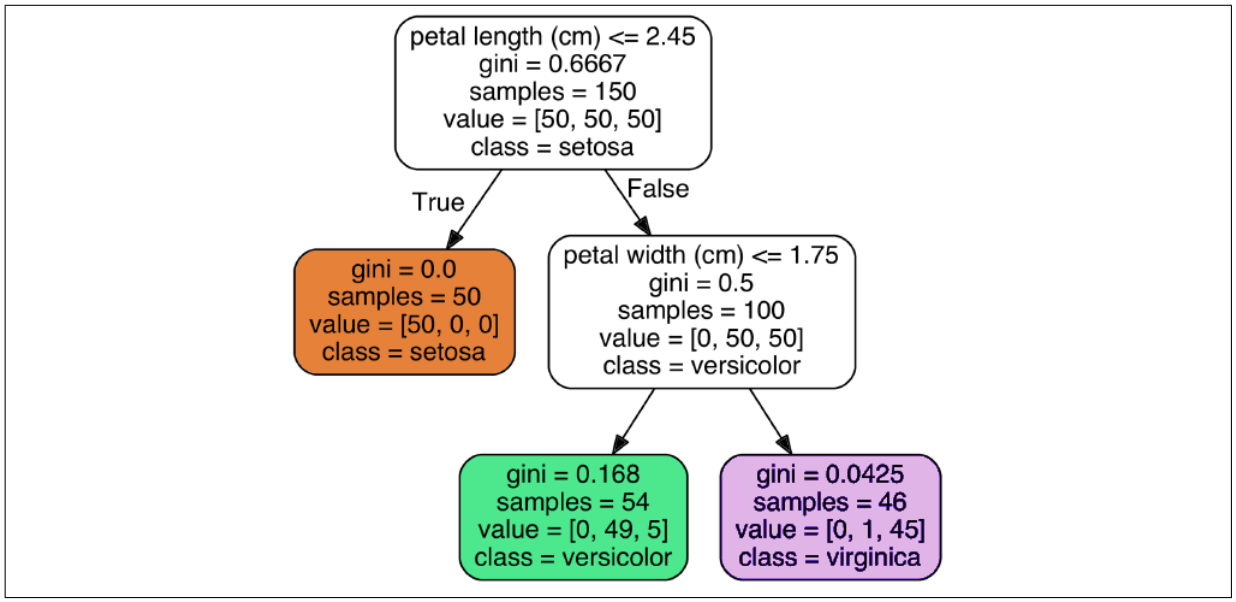
B.2 Árvore de Decisão

A Árvore de Decisão representa uma família de algoritmos de *Machine Learning* bastante versáteis pois conseguem realizar tarefas de classificação e regressão, incluindo a possibilidade de gerar múltiplas saídas, além de manter alto grau de interpretabilidade em relação aos outros modelos (GÉRON, 2017, cap. 6).

A Árvore de Decisão é um algoritmo supervisionado capaz de inferir automaticamente as regras subjacentes a um dado conjunto de amostras. Estas regras podem ser comparadas à estrutura condicional *se..então..senão* comum na programação procedural.

A Figura 54 ilustra uma Árvore de Decisão classificando um conjunto de amostra de flores do tipo *Iris Setosa*, *Versicolor* e *Virgínica* a partir de um *dataset* público disponibilizado pela UCI (FISHER, 1996).

Figura 54 – Árvore de Decisão



Fonte: Géron (2017, p. 168).

O conceito de grau de impureza é utilizado em conjunto com a métrica denominada *gini* (que é normalizada com valores entre 0 e 1, onde 1 representa 100% de probabilidade) para determinar o quão puro são as amostras de uma classe em um nó ou folha da árvore (GÉRON, 2017, cap. 6).

APÊNDICE C – EXEMPLOS DE CÓDIGO-FONTE

Alguns trechos de *códigos-fontes* desenvolvidos para o experimento ou adaptados de outros autores são listados neste apêndice.

C.1 Implementação da rede CNN–Melody

A rede CNN-Melody realiza a extração da melodia principal presente em uma música. Esta rede foi desenvolvida originalmente por Bittner et al. (2017) e seu código-fonte foi adaptado para a proposta desta dissertação.

C.1.1 Trecho de código em Python para criação da CNN–Melody

Configuração da Rede CNN-Melody

```
import keras
from keras.models import Model
from keras.layers import Input, Lambda
from keras.layers.convolutional import Conv2D
from keras.layers.normalization import BatchNormalization
from keras import backend as K

def model_def_001():

    ## DEFINE CNN MODEL ###
    input_shape_cnn = (None, None, 6)
    inputs_cnn = Input(shape=input_shape_cnn, dtype='float32')

    #CNN model
    y1 = Conv2D(128, (5, 5), padding='same', activation='relu', name='
        bendy1')(inputs_cnn)
    y1a = BatchNormalization()(y1)
    y2 = Conv2D(64, (5, 5), padding='same', activation='relu', name='bendy2
        ')(y1a)
    y2a = BatchNormalization()(y2)
    y3 = Conv2D(64, (3, 3), padding='same', activation='relu', name='
        smoothy1')(y2a)
    y3a = BatchNormalization()(y3)
```

```
    y4 = Conv2D(64, (3, 3), padding='same', activation='relu', name='
        smoothy2')(y3a)
    y4a = BatchNormalization()(y4)
    y5 = Conv2D(8, (70, 3), padding='same', activation='relu', name='
        distribute')(y4a)
    y5a = BatchNormalization()(y5)
    y6 = Conv2D(1, (1, 1), padding='same', activation='sigmoid', name='
        squishy')(y5a)
    predictions = Lambda(lambda x: K.squeeze(x, axis=3))(y6)

    cnn_model = Model(inputs=inputs_cnn, outputs=predictions)
    cnn_model.name = 'CNN_Model_Bittner'

    return cnn_model
```

C.2 Implementação da rede LSTM–CQT

A rede LSTM-CQT é responsável pela representação de timbre de instrumento musical similar a um piano.

C.2.1 Trecho de código em Python para criação da LSTM–CQT

Configuração da Rede LSTM–CQT

```
import keras
from keras import backend as K
from keras import Model, Sequential
from keras.layers import Input, Dense, Dropout, RepeatVector, LSTM,
    Bidirectional, Lambda, TimeDistributed, Permute, Reshape, Flatten,
    Concatenate
from keras.layers.normalization import BatchNormalization

def config_models():

    ## DEFINE CNN HCQT INPUT MODEL ###
    input_shape_cnn = (None, None, 6)
    inputs_cnn_hcqt = Input(shape=input_shape_cnn, dtype='float32', name='
```

```
        inputs_cnn_hcqt')
    input_cnn_melody_out = Input(shape=(None, N_BINS), dtype='float32',
        name='input_cnn_melody_out')

    sequence_bilstm_01 = LSTM(latent_dim_02, return_sequences=True,
        activation='softsign', recurrent_activation='tanh', name='
        sequence_bilstm_01')(input_cnn_melody_out)
    sequence_bilstm_02 = TimeDistributed(Dense(N_BINS, activation='relu'),
        name='sequence_bilstm_02')(sequence_bilstm_01)

    adam_opt_lstm = keras.optimizers.adam(learning_rate=0.001, beta_1=0.9,
        beta_2=0.999, amsgrad=True)

    sequence_bilstm = Model(input_cnn_melody_out, sequence_bilstm_02)
    sequence_bilstm.name = 'BiLSTM_Model_CQT'
    sequence_bilstm.compile(optimizer=adam_opt_lstm, loss='mse')
    print(sequence_bilstm.summary(line_length=100))

    #Juntando os models...

    cnn_melody_out = Lambda(lambda x: K.permute_dimensions(
        cnn_trained_model(x),(0,2,1)), name='cnn_melody_out')(
        inputs_cnn_hcqt)
    bilstm_senoidal_cqt = sequence_bilstm(cnn_melody_out)
    sequence_cnn_bilstm = Model(inputs_cnn_hcqt, bilstm_senoidal_cqt)
    sequence_cnn_bilstm.name = 'CNN_BiLSTM_Model_Encoder'
    sequence_cnn_bilstm.compile(optimizer=adam_opt_lstm, loss='mse')
    print(sequence_cnn_bilstm.summary(line_length=100))

    print("Modelo CNN-BiLSTM compilado com sucesso!\n")

(...)
```

C.3 Implementação da rede LSTM–MIDI

A rede LSTM-MIDI é responsável pela identificação de notas musicais presentes nos exercícios de solfejo e possui duas variações de implementação para lidar com classificação

note-byte e *frame-by-frame*.

C.3.1 Trecho de código em Python para criação da LSTM–MIDI

Configuração da Rede LSTM–MIDI

```
import keras
from keras import backend as K
from keras import Model, Sequential
from keras.layers import Input, Dense, Dropout, RepeatVector, LSTM,
    Bidirectional, Lambda, TimeDistributed, Permute, Reshape, Flatten,
    Concatenate
from keras.layers.normalization import BatchNormalization

def config_models():

    ## DEFINE CNN HCQT INPUT MODEL ###
    input_shape_cnn = (None, None, 6)
    inputs_cnn_hcqt = Input(shape=input_shape_cnn, dtype='float32', name='
        inputs_cnn_hcqt')
    input_cnn_melody_out = Input(shape=(None, N_BINS), dtype='float32',
        name='input_cnn_melody_out')

    (...)

    cnn_melody_out = Lambda(lambda x: K.permute_dimensions(
        cnn_trained_model(x),(0,2,1)), name='cnn_melody_out')(
        inputs_cnn_hcqt)
    bilstm_senoidal_cqt = sequence_bilstm(cnn_melody_out)
    sequence_cnn_bilstm = Model(inputs_cnn_hcqt, bilstm_senoidal_cqt)
    sequence_cnn_bilstm.name = 'CNN_BiLSTM_Model_Encoder'
    sequence_cnn_bilstm.compile(optimizer=adam_opt_lstm, loss='mse')
    print(sequence_cnn_bilstm.summary(line_length=100))

    print("Modelo CNN-BiLSTM compilado com sucesso!\n")

  (...)

    #---CNN MIDI Encoder: Nota Musical com Timbre => Onda Senoidal (CQT) =>
        Nota Musical em código MIDI (hot-encoded 2D)---
```

```
28
    #CNN MIDI Model
30  print('Criando CNN-LSTM Decoder (MIDI)...\n')

32  cnn_lstm_midi_in = LSTM(64, return_sequences=False, activation='
        softsign', recurrent_activation='tanh')(bilstm_senoidal_cqt)
    cnn_lstm_midi_out = Dense(128, activation='softmax', name='
        cnn_lstm_midi_out')(cnn_lstm_midi_in)
34
    sequence_cnn_lstm_midi = Model(inputs_cnn_hcqt, cnn_lstm_midi_out)
36  sequence_cnn_lstm_midi.name = 'CNN_LSTM_Decoder_MIDI'
    sequence_cnn_lstm_midi.compile(optimizer='adam', loss='
        sparse_categorical_crossentropy')
38  print(sequence_cnn_lstm_midi.summary(line_length=100))

40  print("Modelo CNN-LSTM Decoder (MIDI) compilado com sucesso!\n")

42  #---CNN Frame Encoder: Nota Musical com Timbre => Onda Senoidal (CQT)
        => Nota Musical em código MIDI para cada Frame (hot-encoded 3D)---

44  #CNN MIDI-Frame Model
    print('Criando CNN-LSTM-Frame Decoder (MIDI)...\n')
46
    #activation='softmax' para classificação binária e activation='sigmoid'
         para multi-classificação.
48  cnn_lstm_midi_frame_in =  TimeDistributed(Dense(360, activation='relu')
        ,     name='cnn_lstm_midi_frame_in')(bilstm_senoidal_cqt)
    cnn_lstm_midi_frame_000 = TimeDistributed(Dense(128, activation='relu')
        ,     name='cnn_lstm_midi_frame_000')(cnn_lstm_midi_frame_in)
50  cnn_lstm_midi_frame_001 = TimeDistributed(Dense(64, activation='relu'),
            name='cnn_lstm_midi_frame_001')(cnn_lstm_midi_frame_000)
    cnn_lstm_midi_frame_002 = TimeDistributed(Dense(24, activation='relu'),
            name='cnn_lstm_midi_frame_002')(cnn_lstm_midi_frame_001)
52  cnn_lstm_midi_frame_out = TimeDistributed(Dense(128, activation='
        softmax'), name='cnn_lstm_midi_frame_out')(cnn_lstm_midi_frame_002)
        #128 features = 128 notas midi

54  cnn_lstm_midi_frame = Model(inputs_cnn_hcqt, cnn_lstm_midi_frame_out)
    cnn_lstm_midi_frame.name = 'CNN_LSTM_MIDI_Frame'
```

```
56      cnn_lstm_midi_frame.compile(optimizer='adam', loss='
            sparse_categorical_crossentropy')
        print(cnn_lstm_midi_frame.summary(line_length=100))
58
        print("Modelo CNN_LSTM_MIDI_Frame compilado com sucesso!\n")
60
    (...)
```

C.4 Implementação da rede LSTM–Wave

A rede LSTM-Wave sintetiza as notas musicais convertendo a representação em CQT do timbre de instrumento musical para áudio bruto (formato Wave) com som similar ao piano.

C.4.1 Trecho de código em Python para criação da LSTM–Wave

Configuração da Rede LSTM–Wave

```
1  import keras
   from keras import backend as K
3  from keras import Model, Sequential
   from keras.layers import Input, Dense, Dropout, RepeatVector, LSTM,
       Bidirectional, Lambda, TimeDistributed, Permute, Reshape, Flatten,
       Concatenate
5  from keras.layers.convolutional import Conv2D, ZeroPadding2D
   from keras.layers.normalization import BatchNormalization
7
   def config_models():
9
       ## DEFINE CNN HCQT INPUT MODEL ###
11     input_shape_cnn = (None, None, 6)
       inputs_cnn_hcqt = Input(shape=input_shape_cnn, dtype='float32', name='
           inputs_cnn_hcqt')
13     input_cnn_melody_out = Input(shape=(None, N_BINS), dtype='float32',
           name='input_cnn_melody_out')

15     (...)
```

```
17    #---LSTM_Wave_Model Decoder : Nota Musical com Timbre => Onda Senoidal
          (CQT) => Wave----

19    latent_dim = 128

21    lstm_midi_out = Lambda(lambda x: K.expand_dims(x, 1))(cnn_lstm_midi_out
          )
      lstm_encoded_01 = LSTM(latent_dim, return_sequences=False, activation='
          tanh', recurrent_activation='sigmoid')(lstm_midi_out)
23    dense_latent_01 = Dense(latent_dim*3, activation='tanh')(
          lstm_encoded_01)
      dense_latent_02 = Dense(latent_dim*5, activation='linear')(
          dense_latent_01)
25    dense_decoded = Dense(4096, activation='linear')(dense_latent_02)

27    lstm_wave_model = Model(inputs_cnn_hcqt, dense_decoded)
      lstm_wave_model.name = 'LSTM_Wave_Model'
29    lstm_wave_model.compile(optimizer='adam', loss='mse')
      print(lstm_wave_model.summary())
31
      print("Modelo LSTM_Wave_Model Decoder compilado com sucesso!\n")
33
  (...)
```

C.5 Implementação da rede CNN–CQT

A rede CNN-CQT foi rede extra criada para testar a detecção de *onsets*, mas não foi integrada à proposta desta dissertação.

C.5.1 Trecho de código em Python para criação da CNN–CQT

Configuração da Rede CNN–CQT

```
1 import keras
  from keras import backend as K
3 from keras import Model, Sequential
```

```
from keras.layers import Input, Dense, Dropout, RepeatVector, LSTM,
    Bidirectional, Lambda, TimeDistributed, Permute, Reshape, Flatten,
    Concatenate
from keras.layers.convolutional import Conv2D, ZeroPadding2D
from keras.layers.normalization import BatchNormalization

def config_models():
    input_cnn_melody_out = Input(shape=(None, N_BINS), dtype='float32',
        name='input_cnn_melody_out')

    cnn_cqt_in = Lambda(lambda x: K.expand_dims(K.expand_dims(x, -1),0),
        name='cnn_cqt_in')(input_cnn_melody_out)
    cnn_cqt_model_01 = TimeDistributed(Conv2D(16, (5, 1), padding='same',
        activation='relu'), name='cnn_cqt_conv2d_01')(cnn_cqt_in)
    cnn_cqt_model_02 = TimeDistributed(BatchNormalization(), name='
        cnn_cqt_bn_01')(cnn_cqt_model_01)
    cnn_cqt_model_03 = TimeDistributed(Conv2D(16, (5, 1), padding='same',
        activation='relu'), name='cnn_cqt_conv2d_02')(cnn_cqt_model_02)
    cnn_cqt_model_04 = TimeDistributed(BatchNormalization(), name='
        cnn_cqt_bn_02')(cnn_cqt_model_03)
    cnn_cqt_model_05 = TimeDistributed(Conv2D(1, (5, 1), padding='same',
        activation='relu'), name='cnn_cqt_conv2d_03')(cnn_cqt_model_04)
    cnn_cqt_model_06 = TimeDistributed(BatchNormalization(), name='
        cnn_cqt_bn_03')(cnn_cqt_model_05)
    cnn_cqt_model_07 = Lambda(lambda x: K.squeeze(x, axis=-1), name='
        cnn_cqt_lambda_01')(cnn_cqt_model_06)
    cnn_cqt_model_out = Lambda(lambda x: K.squeeze(x, axis=0), name='
        cnn_cqt_model_out')(cnn_cqt_model_07)

    cnn_cqt_model = Model(input_cnn_melody_out, cnn_cqt_model_out)
    cnn_cqt_model.name = 'CNN_CQT_Model'
    cnn_cqt_model.compile(optimizer='adam', loss='mse')
    print(cnn_cqt_model.summary(line_length=100))

  (...)
```

APÊNDICE D – DADOS E RESULTADOS DO EXPERIMENTO

O subconjunto de dados extraídos do *dataset VocalSet* para o experimento e os resultados práticos obtidos são apresentados neste apêndice.

Para facilitar o entendimento sobre o uso dos dados no decorrer do experimento, as figuras serão apresentadas em ordem sequencial de acordo com as entradas e saídas de dados nos diversos componentes da arquitetura proposta na seção 3, Figura 23.

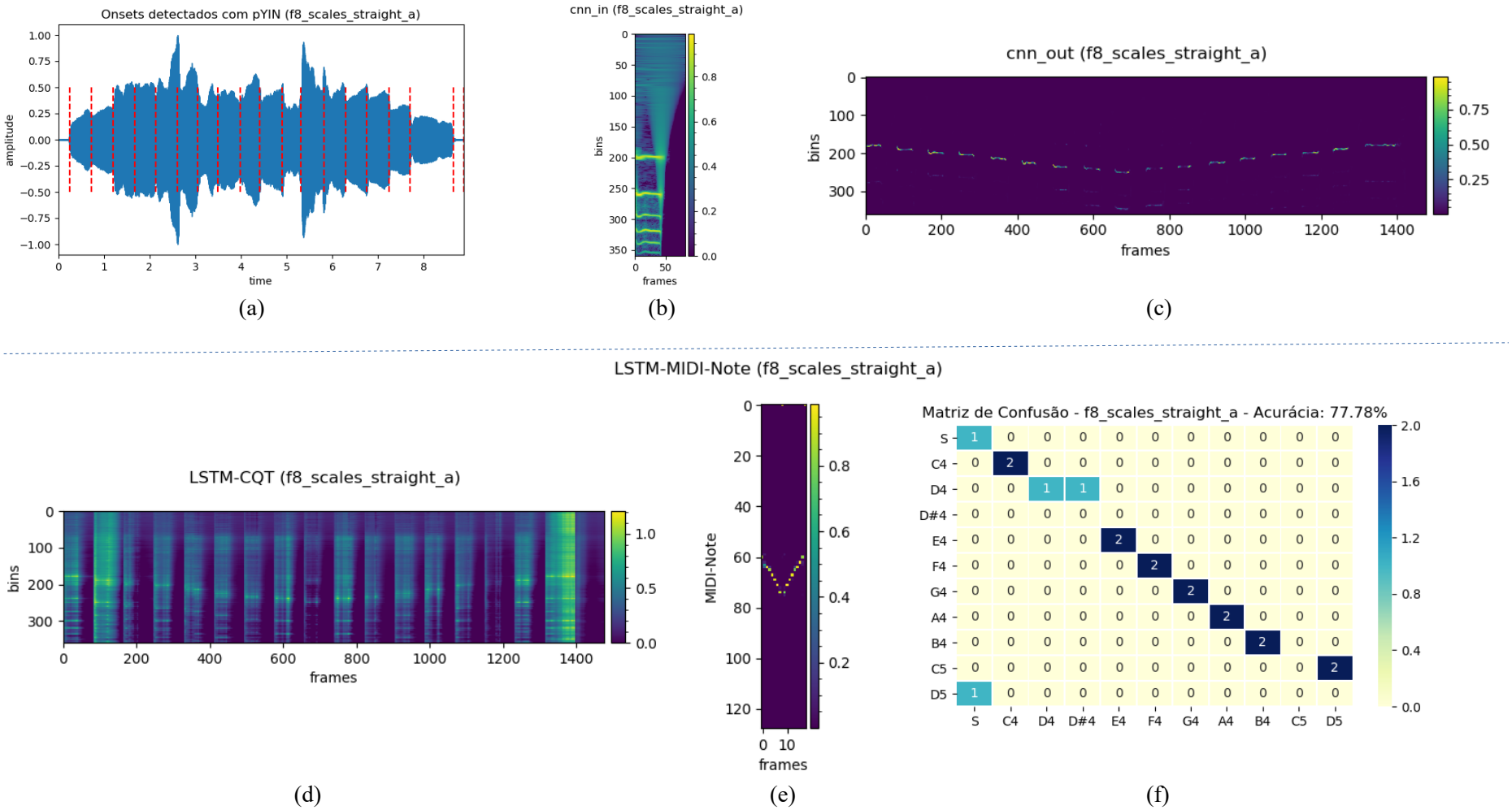
O Quadro 9 explica a lógica no uso dos dados enumerados com as letras *a* até *m*.

Quadro 9 – Explicação dos dados utilizados no Experimento

Item	**Significado**	**Componente**
(a)	Sinal de áudio com *Onsets* identificados	CNN-Melody
(b)	Sinal de áudio convertido em formato *Harmonic-CQT*	CNN-Melody
(c)	Melodia extraída do Solfejo	LSTM-CQT
(d)	Timbre de Piano representando em formato CQT	LSTM-CQT
(e)	Notas Musicais identificadas com *note-by-note*	LSTM-MIDI-Notes
(f)	Matriz de Confusão para a classificação *note-by-note*	LSTM-MIDI-Notes
(g)	Áudio sintetizado com timbre de Piano	LSTM-Wave
(h)	Áudio gerado em formato CQT	LSTM-Wave
(i)	Áudio Esperado em formato CQT	LSTM-Wave
(j)	Análise DTW entre o Áudio Sintetizado e Áudio Esperado	LSTM-Wave
(k)	Função Custo da distância entre o Áudio Sintetizado e Áudio Esperado	LSTM-Wave
(l)	Notas Musicais identificadas com *frame-by-frame*	LSTM-MIDI-Frames
(m)	Partitura de Referência para o exercício de Solfejo	LSTM-MIDI-Notes

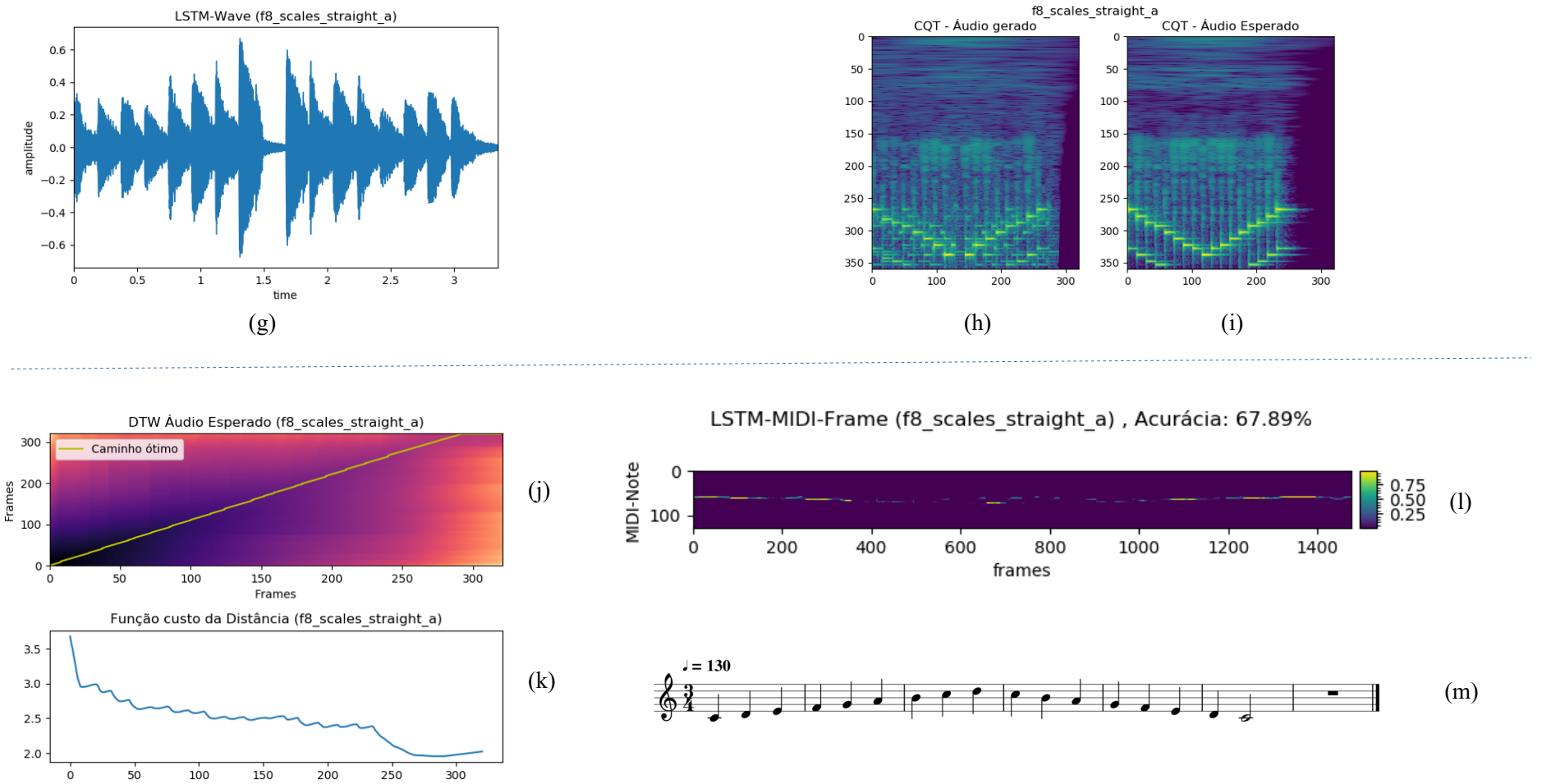
Fonte: Elaborado pelo autor.

Figura 55 – Resultados com Solfejos cantados em Escala Musical - Vogal *a* (1 de 2)



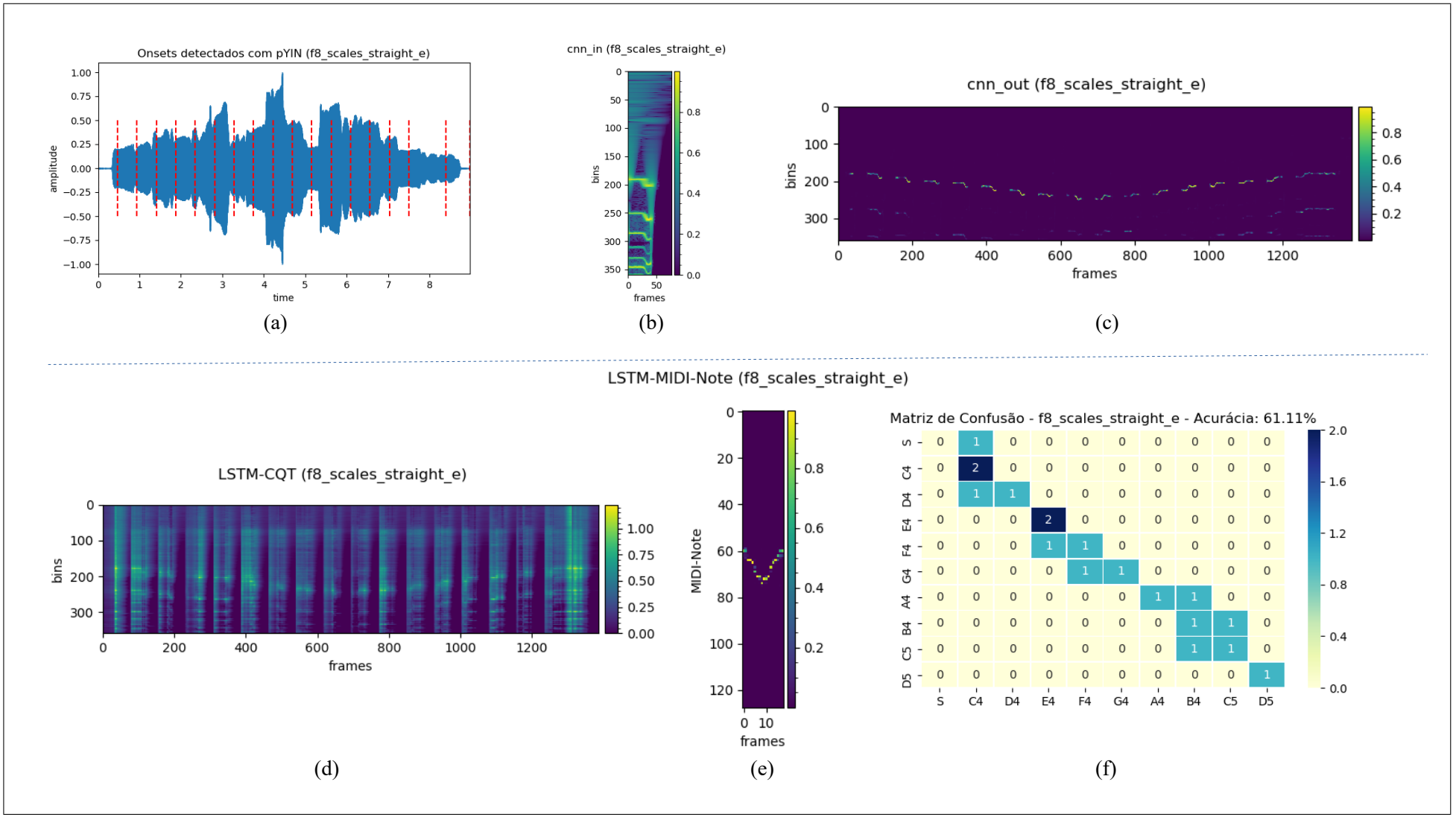
Fonte: Elaborado pelo autor.

Figura 56 – Resultados com Solfejos cantados em Escala Musical - Vogal *a* (2 de 2)



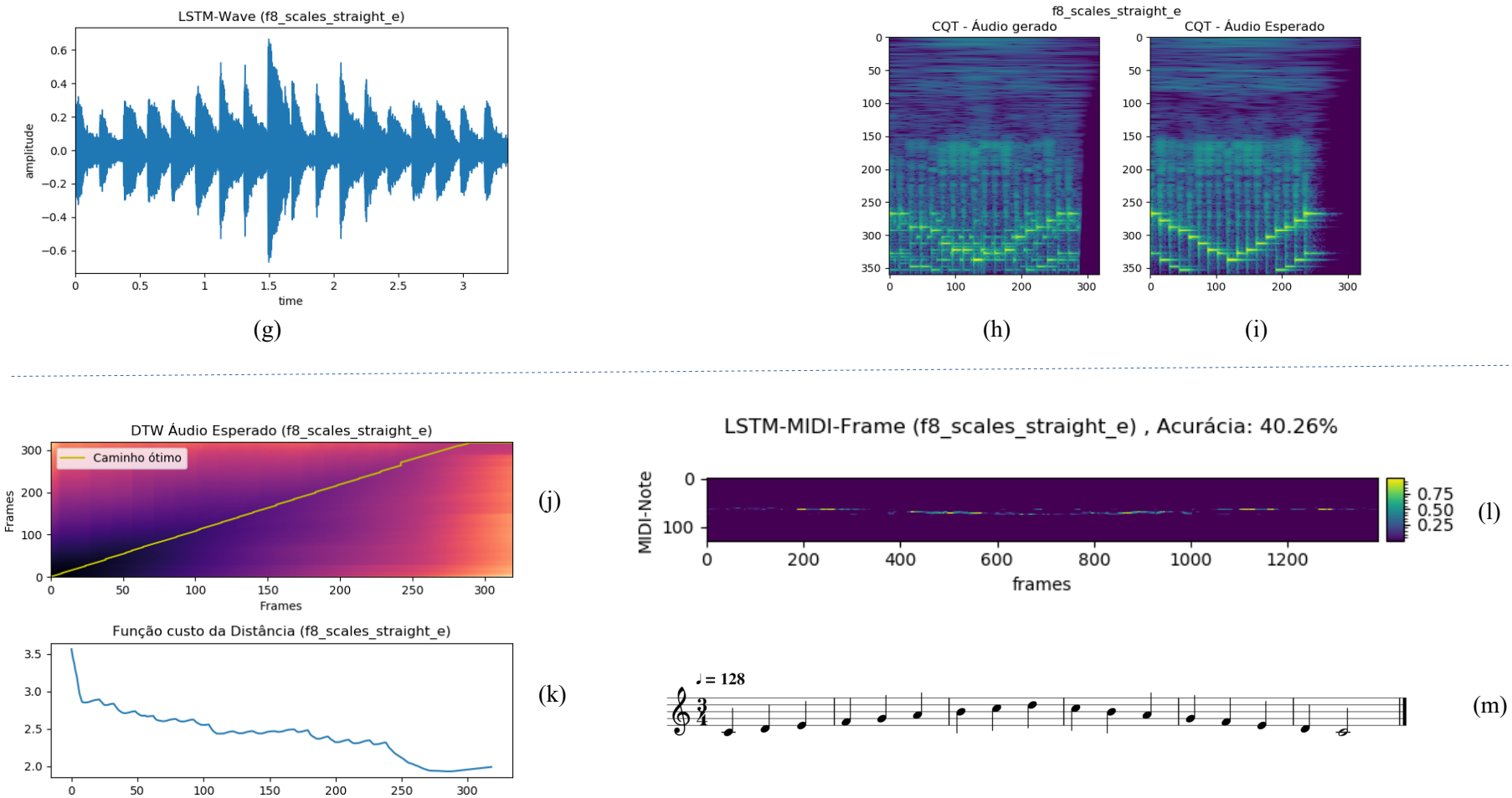
Fonte: Elaborado pelo autor.

Figura 57 – Resultados com Solfejos cantados em Escala Musical - Vogal *e* (1 de 2)



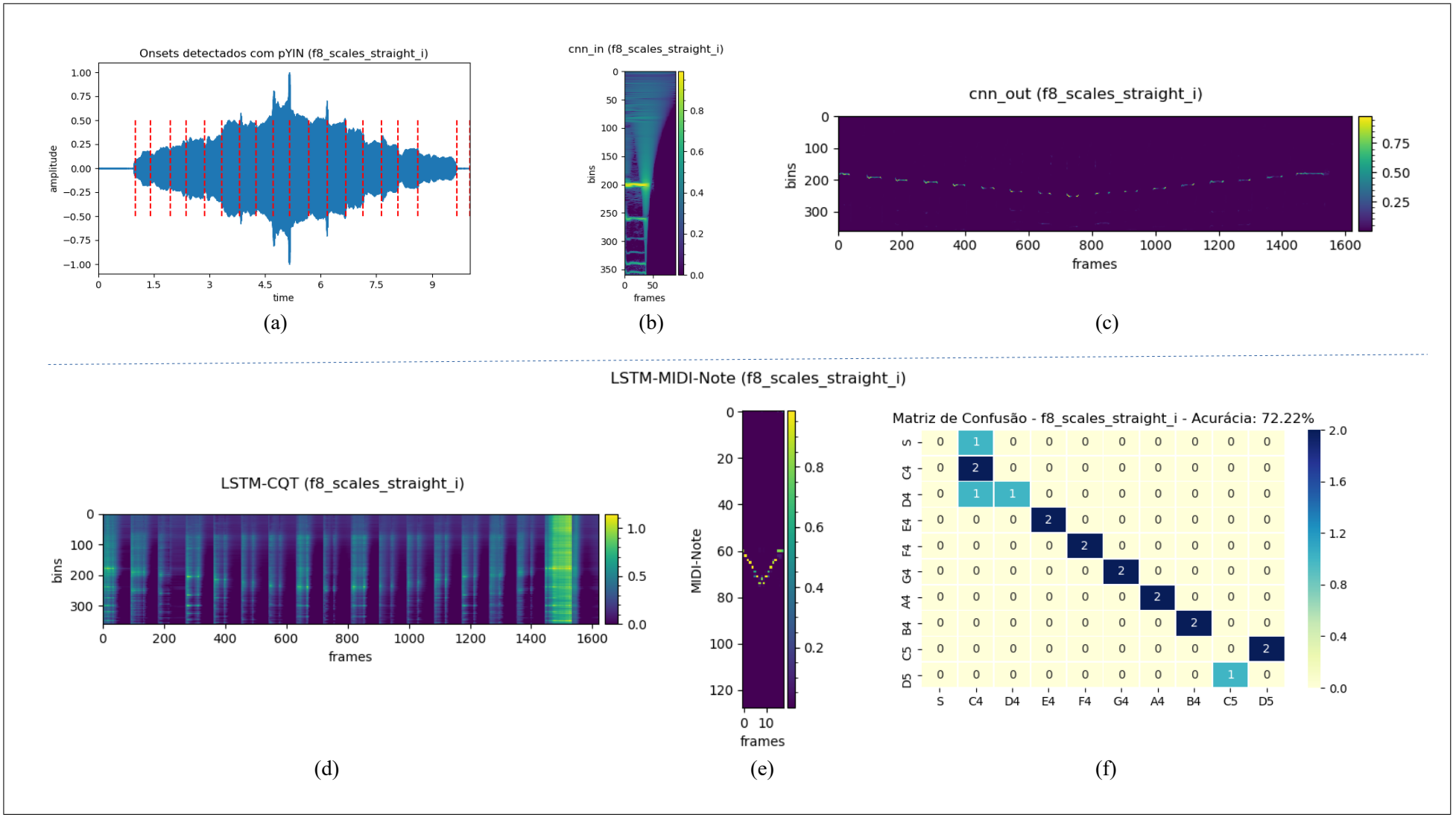
Fonte: Elaborado pelo autor.

Figura 58 – Resultados com Solfejos cantados em Escala Musical - Vogal *e* (2 de 2)



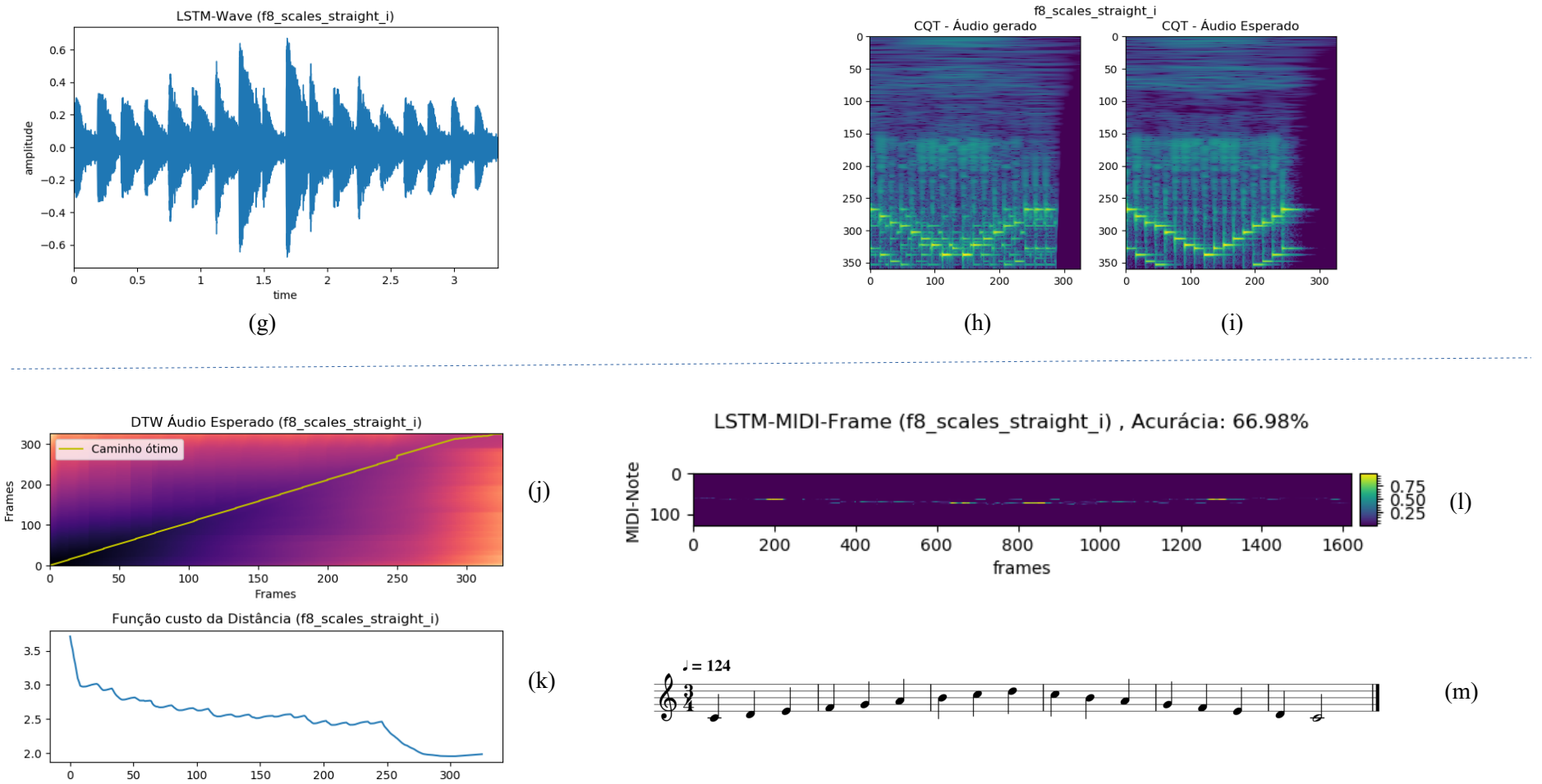
Fonte: Elaborado pelo autor.

Figura 59 – Resultados com Solfejos cantados em Escala Musical - Vogal *i* (1 de 2)



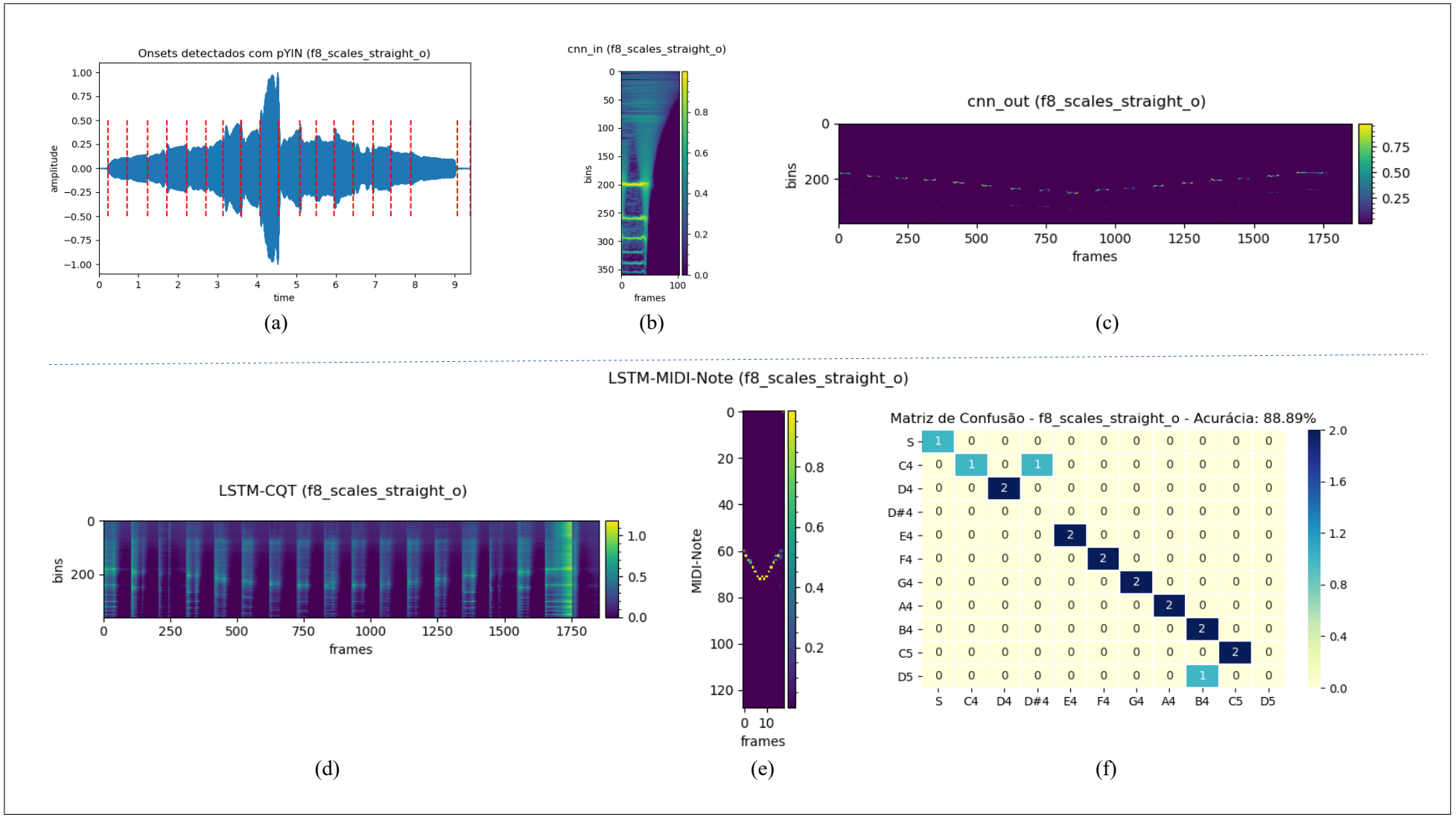
Fonte: Elaborado pelo autor.

Figura 60 – Resultados com Solfejos cantados em Escala Musical - Vogal *i* (2 de 2)



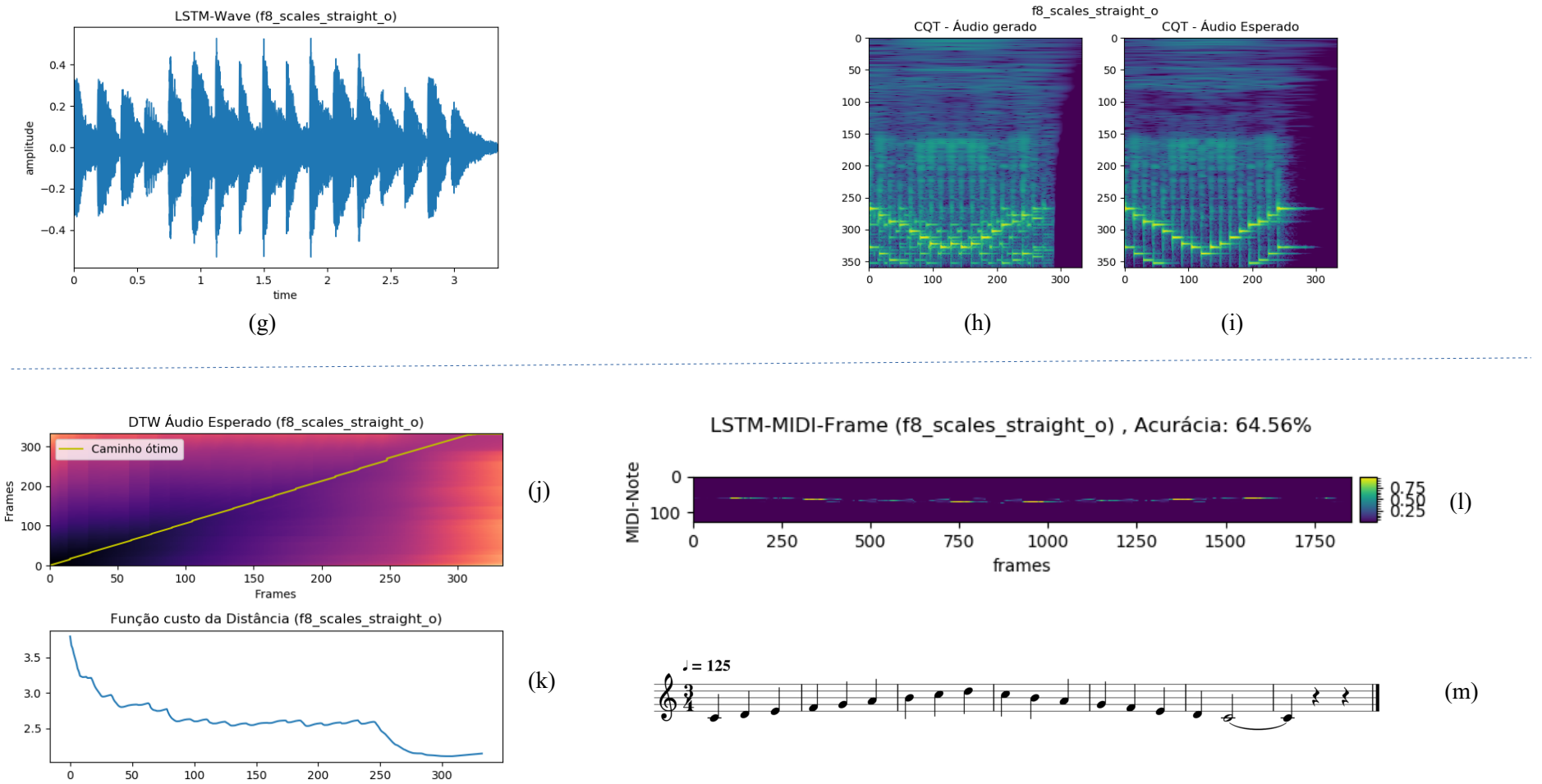
Fonte: Elaborado pelo autor.

Figura 61 – Resultados com Solfejos cantados em Escala Musical - Vogal *o* (1 de 2)



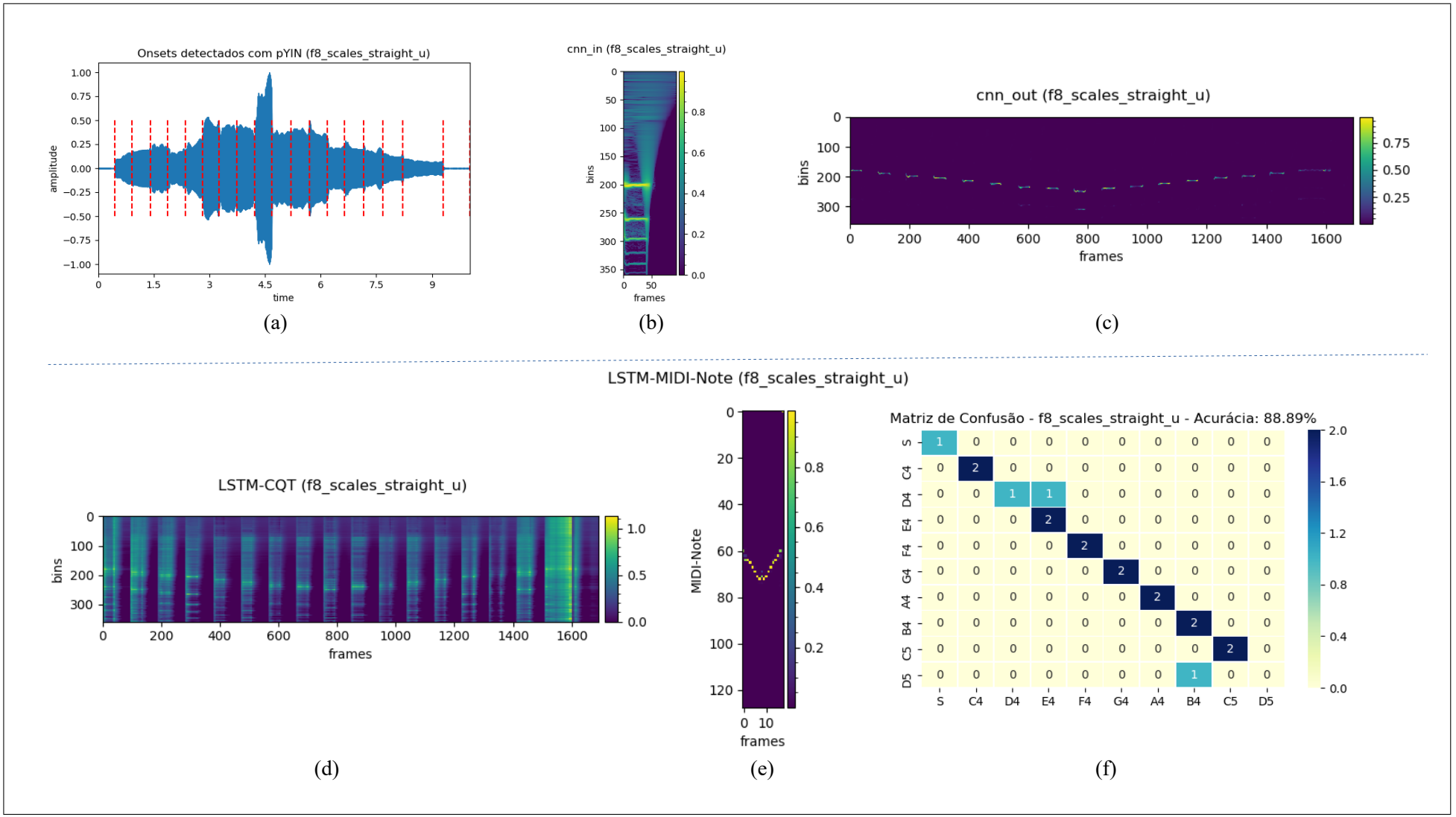
Fonte: Elaborado pelo autor.

Figura 62 – Resultados com Solfejos cantados em Escala Musical - Vogal *o* (2 de 2)



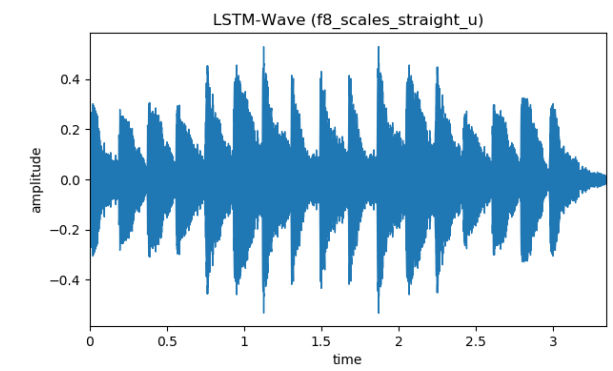
Fonte: Elaborado pelo autor.

Figura 63 – Resultados com Solfejos cantados em Escala Musical - Vogal *u* (1 de 2)

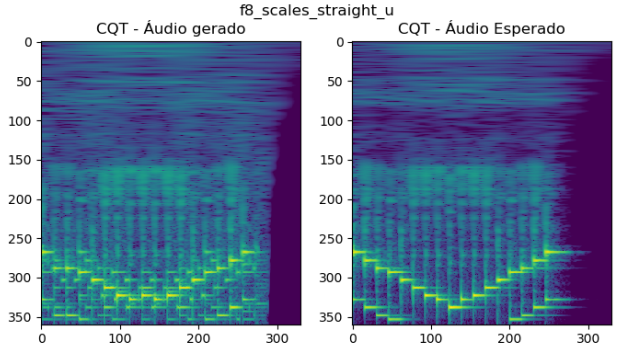


Fonte: Elaborado pelo autor.

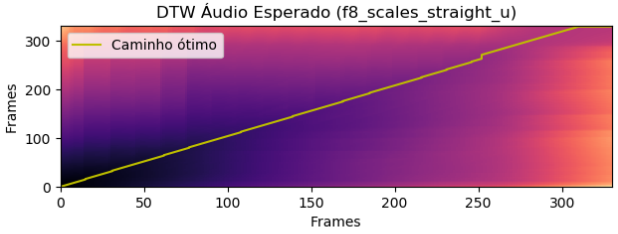
Figura 64 – Resultados com Solfejos cantados em Escala Musical - Vogal *u* (2 de 2)



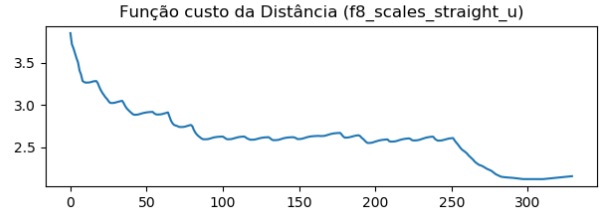
(g)



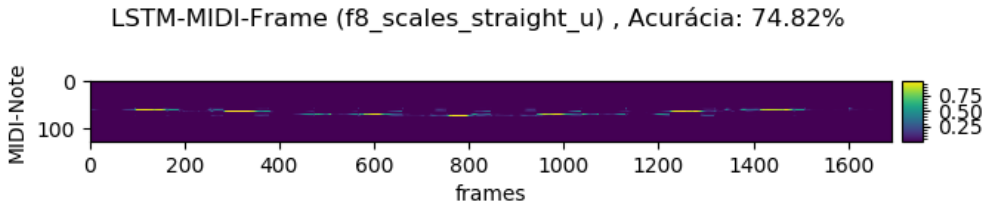
(h) (i)



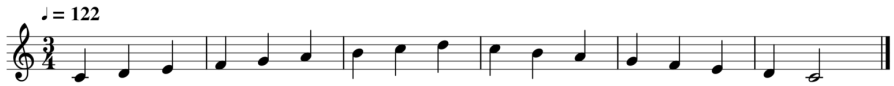
(j)



(k)



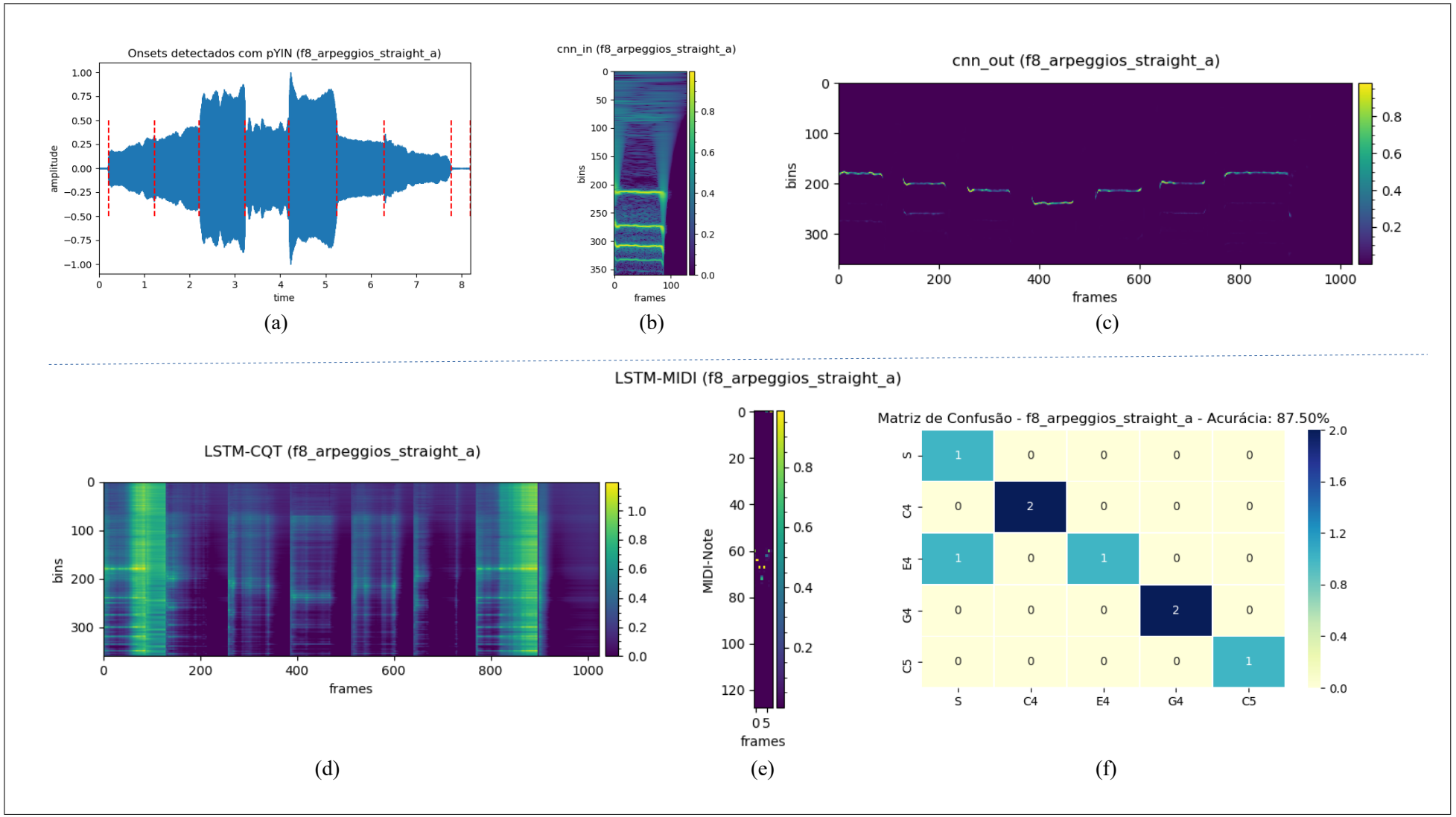
(l)



(m)

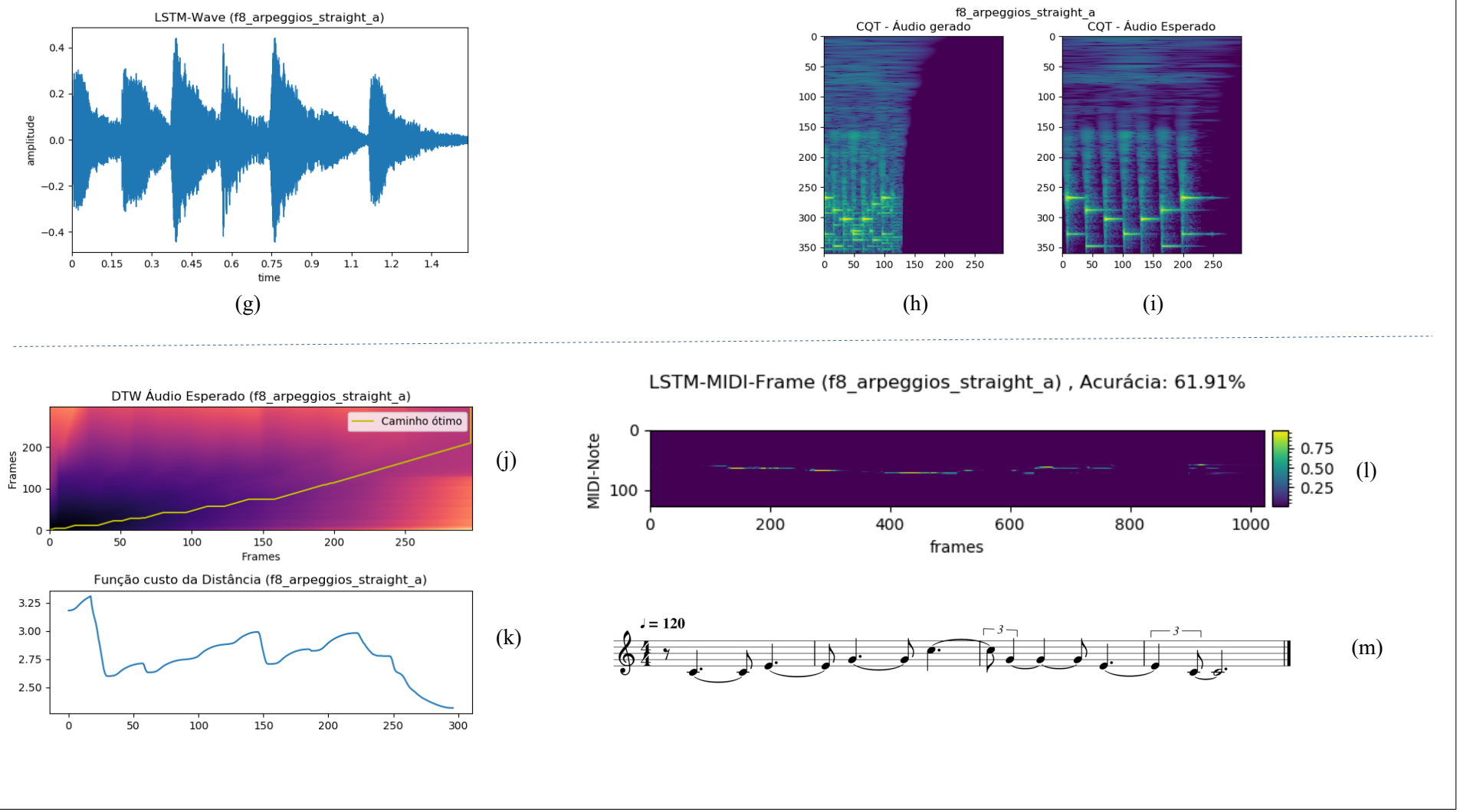
Fonte: Elaborado pelo autor.

Figura 65 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *a* (1 de 2)



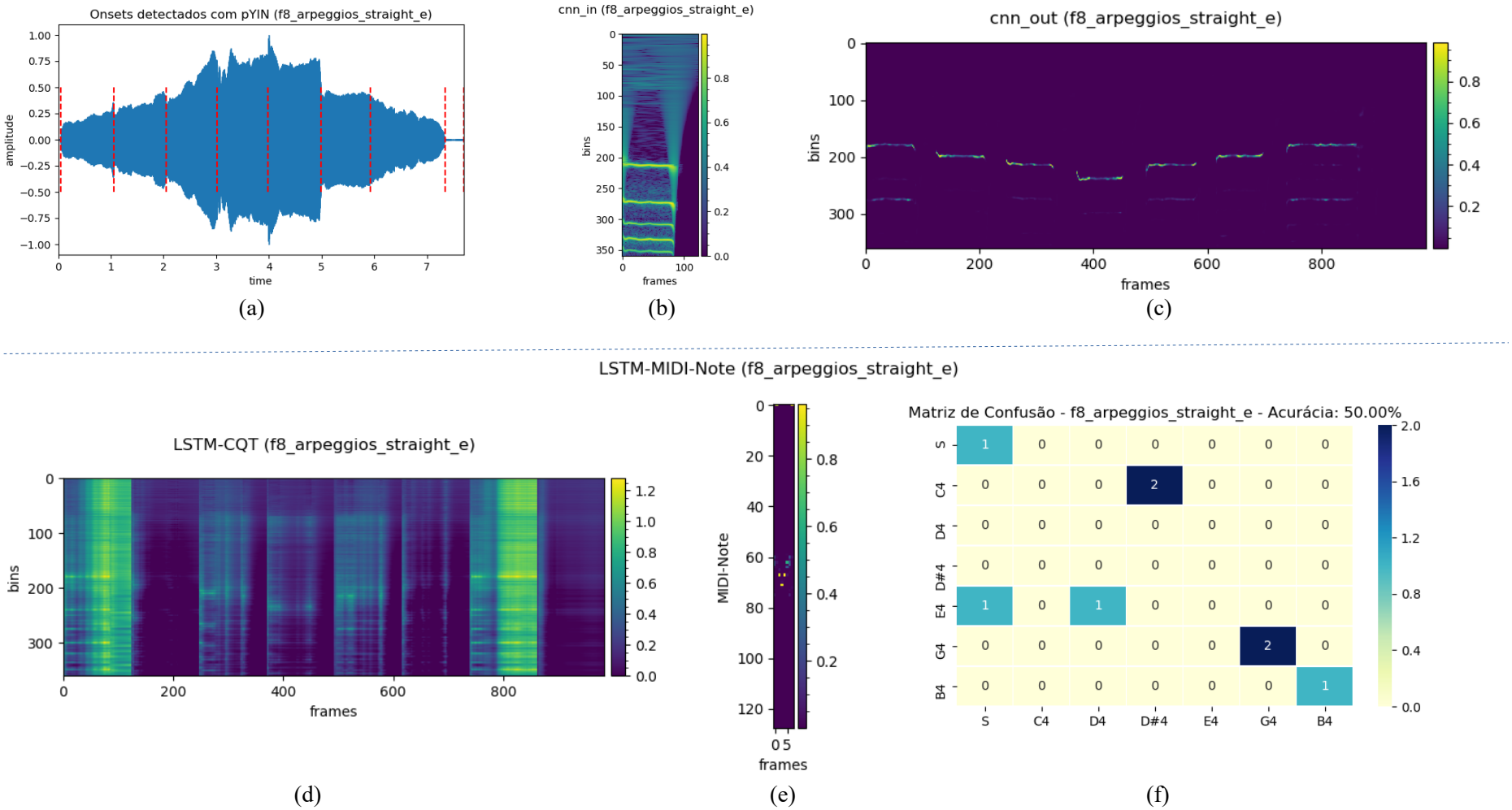
Fonte: Elaborado pelo autor.

Figura 66 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *a* (2 de 2)



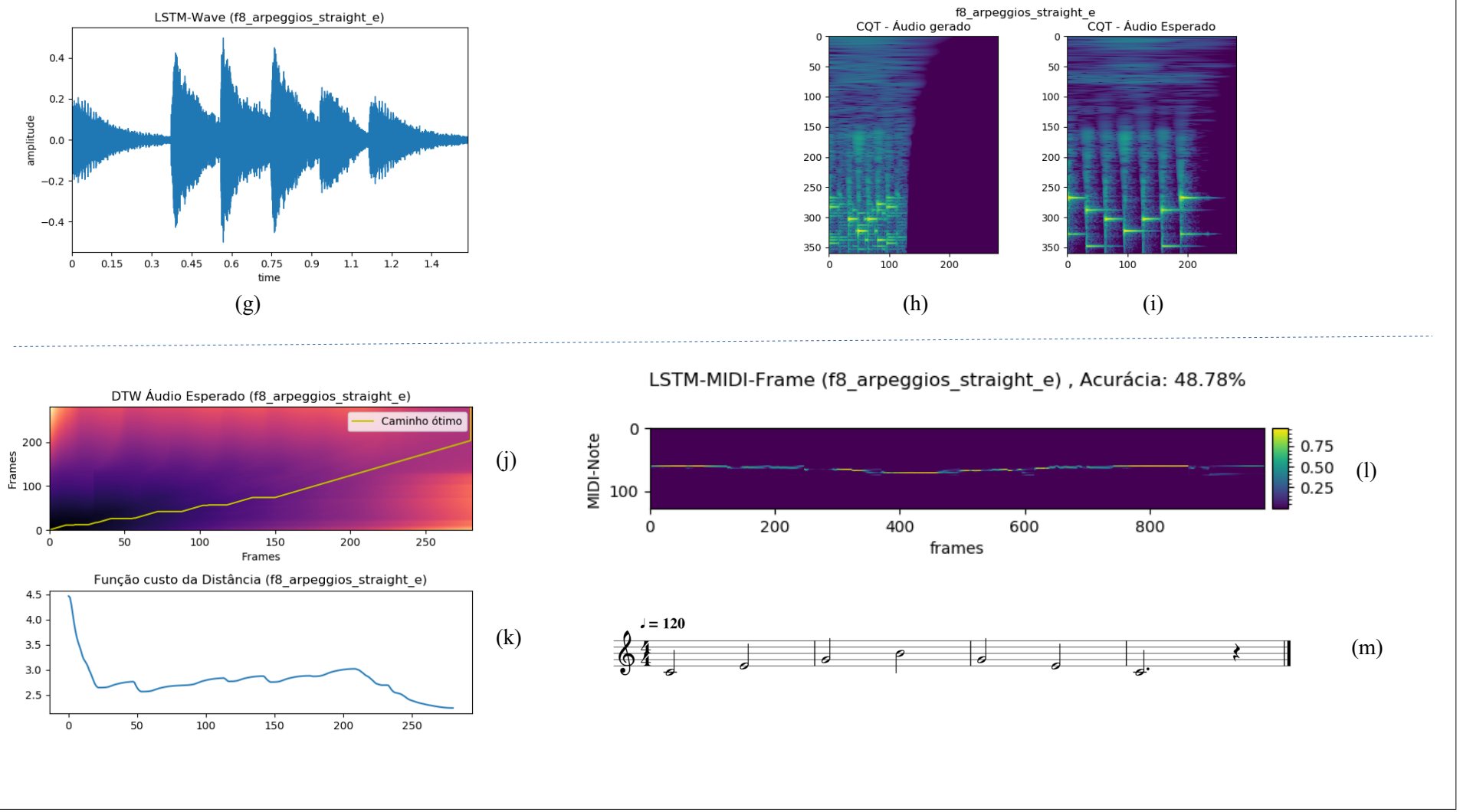
Fonte: Elaborado pelo autor.

Figura 67 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *e* (1 de 2)



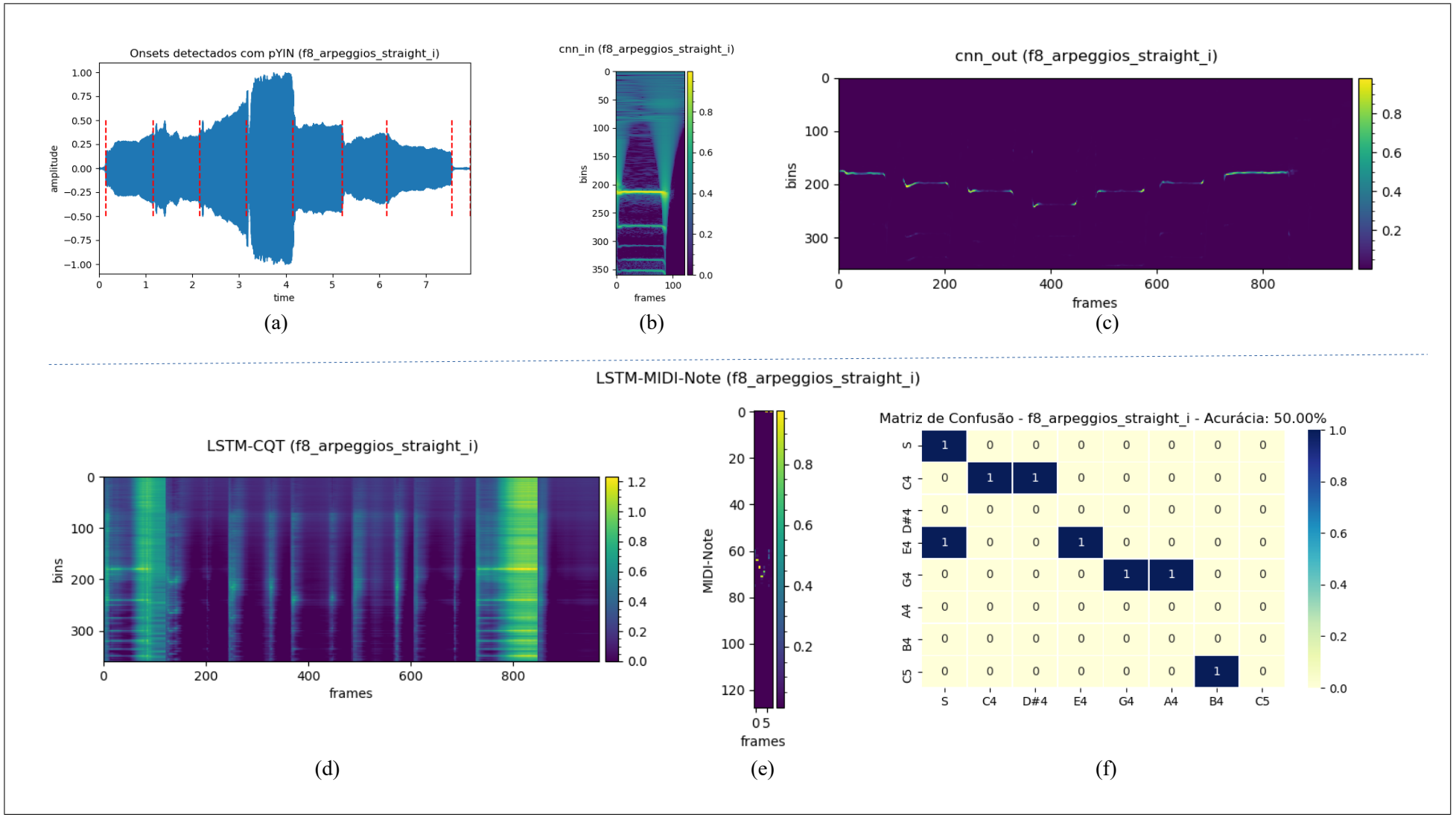
Fonte: Elaborado pelo autor.

Figura 68 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *e* (2 de 2)



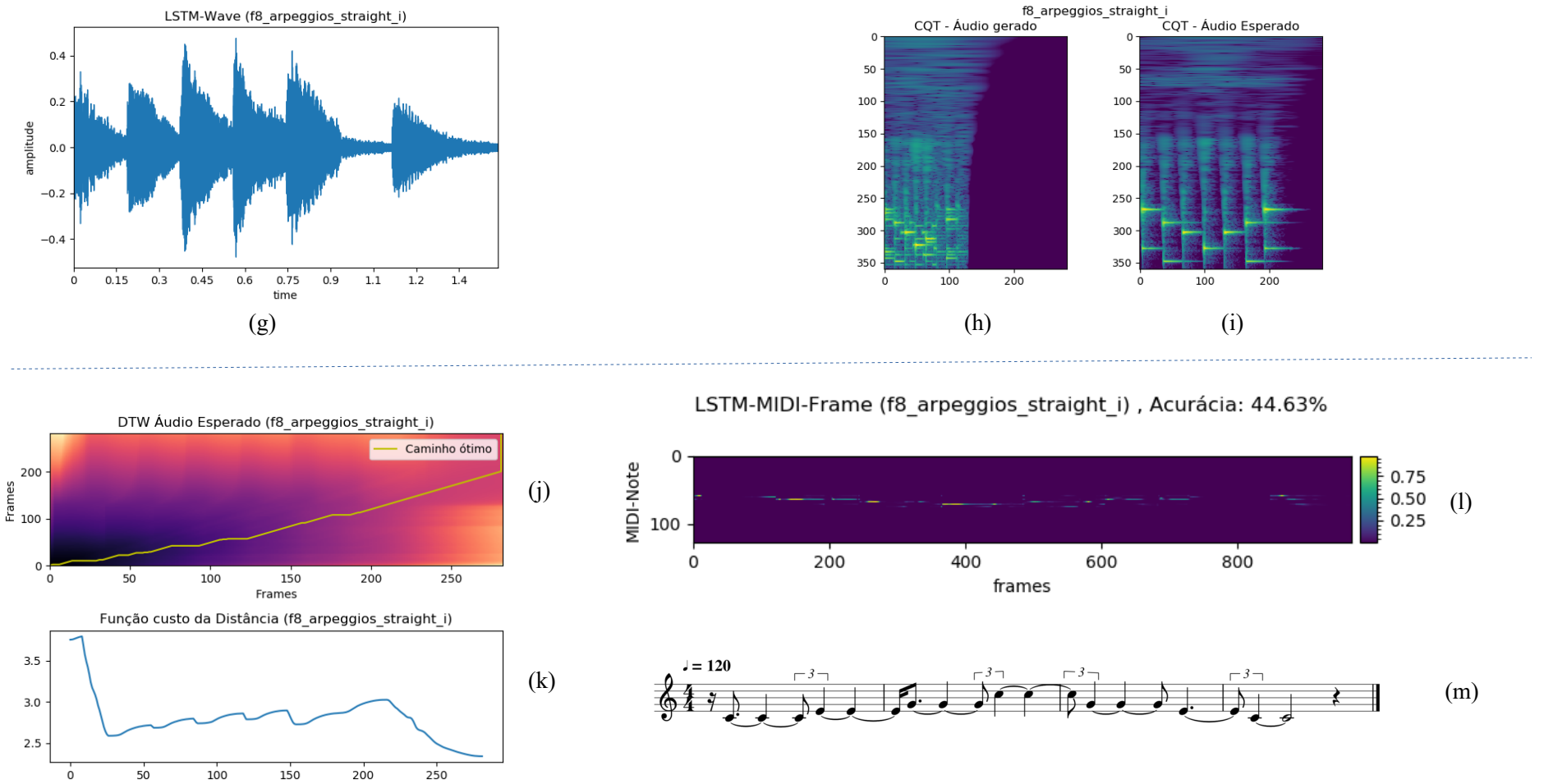
Fonte: Elaborado pelo autor.

Figura 69 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *i* (1 de 2)



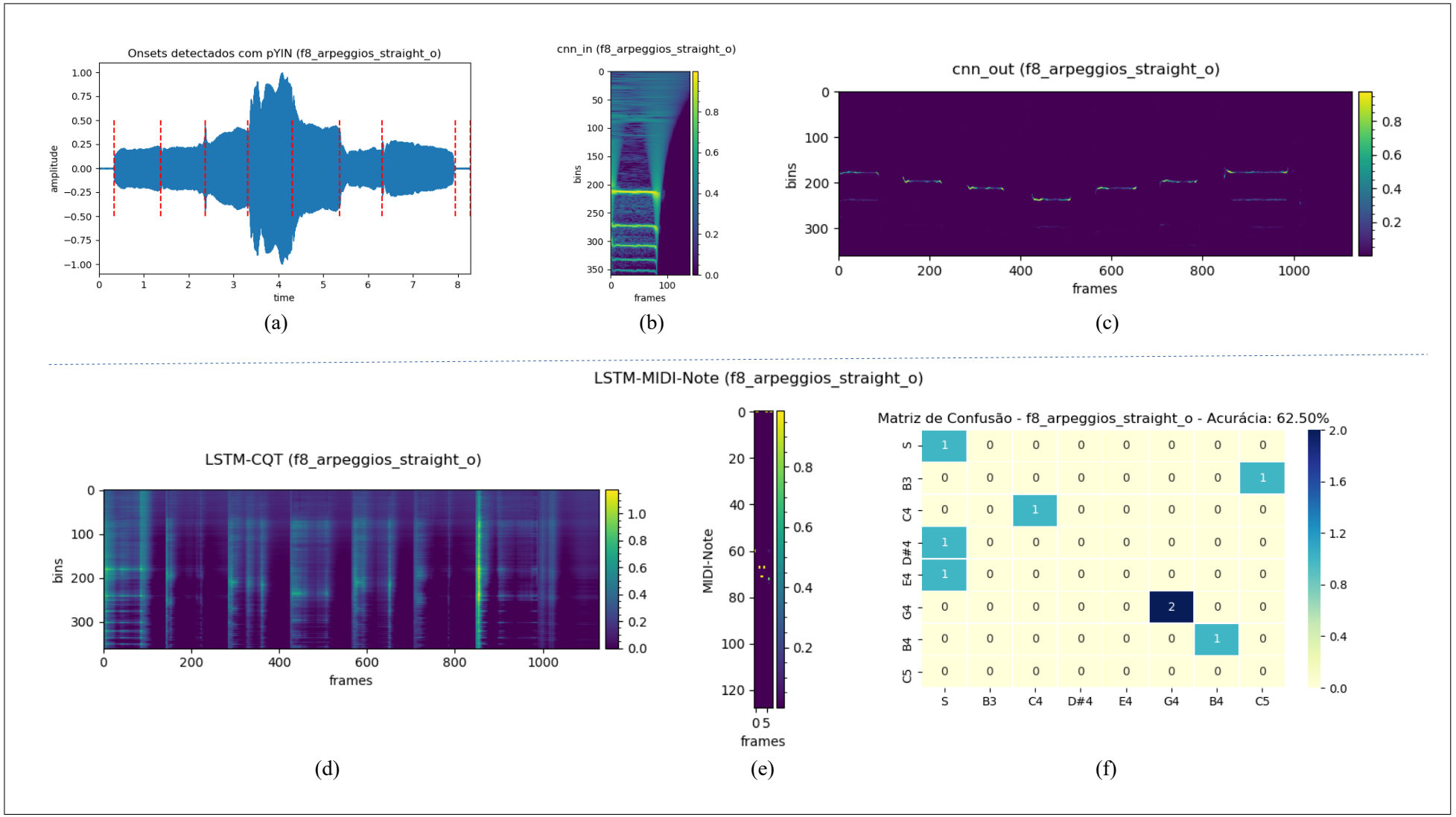
Fonte: Elaborado pelo autor.

Figura 70 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *i* (2 de 2)



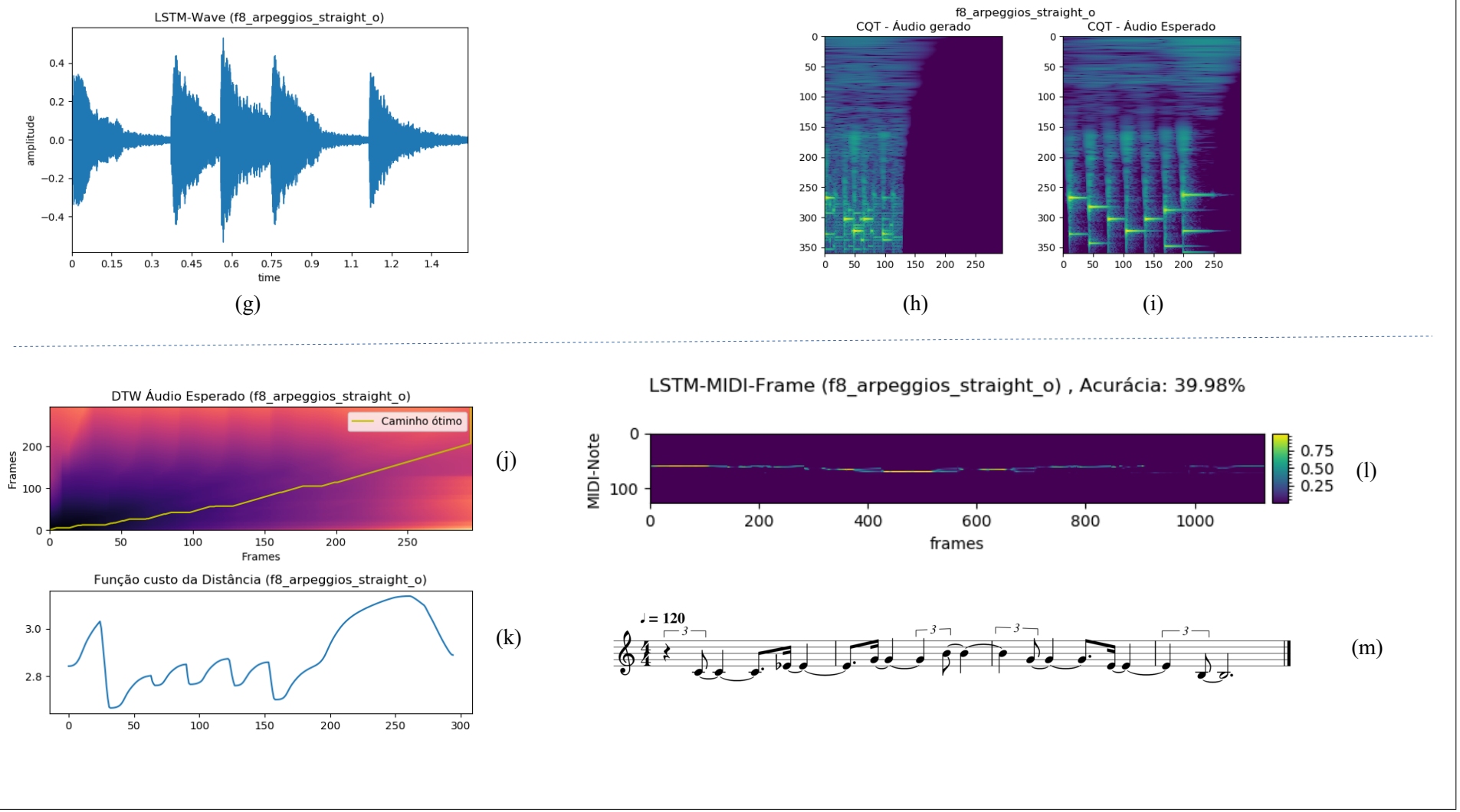
Fonte: Elaborado pelo autor.

Figura 71 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *o* (1 de 2)



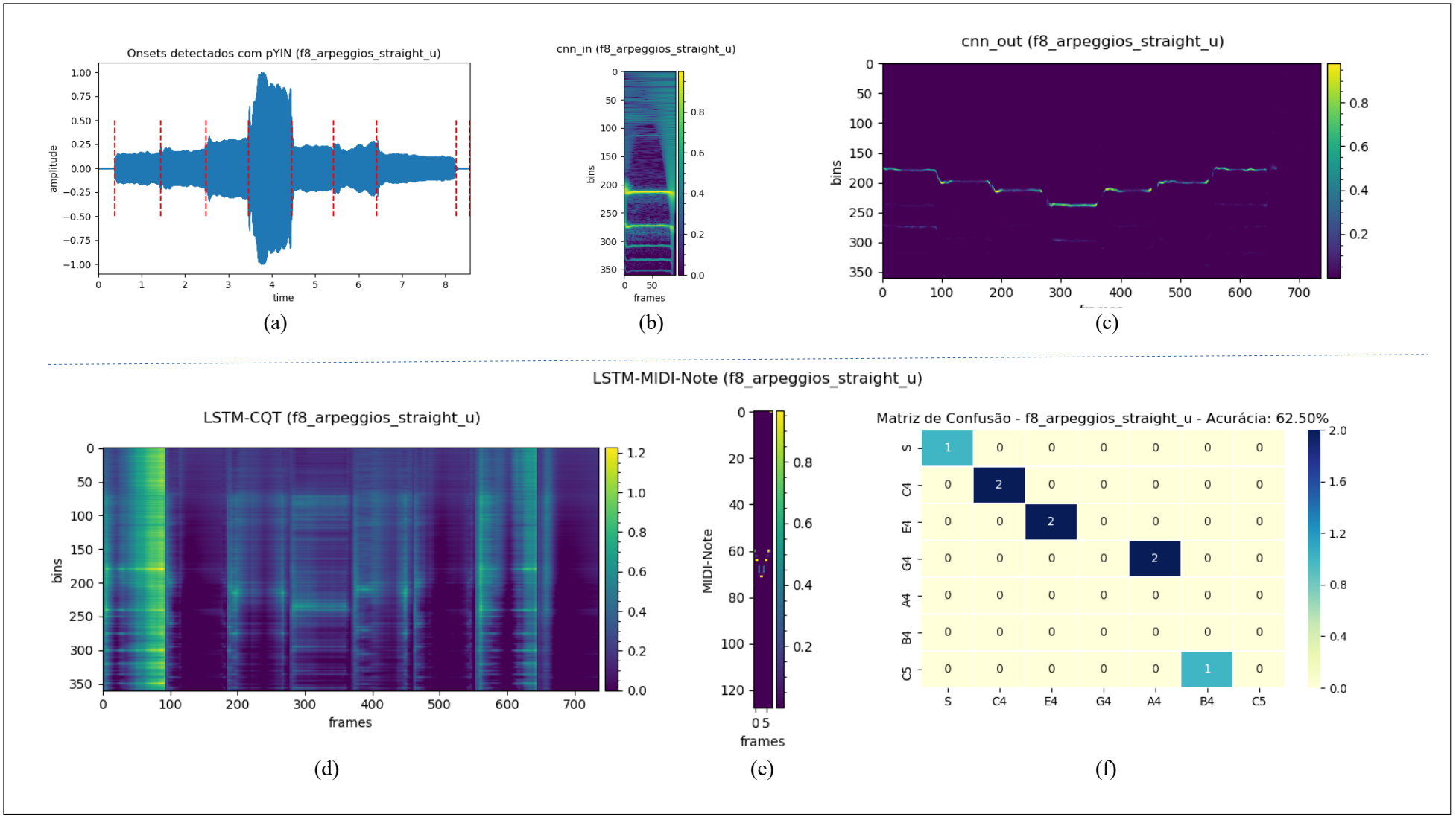
Fonte: Elaborado pelo autor.

Figura 72 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *o* (2 de 2)



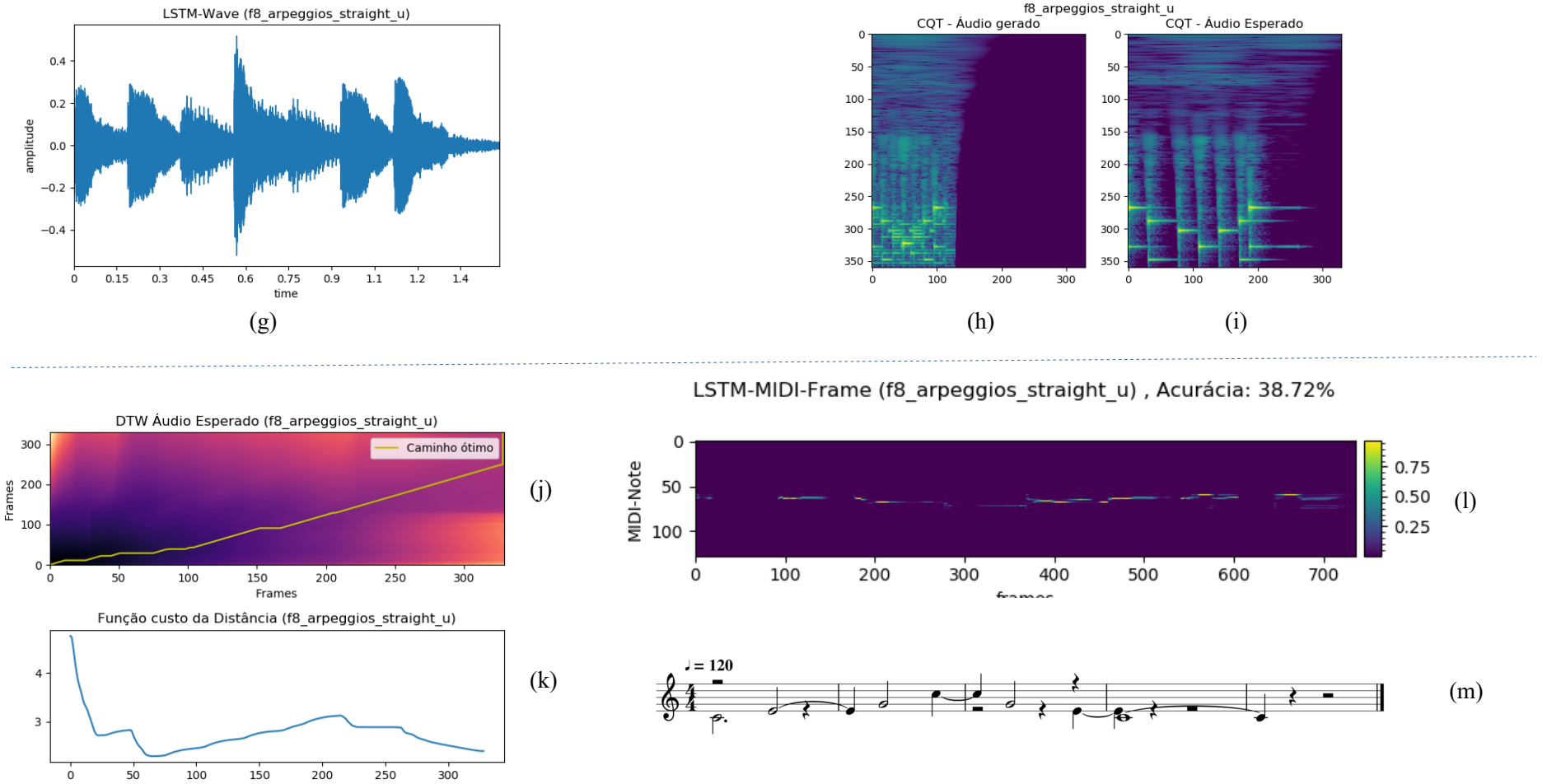
Fonte: Elaborado pelo autor.

Figura 73 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *u* (1 de 2)



Fonte: Elaborado pelo autor.

Figura 74 – Resultados com Solfejos cantados em *Arpeggios* - Vogal *u* (2 de 2)



Fonte: Elaborado pelo autor.

ANEXO A – *FEATURES* DE ÁUDIO MAIS FREQUENTES

Seguindo a taxonomia proposta por Mitrović, Zeppelzauer e Breiteneder (2010), a Figura 76, Figura 77 e Figura 78 contém as *features* mais frequentes agrupadas de forma hierárquica de acordo com características similares. Para conservar sua semântica e evitar problemas de tradução do inglês para português, os termos originais foram preservados.

As abreviaturas, legendas e símbolos ilustrados na a Figura 76, Figura 77 e Figura 78 tem seus significados resumidos na Figura 75 (a) que mostra as Transformações mais frequentes para *features* de áudio, Figura 75 (b) que lista os Filtros mais frequentes para *features* de áudio e a Figura 75 (c) que mostra os símbolos para Agregações e Detectores de *features* de áudio.

Uma importante informação que consta na a Figura 76, Figura 77 e Figura 78 na coluna *Signature* diz respeito aos cálculos matemáticos ou algoritmos necessários para extração manual das *features* de áudio. Por exemplo, para extrair a *feature* denominada *MPEG-7 Spectral Centroid* aplica-se a Transformada Discreta de *Fourier* (indicada pela letra "F" maiúscula) seguida do cálculo de média (indicada pela letra grega μ) que pode ser do tipo aritmética simples, ponderada ou geométrica.

Figura 75 – Filtros, Transformações, Agregações e Detectores para *features* de áudio

transformations	
A	Autocorrelation
R	Cross-Correlation
B	Band-pass Filter Bank
F	Discrete Fourier Transform (DFT)
C	(Inverse) Discrete Cosine Transform (DCT/IDCT)
Q	Constant Q Transform (CQT)
M	Modulated Complex Lapped Transform (MCLT)
V	Adaptive Time Frequency Transform (ATFT)
W	Discrete Wavelet (Packet) Transform (DW(P)T)
E	Phase Space Embedding
I	Independent Component Analysis (ICA)
P	(Oriented) Principal Component Analysis ((O)PCA)
S	Singular Value Decomposition (SVD)

(a)

Filters	
b	Band-pass Filter (Bank)
c	Comb Filter (Bank)
o	Low-pass Filter
f	Framing / Windowing
w	(Non-) Linear Weighting Function
d	Derivation, Difference
e	Energy Spectral Density
g	Group Delay Function
l	Logarithm
x	Exponential Function
n	Normalization
a	Autoregression (Linear Prediction Analysis)
r	Cepstral Recursion Formula

(b)

Aggregations and Detectors	
χ	Maximum
ι	Minimum
μ	Mean (weighted, arithmetic, geometric)
ϕ	Median
Σ	Sum, Weighted Sum
σ	Deviation, Sum of Differences
ϖ	Root Mean Square
ω	Power (Mean Square)
H	Entropy
π	Percentile
ρ	Regression
Λ	Histogram
β	Spectral binning
κ	Peak Detection
ψ	Harmonic Peak Detection
θ	Polynomial Root Finding
ζ	Zero-/Level Crossing Detector

(c)

Fonte: Adaptado de Mitrović, Zeppelzauer e Breiteneder (2010).

Figura 76 – Domínios: Temporal e Frequência

Table 5: This table gives an overview of temporal and frequency domain features. For each feature, we list the domain, a reference to the describing section, temporal scale, whether or not the feature is perceptual and employs psychoacoustic models, the complexity, dimension, application domain, and signature.

Domain	Section	Feature Name	Temporal Scale	Perceptual	Psychoac. Model	Complexity	Dimension	Appl. Domain	Signature
Temporal	5.2.1	Zero Crossing Rate (ZCR)	I	N	N	L	1	VAR	f ζ
		Linear Prediction ZCR	I	N	N	L	1	ASR	f ζ a ζ
		Zero Crossing Peak Amplitudes (ZCPA)	I	N	Y	M	V	ASR	f b ζ κ l Λ Σ
		Pitch Synchronous ZCPA	I	N	Y	M	V	ASR	f b Λ χ ζ κ l Λ Σ
	5.2.2	MPEG-7 Audio Waveform	I	N	N	L	2	-	f χ ι
		Amplitude Descriptor	I	N	N	L	9	ESR	f μ σ ζ μ σ
	5.2.3	Short-Time Energy, MPEG-7 Audio Power	I	N	N	L	1	VAR	f ω
		Volume	I	N	N	L	1	VAR	f ϖ
		MPEG-7 Temporal Centroid	X	N	N	L	1	MIR	[f ϖ] μ
		MPEG-7 Log Attack Time	G	N	N	L	1	MIR	[f ϖ] κ l
Frequency - Physical	5.3.1	Linear Predictive Coding	I	N	N	L	V	ASR	f (b)a (F)
		Line Spectral Frequencies	I	N	N	M	V	VAR	f a θ
	5.3.2	Daubechies Wavelet Coef. Histogr.	I	N	N	M	28	MIR	f W Λ
		Adaptive Time-Frequency Transform	G	N	N	M	42	MIR	V Λ
	5.3.3	Subband Energy Ratio	I	N	N	L	V	VAR	f F β e n
		Spectral Flux	I	N	N	L	1	VAR	[f F] d Σ
		Spectral Slope	I	N	N	L	4	VAR	f F ρ
		Spectral Peaks	X	N	N	L	V	MIR	[f F χ] d
		(Modified) Group Delay	I	N	N	M	V	ASR	f F (o)g (C)
Freq. - Perc.	5.4.1	MPEG-7 Spectral Centroid	G	Y	N	L	1	MIR	F μ
		MPEG-7 Audio Spectrum Centroid	I	Y	Y	M	1	VAR	f F β l μ
		Spectral Centroid	I	Y	N	L	1	VAR	f F (β)(l) μ
		Sharpness	I	Y	Y	M	1	VAR	f F β w w μ
		Spectral Center	I	Y	N	L	1	MIR	f F e ϕ

Fonte: Adaptado de Mitrović, Zeppelzauer e Breiteneder (2010).

Figura 77 – *Features* Perceptivas do Domínio da Frequência

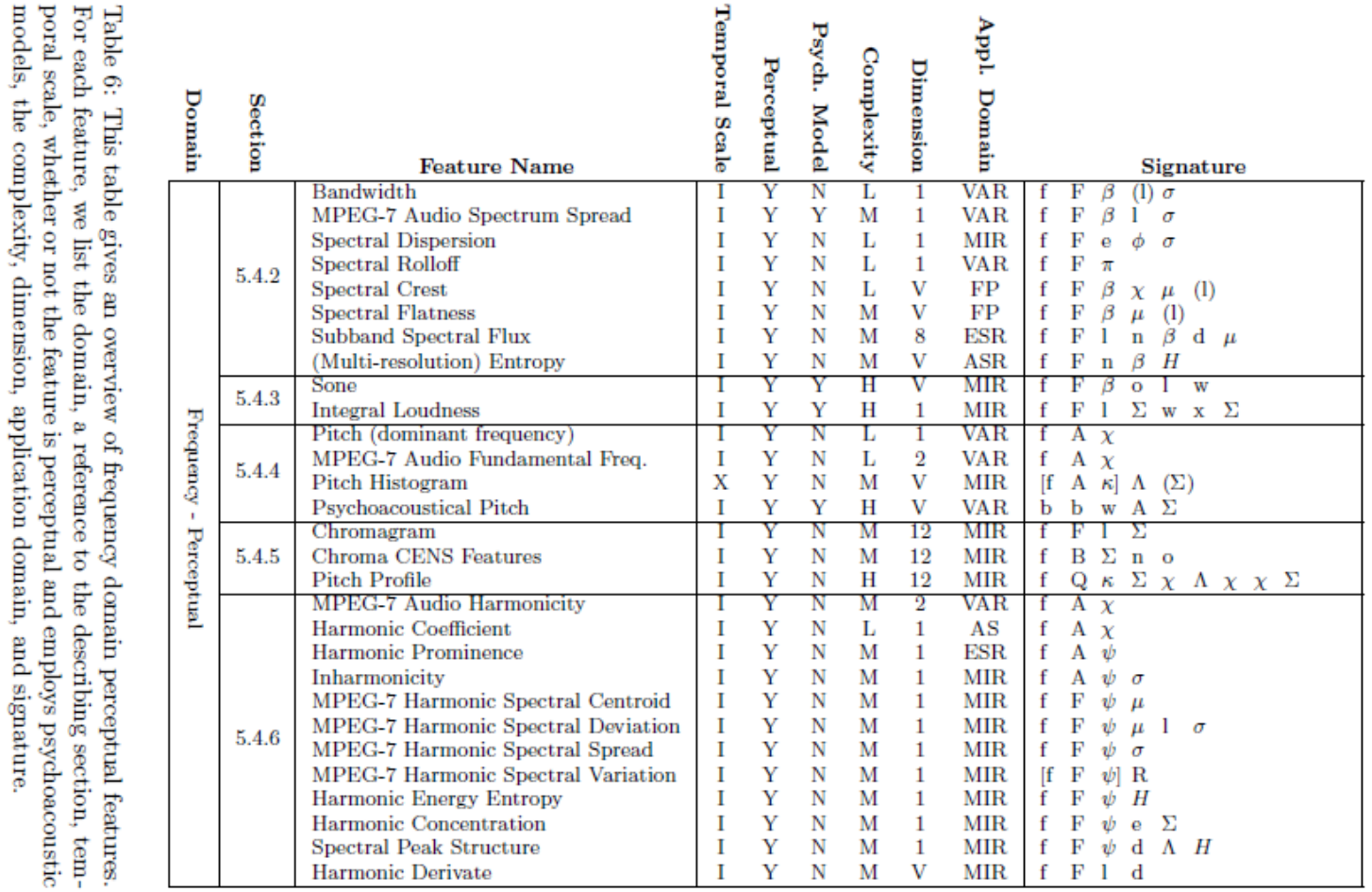


Table 6: This table gives an overview of frequency domain perceptual features. For each feature, we list the domain, a reference to the describing section, temporal scale, whether or not the feature is perceptual and employs psychoacoustic models, the complexity, dimension, application domain, and signature.

Domain	Section	Feature Name	Temporal Scale	Perceptual	Psych. Model	Complexity	Dimension	Appl. Domain	Signature
Frequency - Perceptual	5.4.2	Bandwidth	I	Y	N	L	1	VAR	f F β (l) σ
		MPEG-7 Audio Spectrum Spread	I	Y	Y	M	1	VAR	f F β l σ
		Spectral Dispersion	I	Y	N	L	1	MIR	f F e ϕ σ
		Spectral Rolloff	I	Y	N	L	1	VAR	f F π
		Spectral Crest	I	Y	N	L	V	FP	f F β χ μ (l)
		Spectral Flatness	I	Y	N	M	V	FP	f F β μ (l)
		Subband Spectral Flux	I	Y	N	M	8	ESR	f F l n β d μ
		(Multi-resolution) Entropy	I	Y	N	M	V	ASR	f F n β H
	5.4.3	Sone	I	Y	Y	H	V	MIR	f F β o I w
		Integral Loudness	I	Y	Y	H	1	MIR	f F l Σ w x Σ
	5.4.4	Pitch (dominant frequency)	I	Y	N	L	1	VAR	f A χ
		MPEG-7 Audio Fundamental Freq.	I	Y	N	L	2	VAR	f A χ
		Pitch Histogram	X	Y	N	M	V	MIR	[f A κ] Λ (Σ)
		Psychoacoustical Pitch	I	Y	Y	H	V	VAR	b b w A Σ
	5.4.5	Chromagram	I	Y	N	M	12	MIR	f F I Σ
		Chroma CENS Features	I	Y	N	M	12	MIR	f B Σ n o
		Pitch Profile	I	Y	N	H	12	MIR	f Q κ Σ χ Λ χ χ Σ
	5.4.6	MPEG-7 Audio Harmonicity	I	Y	N	M	2	VAR	f A χ
		Harmonic Coefficient	I	Y	N	L	1	AS	f A χ
		Harmonic Prominence	I	Y	N	M	1	ESR	f A ψ
		Inharmonicity	I	Y	N	M	1	MIR	f A ψ σ
		MPEG-7 Harmonic Spectral Centroid	I	Y	N	M	1	MIR	f F ψ μ
		MPEG-7 Harmonic Spectral Deviation	I	Y	N	M	1	MIR	f F ψ μ l σ
		MPEG-7 Harmonic Spectral Spread	I	Y	N	M	1	MIR	f F ψ σ
		MPEG-7 Harmonic Spectral Variation	I	Y	N	M	1	MIR	[f F ψ] R
		Harmonic Energy Entropy	I	Y	N	M	1	MIR	f F ψ H
		Harmonic Concentration	I	Y	N	M	1	MIR	f F ψ e Σ
		Spectral Peak Structure	I	Y	N	M	1	MIR	f F ψ d Λ H
		Harmonic Derivate	I	Y	N	M	V	MIR	f F l d

Fonte: Adaptado de Mitrović, Zeppelzauer e Breiteneder (2010).

Figura 78 – *Features* do Domínio *Cepstral*, Modulação de Frequência, Autodomínio e Espaço de Fase

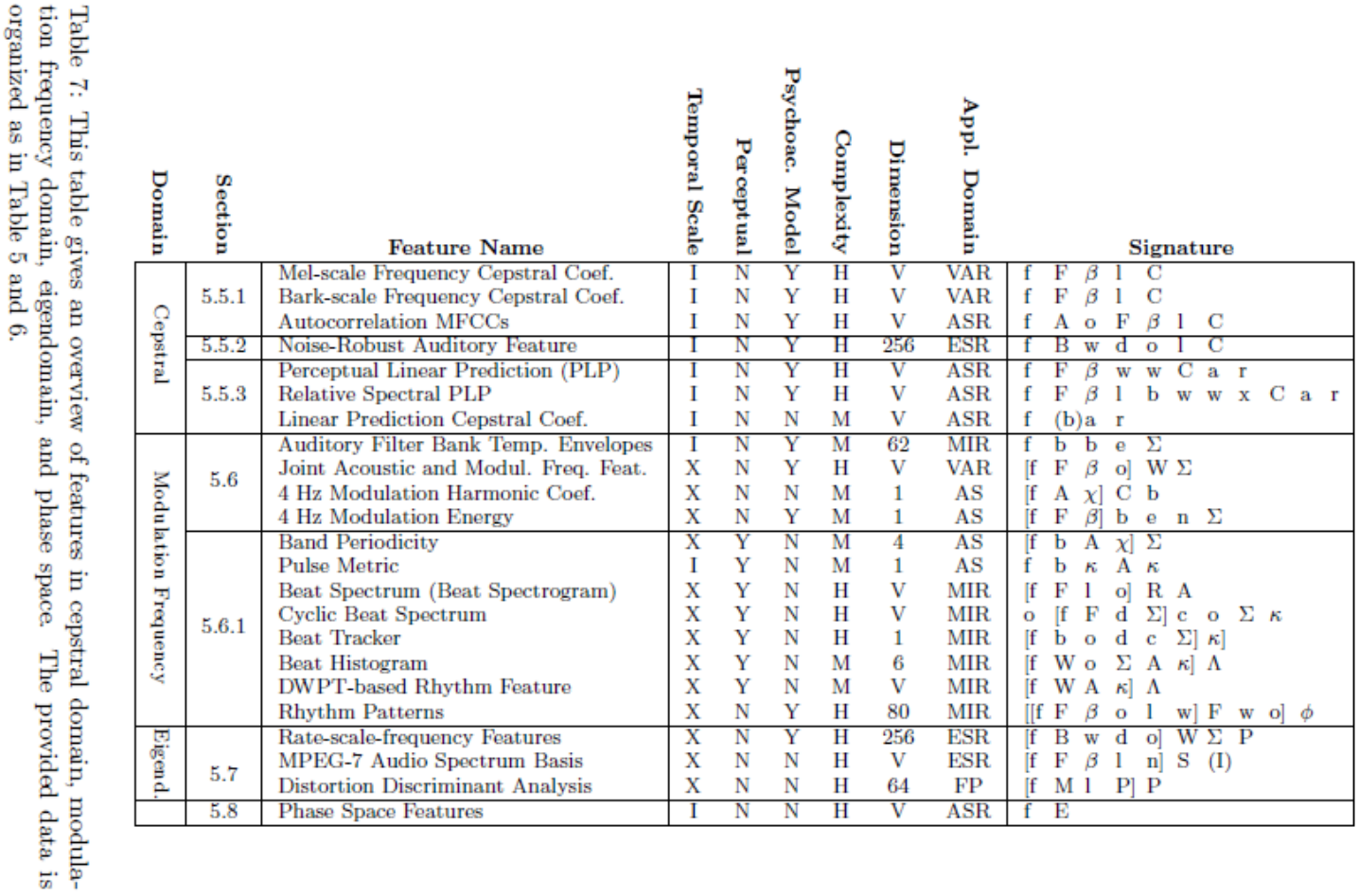


Table 7: This table gives an overview of features in cepstral domain, modulation frequency domain, eigendomain, and phase space. The provided data is organized as in Table 5 and 6.

Domain	Section	Feature Name	Temporal Scale	Perceptual	Psychoac. Model	Complexity	Dimension	Appl. Domain	Signature
Cepstral	5.5.1	Mel-scale Frequency Cepstral Coef.	I	N	Y	H	V	VAR	f F β l C
		Bark-scale Frequency Cepstral Coef.	I	N	Y	H	V	VAR	f F β l C
		Autocorrelation MFCCs	I	N	Y	H	V	ASR	f A o F β l C
	5.5.2	Noise-Robust Auditory Feature	I	N	Y	H	256	ESR	f B w d o l C
	5.5.3	Perceptual Linear Prediction (PLP)	I	N	Y	H	V	ASR	f F β w w C a r
		Relative Spectral PLP	I	N	Y	H	V	ASR	f F β l b w w x C a r
		Linear Prediction Cepstral Coef.	I	N	N	M	V	ASR	f (b)a r
Modulation Frequency	5.6	Auditory Filter Bank Temp. Envelopes	I	N	Y	M	62	MIR	f b b e Σ
		Joint Acoustic and Modul. Freq. Feat.	X	N	Y	H	V	VAR	[f F β o] W Σ
		4 Hz Modulation Harmonic Coef.	X	N	N	M	1	AS	[f A χ] C b
		4 Hz Modulation Energy	X	N	Y	M	1	AS	[f F β] b e n Σ
	5.6.1	Band Periodicity	X	Y	N	M	4	AS	[f b A χ] Σ
		Pulse Metric	I	Y	N	M	1	AS	f b κ A κ
		Beat Spectrum (Beat Spectrogram)	X	Y	N	H	V	MIR	[f F l o] R A
		Cyclic Beat Spectrum	X	Y	N	H	V	MIR	o [f F d Σ] c o Σ κ
		Beat Tracker	X	Y	N	H	1	MIR	[f b o d c Σ] κ]
		Beat Histogram	X	Y	N	M	6	MIR	[f W o Σ A κ] Λ
		DWPT-based Rhythm Feature	X	Y	N	M	V	MIR	[f W A κ] Λ
		Rhythm Patterns	X	N	Y	H	80	MIR	[[f F β o l w] F w o] ϕ
Eigend.	5.7	Rate-scale-frequency Features	X	N	Y	H	256	ESR	[f B w d o] W Σ P
		MPEG-7 Audio Spectrum Basis	X	N	N	H	V	ESR	[f F β l n] S (I)
		Distortion Discriminant Analysis	X	N	N	H	64	FP	[f M l P] P
	5.8	Phase Space Features	I	N	N	H	V	ASR	f E

Fonte: Adaptado de Mitrović, Zeppelzauer e Breiteneder (2010).

ANEXO B – ALGORITMOS DE EXTRAÇÃO DE MELODIA

Figura 79 – Estado da Arte para Extração de *Features* Musicais com técnicas de *Music Information Retrieval*

Algorithm [MIREX year]	Preprocessing	Spectral Transform and Processing	Multipitch Represent. (salience function)	Tracking	Voicing	Approach Type
Paiva et al. (2006) [2005]	-	Auditory model + autocorrelation peaks	Summary correlogram	Multipitch trajectories + note deletion	Salience valleys	Salience based
Marolt (2004) [2005]	-	STFT + SMS harmonics plus noise	EM fit to tone models	Fragments + fragment clustering	Loudness filter	
Goto (2004) [2005]	Bandpass filter	Multirate filterbank + IF-based peak selection	EM fit to tone models	Tracking agents	-	
Cancela (2008) [2008]	-	Constant-Q + high pass filter + log power normalisation	Harmonicity map	Contour tracking + weighting + smoothing	Adaptive threshold	
Ryynänen & Klapuri (2008a) [2008]	-	STFT + spectral whitening	Harmonic summation	Note event HMM + global HMM	Silence model	
Dressler (2011a) [2009]	-	MRFFT + IF peak correction + magnitude threshold	Pairwise comparison of spectral peaks	Streaming rules	Dynamic threshold	
Rao & Rao (2010) [2009]	-	High resolution FFT + main-lobe magnitude matching	SMS + TWM	Dynamic programming	NHC threshold	
Salamon & Gómez (2012) [2011]	Equal loudness filter	STFT + IF peak correction	Harmonic summation	Contour tracking + contour filtering	Salience distribution	
Jo et al. (2010) [2011]	-	STFT with varying window length	Harmonic summation	Stable candidates + rule based selection	Implicit	
Arora & Behera (2013) [2012]	-	STFT + log spectrum + peak selection	IFT of log spectrum	Harmonic cluster tracking + cluster score	Harm. sum. threshold	
Hsu & Jang (2010a) [2010]	Harm./perc. sound sep.	MRFFT + vocal partial discrimination	Normalised subharmonic summation	Global trend + dynamic programming	Classification	Salience based + source sep. preprocessing
Yeh et al. (2012) [2012]	Harm./perc. sound sep.	MRFFT + vocal partial discrimination	Normalised subharmonic summation	Trend estimation + HMM	-	
Durrieu et al. (2010) [2009]	Source/filter model for melody source separation			Viterbi smoothing	Energy threshold	Source separation
Tachibana et al. (2010b) [2011]	Two stage harmonic/percussive sound separation			Dynamic programming	Signal/noise ratio thresh.	
Poliner & Ellis (2005) [2006]	Downsample to 8kHz	STFT + limit to 2kHz + normalise magnitude	N/A	Support Vector Machine classifier	Energy threshold	Data driven
Sutton (2006) [2006]	Semitone att. + bandpass	N/A	N/A	HMM combination of monophonic pitch trackers	Confidence HMM	Monophonic

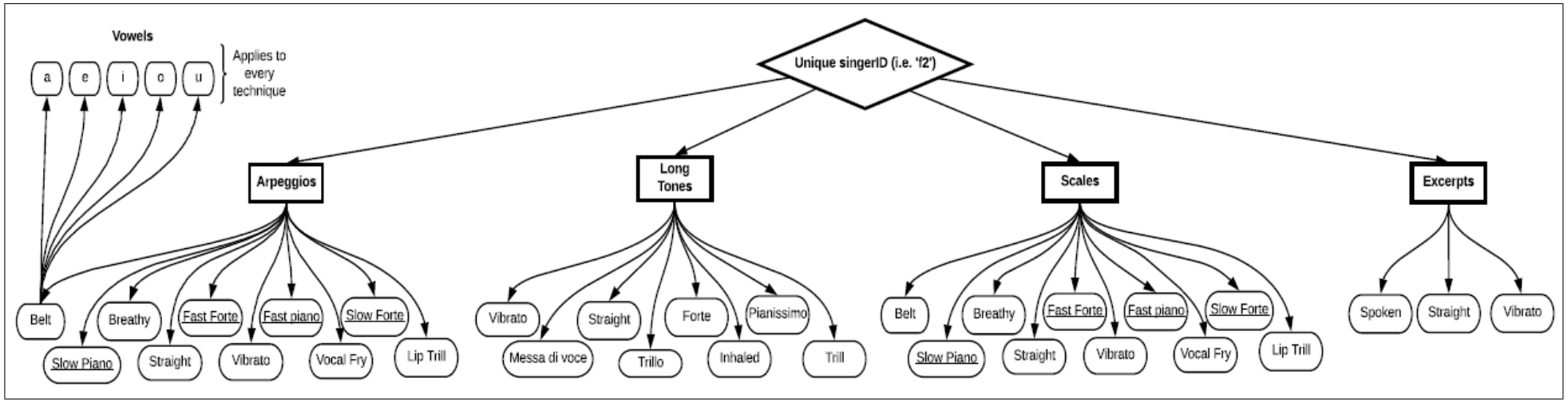
Table 2.1: Algorithm architecture of 16 melody extraction algorithms that have participated in MIREX between 2005 and 2012.

Fonte: Salamon (2013b, p. 31).

ANEXO C – TÉCNICAS DE CANTO PRESENTES NO *VOCALSET*

A Figura 80 mostra a divisão hierárquica das técnicas de canto presentes no *VocalSet* conforme proposto por Wilkins et al. (2018).

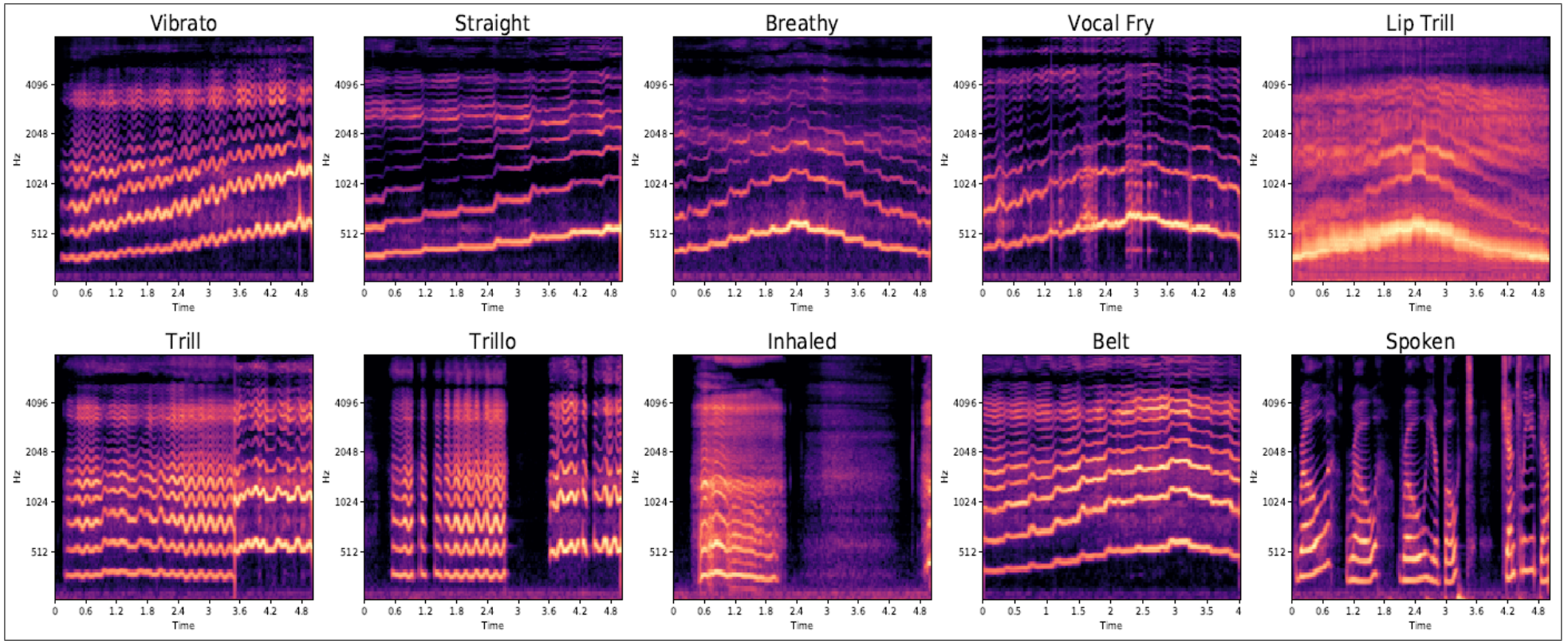
Figura 80 – Técnicas de Canto presentes no *dataset VocalSet*



Fonte: Wilkins et al. (2018).

Para ilustrar como as diferentes técnicas de canto são representadas no *domínio da frequência*, a Figura 81 contém espectrogramas em escala *mel* correspondentes a exercícios de solfejos da cantora *f2*.

Figura 81 – Exemplos de *Mel-Spectrograms* extraídos do *dataset VocalSet*



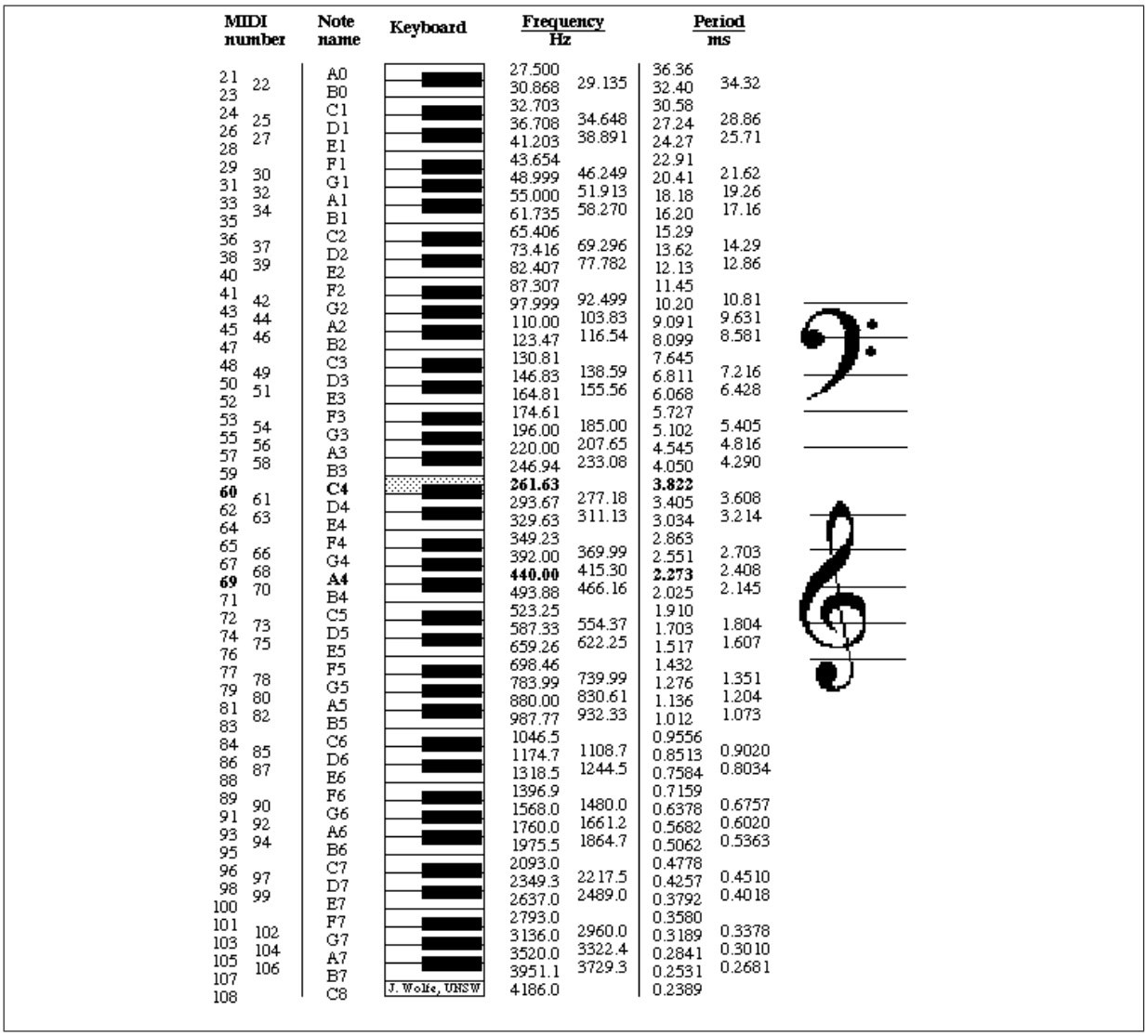
Fonte: Wilkins et al. (2018).

ANEXO D – REFERÊNCIA DE CÓDIGOS MIDI, NOTAS MUSICAIS E FREQUÊNCIAS (*HZ*)

A Figura 82 mostra a relação de códigos MIDI, o nome das Notas Musicais e as respectivas frequências (*Hz*). A frequência em *Hertz* de cada nota musical da escala ocidental (dividida em 12 semitons) pode ser obtida através da progressão geométrica calculada pela Equação D.1, onde f_n é a frequência calculada e n é o semitom desejado (WOLFE, 2005).

$$f_n = 2^{n/12} * 440Hz \tag{D.1}$$

Figura 82 – Códigos MIDI, Notas Musicais e Frequências (*Hz*)



Fonte: Adaptado de Wolfe (2005).