UNIVERSIDADE FEDERAL DA BAHIA

DISSERTAÇÃO DE MESTRADO

ESTIMATIVA DE ANDAMENTO MUSICAL ATRAVÉS DE ESCALOGRAMAS WAVELET E REDES NEURAIS CONVOLUCIONAIS

Luiz Alberto Guimarães Viana

Programa de Pós-Graduação em Engenharia Elétrica

Salvador
13 de fevereiro de 2023

LUIZ ALBERTO GUIMARÃES VIANA

ESTIMATIVA DE ANDAMENTO MUSICAL ATRAVÉS DE ESCALOGRAMAS WAVELET E REDES NEURAIS CONVOLUCIONAIS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Orientadores: Prof. Dr. Antônio C. L. Fernandes Júnior
Prof. Dr. Eduardo F. de Simas Filho

Salvador
13 de fevereiro de 2023

V614 Viana, Luiz Alberto Guimarães
Estimativa de andamento musical através de escalogramas wavelet e redes neurais convolucionais / Luiz Alberto Guimarães Viana. -- Salvador, 2023.
83 f.: il. color.

Orientador: Prof. Dr. Antônio C. L. Fernandes Júnior.
Orientador: Prof. Dr. Eduardo F. de Simas Filho.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal da Bahia, Universidade Federal da Bahia - Escola Politécnica, 2023.

1. Andamento Musical. 2. Wavelet. 3. Escalograma. 4. Rede Neural Convolucional. 5. Aumento Artificial de Dados. 6. MIR. I. Fernandes Júnior, Antônio C. L. II. Simas Filho, Eduardo F. de. III. Universidade Federal da Bahia. IV. Título.

CDD: 006.3

TERMO DE APROVAÇÃO

LUIZ ALBERTO GUIMARÃES VIANA

ESTIMATIVA DE ANDAMENTO MUSICAL ATRAVÉS DE ESCALOGRAMAS WAVELET E REDES NEURAIS CONVOLUCIONAIS

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Bahia.

Salvador, 13 de fevereiro de 2023

Prof. Dr. Antônio Carlos Lopes Fernandes Júnior
(Orientador)
UFBA

Prof. Dr. Eduardo Furtado de Simas Filho
(Coorientador)

Documento assinado digitalmente
gov.br JUGURTA ROSA MONTALVAO FILHO
Data: 08/04/2023 10:40:09-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Jugurta Rosa Montalvão Filho
UFS

Documento assinado digitalmente
gov.br RICARDO MENEZES PRATES
Data: 10/04/2023 10:44:02-0300
Verifique em https://validar.iti.gov.br

Prof. Dr.

Prof. Dr. Romis Ribeiro de Faissol Attux
UNICAMP

Dedico esta dissertação à minha esposa Micaelle e à minha mãe Celeste.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado saúde e a possibilidade de concluir mais esta etapa na minha vida. Agradeço à minha Mãe, Celeste, que sempre esteve ao meu lado e foi a base de tudo que sou hoje. Agradeço ao meu Pai, Ayrton (*In Memorian*), pelo exemplo dado e pela influência deixada.

Agradeço à minha esposa, Micaelle, por tudo: me escutar, dar ideias, opinar e me ajudar a percorrer todo este caminho. Estivemos juntos em toda esta jornada.

Agradeço à minha família e amigos pela torcida: meus irmãos Giovanni, José Pedro, Melissa, Acioli, Patrick, Melânia e suas respectivas famílias.

Um agradecimento em especial aos meus orientadores. Ao Prof. Dr. Antônio Fernandes Jr. por me direcionar desde o início da elaboração deste trabalho. Sob a sua orientação, a minha evolução profissional foi muito maior do que eu imaginava. Ao Prof. Dr. Eduardo Simas por todo o tempo dedicado às avaliações e pelos direcionamentos valiosos que me permitiram concluir este trabalho.

Aos professores que tive ao percorrer do curso por todo aprendizado adquirido.

Aos colegas e ao grupo GPAD pelas discussões ao longo da pandemia que me fizeram evoluir muito.

Agradeço à empresa ADM Engenharia, que permitiu conciliar o trabalho com o estudo do curso ao longo destes anos.

Por fim, agradeço à UFBA pela oportunidade de realizar este curso de Mestrado em Engenharia Elétrica.

"Não há nenhuma área do mundo que não deva ser investigada pelos cientistas. Sempre haverá perguntas que ainda não foram respondidas. Geralmente, essas são as perguntas que ainda não foram feitas."

—LINUS PAULING

RESUMO

A estimativa de andamento é uma das tarefas mais fundamentais da Recuperação da Informação Musical (MIR - *Musical Information Retrieval*). Neste trabalho foi utilizada uma representação do sinal de áudio como uma imagem bidimensional através do escalograma wavelet. Foram testadas diferentes formas de geração do escalograma wavelet, variando a função wavelet analisadora e os níveis de escala. As imagens foram utilizadas para treinar uma Rede Neural Convolucional (CNN - *Convolutional Neural Network*) realizando um aprendizado supervisionado, relacionando a imagem com um valor de andamento alvo. O método de validação cruzada *k-fold* foi utilizado para gerar uma maior confiabilidade estatística do modelo proposto e definir o melhor resultado para as escolhas envolvendo os parâmetros de geração dos escalogramas. Foi implementado o aumento artificial de dados de forma *online*, modificando os escalogramas durante a rotina de treinamento. Por fim, o modelo foi avaliado em bancos de dados amplamente utilizados na literatura e os resultados foram comparados ao estado da arte. Resultados compatíveis ao estado da arte foram atingidos em um dos bancos de dados de avaliação, o "GiantSteps", atingindo uma acurácia (Tipo 2 - ACC2) de 92,6% com as wavelets analisadoras Morlet e Shannon.

Palavras-chave: Andamento Musical, Wavelet, Escalograma, Rede Neural Convolucional, Aumento artificial de dados, MIR.

ABSTRACT

Audio tempo estimation is one of the most fundamental tasks in Music Information Retrieval (MIR). In this work, a wavelet scalogram is used as a two-dimensional image representation of the audio signal. Different ways of generating the wavelet scalogram were tested by varying the mother wavelet function and scale levels. The images were used to train a Convolutional Neural Network (CNN) through supervised learning, relating the image to a target tempo value. The k-fold cross-validation method was used to produce greater statistical reliability of the proposed model and to define the best result for choices involving the parameters of scalogram generation. Data augmentation was implemented online, modifying the scalograms during training. Finally, the model was evaluated on widely used databases in the literature, and the results were compared to the state-of-the-art. Results compatible with state-of-the-art were achieved on the "GiantSteps" evaluation database achieving an accuracy (Type 2 - ACC2) of 92.6% with the Morlet and Shannon mother wavelets.

Keywords: Audio Tempo Estimation, Musical Tempo, Wavelet, Scalogram, Convolutional Neural Network, Data Augmentation, MIR.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

INTRODUÇÃO

1.1 MOTIVAÇÃO

Com o avanço tecnológico dos últimos anos, os sistemas computacionais estão cada vez mais interligados através da internet. O ser humano vem modificando a forma de interagir com os diversos tipos de serviços e não é diferente com o consumo da música. Plataformas de *streaming* ganham mais espaço entre os usuários e apresentações no meio digital são comumente realizadas. Neste contexto, a extensa área de Recuperação da Informação Musical (MIR - *Musical Retrieval Information*) vem aumentado a sua importância através de tarefas como identificação de músicas, recomendações, transcrição de letras, classificação de gêneros, entre outras. Pode-se destacar a *International Society for Music Information Retrieval* (ISMIR), que é uma organização sem fins lucrativos que busca o avanço da pesquisa no campo da MIR [8].

O andamento musical é a velocidade com a qual uma peça musical é executada, em BPM (batidas por minuto), e a sua estimativa é uma das tarefas mais fundamentais da MIR [9]. Através dela é possível definir o andamento de uma peça musical, o que abre possibilidades para classificação automática e até mesmo automatização de um acompanhamento musical para uma performance ao vivo. Começando com os trabalhos de Goto e Muraoka [10] e Scheirer [11], a comunidade de MIR tem conduzido pesquisas sobre a estimativa de andamento ao longo dos últimos 25 anos [12]. Gouyon *et al.* [13] forneceram o primeiro método de validação em larga escala para algoritmos de indução de andamento que foi o critério comparativo entre os sistemas que participaram do ISMIR 2004 *Contest* [8]. Em 2011, Zapata e Goméz [14] atualizaram a visão geral sobre sistemas de estimativa de andamento. Diversos trabalhos recentes passaram a utilizar Redes Neurais combinadas com o pré-processamento do sinal de áudio. Böck e Schedl [15] introduziram um método para detecção de *onsets* baseado em Redes Neurais BLSTM (*Bidirectional Long Short-Term Memory*). Gkiokas *et al.* [16] utilizaram Redes Neurais Convolucionais (CNN - *Convolutional Neural Networks*) para estimativa de andamento e *beat tracking* em tempo real.

As Máquinas de Aprendizado Profundo (*Deep Learning*) se consolidaram como uma poderosa ferramenta devido ao recente avanço computacional, que permitiu a aquisição e

processamento de enormes bancos de dados e, por isso, vêm sendo cada vez mais aplicadas em diversas áreas e há uma convergência expressiva com a área de processamento digital de sinais. Para construir modelos de Redes Neurais com sinais de áudio como entrada é necessário decidir como os dados serão representados. Alguns trabalhos utilizaram o sinal de áudio bruto, unidimensional, para pré-processamento e extração de atributos, como Fernandes Júnior [2] e Giokas *et al.* [16]. Outros trabalhos optaram por uma representação bidimensional do sinal de áudio, gerando imagens para treinar uma CNN, como Schreiber *et al.* [7], Choi *et al.* [17] e Nasrullah *et al.* [18].

A representação de sinais de áudio como imagens, em duas ou três dimensões, pode ser realizada de diversas formas utilizando abordagens tempo-frequência: MFCCs (*Mel-Frequency Cepstrum Coefficients*), forma de onda do sinal de áudio, espectrogramas, etc. Os espectrogramas, que são gerados a partir da Transformada de Fourier, se tornaram bastante populares recentemente porque eles geram bons resultados com as Redes Neurais Convolucionais [19]. Uma alternativa para a utilização dos espectrogramas são os Escalogramas Wavelet. Fernandes Júnior [2] utilizou a transformada wavelet para o pré-processamento do sinal de áudio no problema de extração de andamento musical, mostrando que a transformada wavelet possui boa capacidade de localizar eventos ao longo do tempo, quando comparada com a Transformada de Fourier. Chen *et al.* [20] aplicou os escalogramas wavelet e conseguiu bons resultados no problema de modelagem de cenas de áudio.

Em 2020, Schreiber *et al.* [12] argumentaram que, apesar dos ótimos resultados conseguidos pelos trabalhos de Böck *et al.* [9] e Schreiber e Müller [7], o problema de estimativa de andamento musical ainda está em aberto. Isto porque as métricas utilizadas para validação desses modelos não apresentam relação direta com aplicações reais e são focadas em eliminar os chamados erros de oitava (*octave erros*). Este fato não diminui a necessidade de estudo deste tipo de métrica, mas enfatiza que os resultados para outras métricas ainda podem ser melhorados. Entre trabalhos recentes publicados podem-se destacar Souza *et al.* [21], Sun *et al.* [22] e Quinton [23]. Souza *et al.* obtiveram bons resultados com exemplos musicais exclusivamente percussivos, utilizando redes B-LSTM. Sun *et al.* obtiveram bons resultados para a Acurácia 1, utilizando a técnica de multi-scale network. Quinton trouxe uma abordagem de método auto supervisionado, onde não é necessário que os exemplos do banco estejam rotulados. Isto abre uma nova perspectiva confrontando um dos maiores problemas da estimativa de andamento musical que é a pouca quantidade de exemplos rotulados para treinamento dos modelos.

Neste contexto, este trabalho visa contribuir para o problema da estimativa de andamento musical trazendo uma forma de representação do sinal de áudio que ainda não foi utilizada para este tipo de tarefa, os escalogramas wavelets. Foi possível estudar diferentes formas para se gerar um escalograma e observar os impactos na estimativa de andamento. Neste trabalho foi utilizada a transformada contínua wavelet para gerar escalogramas a partir de sinais de áudio de peças musicais. Os sinais de áudio possuem andamentos pré-definidos, organizados em bancos de dados comumente utilizados na literatura. Os escalogramas gerados serão utilizados para o treinamento de uma Rede Neural Convolucional (CNN). Como a quantidade de exemplos nos principais bancos de dados da literatura é limitada, foi utilizada a técnica de aumento de dados, na qual os exemplos

reais sofrem modificações para que novos exemplos sejam gerados. Por fim, o método de validação cruzada k-fold foi utilizado para gerar uma maior confiabilidade estatística do modelo proposto, que foi avaliado analisando a relação entre os escalogramas e os resultados dos valores estimados. Os resultados se aproximaram do estado da arte e foi possível determinar qual wavelet se comportou melhor para os parâmetros estudados de geração dos escalogramas.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é contribuir para o problema da estimativa de andamento musical, implementando um algoritmo que utiliza escalogramas wavelet como representação de sinais musicais e, posteriormente, como entradas de redes neurais convolucionais.

1.2.2 Objetivos Específicos

Dentre os principais estudos e contribuições que foram realizados neste trabalho, destacam-se:

- Realização de pesquisa e apresentação dos principais bancos de dados utilizados na comunidade científica para a estimativa de andamento musical.
- Desenvolvimento de algoritmo para organização do banco de dados para utilização do método de validação k-fold.
- Desenvolvimento de algoritmo para padronização e aquisição das músicas do banco de dados e geração dos escalogramas wavelet.
- Implementação de arquitetura da Rede Neural Convolucional.
- Desenvolvimento de algoritmo para implementação da técnica para aumento de dados de maneira artificial, comprimindo ou expandindo os exemplos existentes e ajustando o valor do andamento musical de forma online.
- Desenvolvimento algoritmo para treinamento e validação do modelo proposto.
- Realização de comparativo entre diferentes quantidades de aumento de dados aplicados no treinamento.
- Desenvolvimento de algoritmo de classificação do andamento musical global através da geração de janelas com faixas aleatórias da peça musical.
- Realização de comparativo dos resultados entre os diferentes métodos com trabalhos publicados para a estimativa de andamento musical.

1.3 ORGANIZAÇÃO DO TEXTO

Esta dissertação é composta por 5 capítulos. O primeiro capítulo contendo a introdução ao tema e os objetivos do trabalho. O capítulo 2 traz as principais bases teóricas sobre sinais de áudio, wavelets e ferramentas de inteligência computacional utilizadas ao longo do texto. O capítulo 3 detalha o modelo proposto e toda a metodologia utilizada para implementação do trabalho. São explorados os parâmetros de geração dos escalogramas wavelet e definição dos diferentes experimentos que são implementados. O capítulo 4 mostra os resultados obtidos a partir dos experimentos propostos no capítulo anterior. Por fim, no capítulo 5 são discutidas as conclusões a respeito do modelo proposto e são feitas sugestões de trabalhos futuros.

Capítulo

2

BASES TEÓRICAS

Neste capítulo serão explanadas as principais bases teóricas necessárias para a implementação desta pesquisa. Inicialmente são abordados os principais conceitos da MIR que são essenciais para se entender como detectar informações a partir de um sinal de áudio musical. Para entender como os sinais de áudio podem ser representados como imagens, são explicadas as transformadas wavelet e os Escalogramas gerados. Por fim, são estudadas as Máquinas de Aprendizado Profundo, que são as ferramentas utilizadas para classificar as informações extraídas das imagens do áudio.

2.1 RECUPERAÇÃO DA INFORMAÇÃO MUSICAL

Um dos principais termos utilizados para definir o estudo de informações extraídas dos sinais de áudio é o "Análise de Conteúdo de Áudio" (ACA). Quando se trata de um sinal de áudio musical, este campo de pesquisa é comumente chamado Recuperação de Informação Musical (MIR), porém também englobando informações físicas ou simbólicas, como o protocolo MIDI. A informação a ser extraída é geralmente referida como metadado, que nada mais é do que dados que representam um sinal de áudio bruto. O metadado representa um determinado conteúdo musical do sinal [1].

A Figura 2.1 mostra a estrutura que pode englobar a maioria dos sistemas ACA. No primeiro bloco, atributos são extraídos do sinal de áudio e basicamente possui dois objetivos: Redução de Dimensionalidade e Representação Significativa. Quando se processa um sinal de áudio bruto, a série de dados é enorme, o que dificulta a análise de uma maneira significativa. Por isso, alguns dados irrelevantes são suprimidos. Apesar de toda a informação contida em um sinal de áudio bruto, é necessário enfocar em aspectos relevantes para que o sinal tenha uma Representação Significativa que possa facilmente ser interpretada por humanos ou máquinas.

Os atributos extraídos não precisam necessariamente possuir um significado musical e não precisam ser interpretados por humanos. Eles podem ser projetados apenas para fornecer informações condensadas para o segundo estágio de processamento. Geralmente, podem-se diferenciar em atributos de alto nível e de baixo nível. Os atributos de baixo

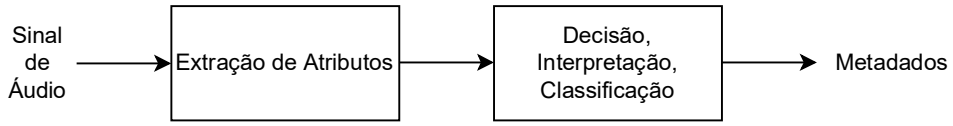


Figura 2.1 Etapas gerais de um sistema de Análise de Conteúdo de Áudio. É mostrado a entrada do sinal de áudio, o primeiro bloco para extração de atributos, o segundo bloco para Decisão, Interpretação ou Classificação destes atributos e, por fim, a saída que é definida como Metadado [1].

nível normalmente não possuem significados diretos (não são interpretados por humanos), em contrapartida, os atributos de alto nível representam formas de como os humanos se referem à música, como andamento, estrutura, gênero, etc. Estas características de alto nível são extraídas geralmente do segundo bloco de processamento.

2.1.1 Sinais de Áudio

O sinal de áudio como os humanos o percebem pode ser descrito como uma função contínua da variação do nível de pressão do ar ao longo do tempo. O microfone pode ser utilizado como transdutor, convertendo a energia acústica em sinal elétrico. Inversamente, o alto-falante pode converter o sinal elétrico em ondas sonoras [1].

Sinais Periódicos Um sinal periódico é um sinal que se repete em um intervalo de tempo constante. Matematicamente, um sinal $x_T(t)$ periódico é definido por:

$$x_T(t) = x_T(t + T_0) \tag{2.1}$$

onde T_0 é uma constante positiva. O menor valor de T_0, que satisfaz a condição de periodicidade da Equação 2.1, é o período fundamental e, sua frequência fundamental é definida por $f_0 = 1/T_0$. As frequências das outras componentes tonais, as harmônicas, possuem valores múltiplos inteiros de f_0.

Todo sinal periódico pode ser representado como uma soma de sinais senoidais, a série de Fourier:

$$x(t) = \sum_{k=-\infty}^{\infty} a(k)e^{-j\omega_0 kt} \tag{2.2}$$

a periodicidade do sinal é representada pela frequência angular $\omega_0 = 2\pi f_0$.

Sinais Aleatórios Diferentemente dos sinais periódicos, valores futuros de um sinal aleatório, ou não periódico, não podem ser previstos com total exatidão, independentemente do intervalo de tempo que o sinal está sendo observado. Cada materialização deste tipo de sinal é denominada de observação ou realização.

Um processo aleatório cujas características estatísticas não se alteram com o tempo é classificado como um processo aleatório no sentido estrito (SSS - *Strict Sense Stationary*).

Isto significa que os processos $x(t)$ e $x(t+c)$ possuem as mesmas características estatísticas apara qualquer valor de c [24].

O sinal de áudio musical é um exemplo de processo não estacionário, pois suas características estatísticas variam ao longo do tempo. A variação pode ocorrer em diversas dimensões, como amplitude, frequência e fase. Contudo, pode-se dizer que o processo de ruído é estacionário, pois suas estatísticas (valor médio e valor quadrático médio, por exemplo) não variam com o tempo.

É possível que um processo que não seja estacionário no sentido estrito e tenha valor médio e função de autocorrelação que independam do deslocamento da origem do tempo. Um processo desse tipo é conhecido como processo estacionário no sentido amplo (WSS - *Wide Sense Stationary*). Na prática, todos os processos são não estacionários, pois devem ter início em algum tempo finito e devem terminar em algum tempo finito. Um processo verdadeiramente estacionário deve iniciar em $t = -\infty$ e durar eternamente. Todavia, muitos processos podem ser considerados estacionários para o intervalo de tempo de interesse, e a hipótese de estacionariedade possibilita um modelo matemático maneável [25].

Amostragem e Quantização Um sinal em tempo contínuo pode ser processado, a partir de suas amostras, por um sistema que opere em tempo discreto. Para isso, é importante manter a taxa de amostragem do sinal suficientemente alta para permitir a reconstrução sem erro (ou com um erro dentro de uma dada tolerância) do sinal original. O teorema da amostragem é a ponte entre os domínios de tempo contínuo e de tempo discreto.

A discretização no tempo é feita pela amostragem do sinal em tempos específicos. O período de amostragem T_S, em segundos, geralmente é constante nos sinais de áudio e pode ser obtido diretamente da taxa de amostragem f_S, conforme equação 2.3.

$$T_S = \frac{1}{f_S} \tag{2.3}$$

Uma importante propriedade dos sinais discretizados é que a representação discreta é ambígua. Diferentes sinais de entrada podem ter a mesma série de amostras, a depender da taxa de amostragem. Um sinal real cujo espectro é limitado em faixa a B Hz pode ser reconstruído exatamente de suas amostras tomadas uniformemente a uma taxa de $f_S \geq 2.B$ amostras por segundo. A menor taxa de amostragem, $f_S = 2B$, necessária para recuperar o sinal analógico a partir de suas amostras é chamada taxa de Nyquist [26]. Em muitas aplicações, no entanto, amostrar na taxa de Nyquist é ineficiente porque os sinais de interesse contêm apenas um pequeno número de frequências significativas em relação ao limite de banda [27].

Um sinal amostrado ainda não é um sinal digital, isto porque a sua amplitude pode assumir infinitos valores. Desta forma, é necessário realizar a aproximação do valor de sua amplitude para o valor mais próximo possível dos números passíveis de representação numérica por intermédio do processo de Quantização. Para L níveis de quantização, precisa-se de um mínimo de b dígitos do código binário, tal que $2^b = L$ ou $b = log_2 L$ [26].

Transformada de Fourier Um sinal aleatório pode ser visto como um sinal periódico

com período infinito. À medida que o período se torna infinito, os componentes de frequência se aproximam de modo a formar um conjunto contínuo e a soma da série de Fourier torna-se uma integral [28]. A transformada de Fourier pode ser entendida como uma extensão da série em que uma função não periódica é considerada periódica no intervalo $[-L/2, L/2]$, com L tendendo ao infinito [29]. A Transformada de Fourier é amplamente utilizada na análise dos sinais de áudio. Entender seus conceitos e propriedades é crucial para o desenvolvimento de sistemas de processamento de áudio [1].

Simplificadamente, para a análise de sinais aleatórios, podemos considerar o período fundamental $T_0 \to \infty$. A equação 2.2 para a série de Fourier se torna então na transformada de Fourier ($\mathfrak{F}$) exposta na equação 2.4:

$$X(j\omega) = \mathfrak{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt \tag{2.4}$$

onde $X(j\omega)$ é chamado de espectro do sinal $x(t)$. Mais detalhes podem ser obtidos em Oppenheim *et al.* [28] ou Lathi [26].

O espectro $X(j\omega)$ pode ser convertido de volta ao domínio do tempo aplicando a Transformada Inversa de Fourier ($\mathfrak{F}^{-1}$):

$$x(t) = \mathfrak{F}^{-1}\{X(j\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega \tag{2.5}$$

Isso significa que as representações em tempo e frequência são equivalentes. Nenhuma informação é perdida aplicando-se a Transformada de Fourier ao sinal [1].

É possível se deparar com situações em que seja necessário calcular o espectro de frequência de um sinal apenas em uma certa região temporal. Nessa situação a transformada de Fourier, como apresentada na equação 2.4, não será uma ferramenta apropriada, pois levará em conta a duração completa do sinal. Num esforço para superar esta dificuldade, Dennis Gabor, em 1946, fez uma adaptação à Transformada de Fourier de maneira que simplesmente uma parte do sinal no tempo fosse analisada [30]. A técnica em questão ficou conhecida como *Short Time Fourier Transform - STFT* e consiste em aplicar uma função janela à transformada de Fourier original:

$$STFT(\omega, \tau) = \int_{-\infty}^{\infty} f(t)w(t-\tau)e^{-j\omega t}dt \tag{2.6}$$

$w(t)$ é a função janela cuja posição é deslocada através da variável τ. A STFT transforma um sinal no domínio do tempo em uma função bidimensional, permitindo uma análise tempo-frequência.

Uma típica visualização de um sinal de áudio que combina os domínios tempo e frequência é o espectrograma [1]. Um dos arquivos que será utilizado ao longo do trabalho, o ACM Mirum 148.clip, é utilizado como exemplo. A Figura 2.2 mostra este sinal de áudio ao longo do tempo, onde o eixo horizontal representa o tempo em segundos e o eixo vertical a amplitude do sinal. Após se aplicar a STFT, é possível se gerar o espectrograma deste sinal, como mostra a Figura 2.3. No espectrograma é possível visualizar o tempo, em segundos, ao longo do eixo horizontal e a frequência, em hertz, ao longo do eixo vertical. As cores representam a magnitude do sinal.

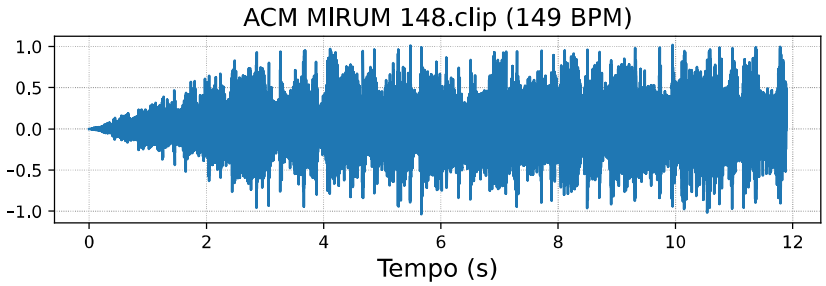


Figura 2.2 Representação dos 12 segundos iniciais da forma de onda do sinal de áudio do exemplo 148.clip, relativo ao banco de dados ACM Mirum. A amplitude da forma de onda, representada pelo eixo vertical, evolui ao longo de tempo, representado pelo eixo horizontal.

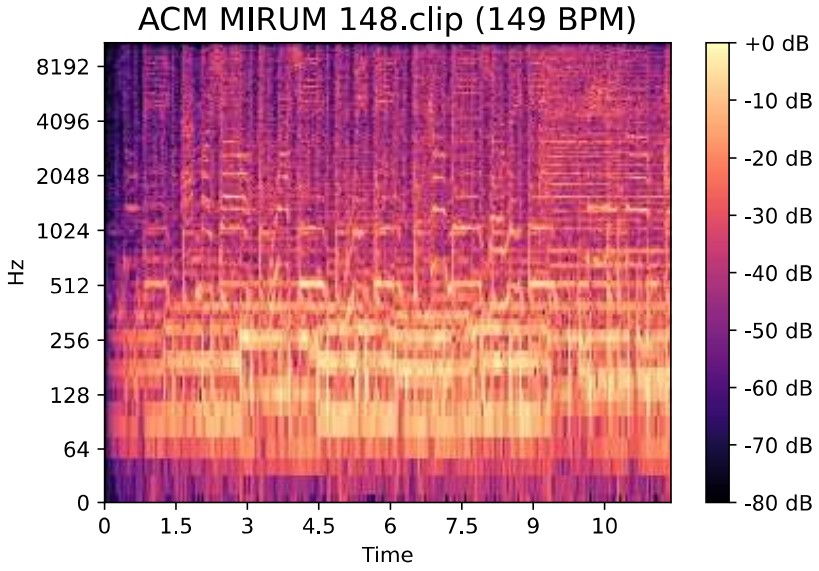


Figura 2.3 Espectrograma gerado a partir da STFT aplicada ao exemplo ACM Mirum 148.clip, mostrado na Figura 2.2. O espectrograma mostra as frequências, no eixo vertical, evoluindo ao longo do tempo, eixo horizontal. A paleta de cores mostra a intensidade em decibéis.

2.1.2 Percepção Humana de Eventos Temporais

Os aspectos temporais de sinais de áudio, como andamento e ritmo, são importantes propriedades musicais. Uma parte fundamental destes aspectos é o *onset*: o início de um evento sonoro musical, como uma nota de um instrumento harmônico ou uma batida em um instrumento percussivo. O momento inicial de um evento é considerado mais importante do que o tempo de duração do mesmo evento, pois os ouvintes aparentemente percebem os eventos musicais em termos de intervalos entre *onsets* [31].

Durante o processo da percepção humana, o fluxo de áudio pode ser segmentado como uma série de eventos. A segmentação pode ser considerada uma simplificação porque o

significado musical, e até o ritmo, podem ser transmitidos por uma série de fluxos de áudio sem uma divisão clara dos eventos. Entretanto, esta simplificação é suficiente para a grande maioria das músicas ocidentais [32] [1].

Onsets O início de um som musical não é um ponto exato no tempo, mas sim um evento espraiado. O modelo geral da evolução temporal de um som passa por fases como o ataque (A), o decaimento (D), a sustentação (S), e o relaxamento (R) (ADSR) como mostrado na Figura 2.4. O termo *onset* é usado com frequência como sinônimo do instante de tempo do *onset*, porém é mais preciso definir que seu instante de tempo é sua propriedade mais importante [2].

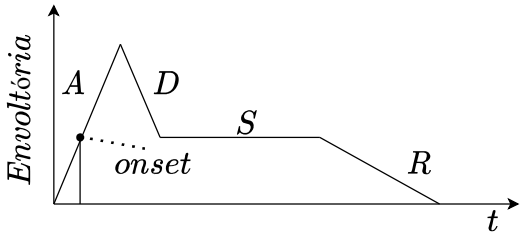


Figura 2.4 Evolução temporal da envoltória de uma nota musical, indicando as fases de ataque (A), decaimento (D), sustentação (S) e relaxamento (R). O *onset* ocorre pouco após o início do ataque [2].

O tempo de ataque ou subida é o intervalo entre a primeira oscilação de um instrumento musical e uma amplitude máxima. Os instrumentos musicais exibem tempo de ataque de aproximadamente 5 ms para instrumentos de percussão e até 200 ms para instrumentos de sopro [1]. O uso dos termos *onset*, ataque e transiente na literatura é, em alguns momentos, inconsistente. Por isso, utilizaremos aqui as seguintes definições [2]:

1. O ataque do evento sonoro é o intervalo de tempo no qual sua envoltória atinge a amplitude máxima, após o início do evento;

2. O transiente do evento sonoro é o intervalo de tempo no qual decorrem as fases de ataque e decaimento;

3. Tempo de *onset* da nota (NOT): instante de tempo em que o instrumento é excitado para produzir o som;

4. Tempo de *onset* acústico (AOT): instante de tempo em que o som pode ser medido;

5. Tempo de *onset* perceptual (POT): instante de tempo em que o evento é percebido pelo ouvinte.

A escolha entre AOT e POT é dependente da aplicação. Assumindo que os músicos se adaptam ao tempo da música a partir de sua percepção auditiva, utilizaremos POT como sinônimo de *onset*. A Figura 2.4 mostra uma possível localização para o POT.

Andamento O andamento musical (ou simplesmente "tempo") é a taxa na qual os pulsos do sinal de áudio são percebidos, com unidades de duração iguais, ocorrendo em uma taxa moderada e natural. Este andamento percebido é chamado de *tactus* e às vezes é simplesmente referido como a taxa de batidas do pé. Uma taxa natural típica estaria localizada em torno de 100 batidas por minuto (BPM) [1].

Para segmentos de música com andamento constante, o andamento Γ em BPM pode ser calculado usando o comprimento do segmento Δt_s, em segundos, e o número de batidas $\mathfrak{B}$ no segmento:

$$\Gamma = \frac{\mathfrak{B} \cdot 60s}{\Delta t_s}[BPM] \tag{2.7}$$

Se o andamento se mantém constante durante toda a peça musical, isto é chamado de *andamento global*. O andamento global geralmente ocorre em gêneros como *Rock*, *Pop* e *Dance Music* [7]. No caso de um andamento dinâmico, o andamento local pode ser extraído identificando o momento em que cada batida ocorre (t_b) e calculando a distância entre duas batidas vizinhas com índices j e $j+1$.

$$\Gamma_{local}(j) = \frac{60s}{t_b(j+1) - t_b(j)}[BPM] \tag{2.8}$$

Extrair o andamento global se torna cada vez mais difícil quando o andamento não é constante. Neste caso, o andamento médio dado na equação 2.7 não é necessariamente o mesmo que o andamento percebido por um ouvinte.

Métrica e Ritmo Assim como uma melodia é mais do que simplesmente uma série de tons, o ritmo é mais do que uma mera sequência de proporções temporais. Perceber o ritmo é agrupar sons separados em padrões estruturados. Tal agrupamento é o resultado da interação entre os vários aspectos dos materiais da música: tom, intensidade, timbre, textura e harmonia, assim como a duração [33].

O ritmo é definido através das propriedades de agrupamento da peça musical. Os grupos de duração entre uma batida e a duração da métrica são mais comumente chamados de ritmo. O ritmo é então definido por seus acentos musicais e intervalos de tempo. O acento musical nada mais é que a nota na qual recai uma intensidade distinta de execução das demais notas [1].

A métrica é o padrão rítmico constituído pelo agrupamento de unidades temporais básicas, chamadas batidas, em medidas regulares ou compassos. Uma assinatura de tempo (*time signature*), ou fórmula de compasso musical, é encontrada no início de uma peça musical. A métrica indica o número de batidas em um compasso e duração de cada batida. Por exemplo, a medida 3/4 tem três batidas de semínima por compasso. A assinatura de tempo implica que um acento ocorre regularmente na primeira batida de cada compasso.

2.2 WAVELETS

As wavelets estendem os conceitos da análise de Fourier para bases ortogonais mais

gerais, explorando uma decomposição de multirresolução. A Figura 2.5 mostra os esquemas de resolução para cada tipo de análise. A Figura 2.5 (a) mostra a resolução para uma série temporal. A Figura 2.5 (b) mostra a resolução de frequência quando aplicada a transformada de Fourier. A Figura 2.5 (c) mostra o espectrograma, onde é possível a análise tempo-frequência em intervalos constantes Δt e $\Delta\omega$. E, por fim, a Figura 2.5 (d) mostra a análise wavelet multiresolução, onde os valores dos intervalos não são fixos.

Esta abordagem permite uma maior fidelidade de tempo e frequência em bandas diferentes, o que é particularmente útil para decompor sinais complexos que surgem de processos multiescala, como os encontrados em climatologia, neurociência, epidemiologia, finanças, etc. Imagens e sinais de áudio também são passíveis de análise wavelet, que atualmente é utilizada nos principais métodos de compressão [3].

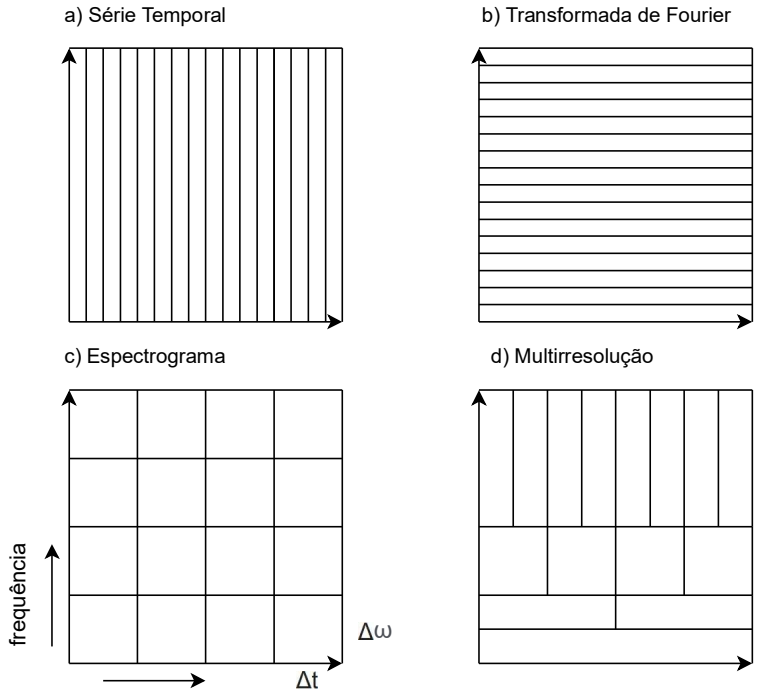


Figura 2.5 Esquemas de resolução para cada tipo de análise tempo-frequência. A figura (a) mostra a resolução para uma série temporal. A figura (b) mostra a resolução de frequência quando aplicada a transformada de Fourier . A figura (c) mostra o espectrograma, na qual é possível a análise tempo-frequência em intervalos Δt e $\Delta\omega$. A figura (d) mostra a análise wavelet multirresolução, onde o valor dos intervalos não são fixos [3].

Uma das principais ferramentas da teoria wavelet é a transformada wavelet. A transformada wavelet equivale a um microscópio matemático cuja ampliação é dada pelo inverso do parâmetro de dilatação, designado a, e a capacidade óptica pela escolha da função wavelet analisadora escolhida, designada ψ [29].

2.2.1 Transformada Wavelet Contínua

A transformada wavelet contínua (*continuous wavelet transform* - CWT) é a transformada integral linear que pode ser utilizada na exploração de características de sinais não estacionários para extrair informações de variações em certas bandas de frequência e/ou detectar estruturas locais presentes [29].

Dado um sinal $f(t)$, sua transformada integral é definida como:

$$\mathcal{W}_f^{\psi}(a,\tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)\psi^* \left(\frac{t-\tau}{a}\right) dt \tag{2.9}$$

em que o parâmetro $a(>0)$ se refere à escala e τ à translação ou localização da função analisadora wavelet ψ, ou wavelet-mãe, sendo a e $\tau \in \mathbb{R}$. O parâmetro a controla a dilatação/contração da função analisadora wavelet. À medida que o parâmetro τ varia, o sinal $f(t)$ é analisado localmente em torno dele. Assim, podem-se analisar os aspectos multiescala do sinal não estacionário estudado. O asterisco superior em ψ^* denota o complexo conjugado da função ψ. $\mathcal{W}_f^{\psi}(a,\tau)$ é uma matriz de magnitudes $m \times n$, em que m e n indicam respectivamente o número de escalas e o comprimento do sinal $f(t)$. Esta matriz é conhecida como matriz dos coeficientes wavelets [34]. Esta transformada é dita contínua, pois esses parâmetros de escala a e de localização τ assumem valores contínuos nessa representação [29].

Para os fins práticos da análise de sinais discretos, tais valores podem ser discretizados tendo o cuidado de se manter o plano escala-tempo discreto totalmente preenchido. Isso resulta numa representação redundante do sinal analisado em relação aos seus coeficientes wavelets. Por isso, a implementação direta dessa transformada é pouco eficiente do ponto de vista computacional. Contudo, esse fato auxilia bastante o estudo exploratório de sinais [29].

Escolhida a função wavelet, é possível obter uma constante c_ψ de modo que a transformada wavelet admita a transformada inversa:

$$I\mathcal{W}_f^{\psi}(t) = \frac{1}{c_\psi} \int_{-\infty}^{\infty} \int_{0}^{\infty} \mathcal{W}_f^{\psi}(a,\tau)\psi^* \left(\frac{t-\tau}{a}\right) \frac{da}{a^2} d\tau \tag{2.10}$$

Essa constante c_ψ é conhecida como constante de admissibilidade da função wavelet, conforme equação 2.12.

2.2.2 Funções Wavelets

Existem diversas funções analisadoras wavelets que podem ser utilizadas na análise de sinais. Essas escolhas podem alterar os resultados obtidos de forma a enfatizar uma certa característica do sinal analisado. Seja uma função $\psi(t)$ pertencente ao espaço das funções mensuráveis reais $\mathbb{R}^2$, ela pode ser admitida como wavelet se forem atendidas as seguintes condições [34]:

1. Condição de admissibilidade: também é conhecida como propriedade de média nula. Ela garante que ψ apresente um comportamento oscilatório.

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \tag{2.11}$$

que também pode ser expressa no domínio de Fourier como:

$$c_\psi = 2\pi \int_{-\infty}^{\infty} \frac{|\Psi(j\omega)|^2}{|\omega|} d\omega < \infty \tag{2.12}$$

em que Ψ é a transformada de Fourier da função wavelet ψ. Essa característica oscilatória dá origem ao nome wavelet. O termo original veio do francês *ondelette*, proposto pelo físico Alex Grosman [29].

2. Energia unitária: esta condição impõe uma normalização da medida de energia da função wavelet.

$$\int_{-\infty}^{\infty} |\psi(t)|^2 \, dt = 1 \tag{2.13}$$

Podem-se destacar as wavelets analisadoras, do tipo contínuas, que serão utilizadas neste trabalho [35]:

- Wavelet Chapéu Mexicano:

$$\psi(t) = \frac{2}{\sqrt{3}\sqrt[4]{\pi}} e^{-\frac{t^2}{2}} (1 - t^2) \tag{2.14}$$

- Wavelet Morlet:

$$\psi(t) = e^{-\frac{t^2}{2}} cos(5t) \tag{2.15}$$

- Wavelet Shannon:

$$\psi(t) = \sqrt{B} \frac{sen(\pi Bt)}{\pi Bt} e^{j2\pi Ct} \tag{2.16}$$

onde B é a largura de banda e C a frequência central.

2.2.3 Escalograma

Os escalogramas são gráficos que representam a visualização bidimensional da matriz dos coeficientes wavelets $\mathcal{W}_f^\psi(a, \tau)$. Estes podem ser visualizados por meio de um campo de isolinhas ou imagem. Analogamente aos espectrogramas da STFT, que são representações das variações das amplitudes ou das energias dos coeficientes de Fourier em bandas de frequência fixas no tempo, os escalogramas wavelets são a representação das amplitudes ou das energias associadas aos coeficientes wavelet simultaneamente em escalas no tempo [29].

A Figura 2.6 mostra um escalograma wavelet gerado a partir da forma de onda do áudio da Figura 2.2. Esse escalograma foi gerado após a aplicação da CWT utilizando a

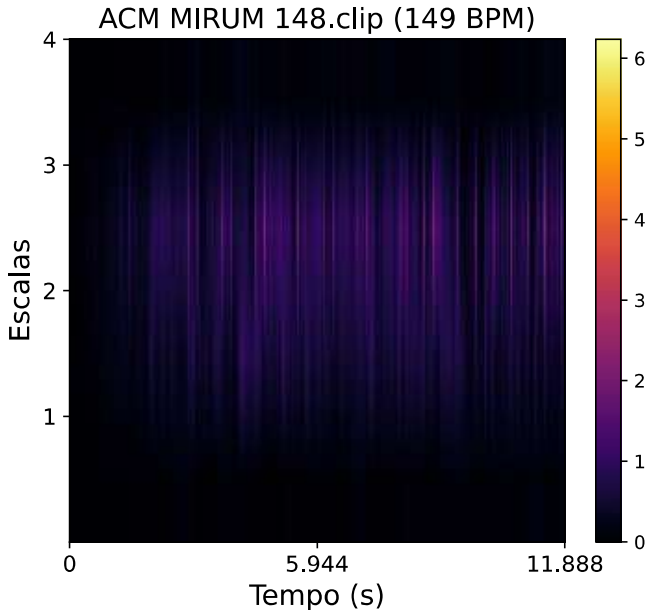


Figura 2.6 Escalograma gerado a partir da CWT aplicada ao exemplo ACM Mirum 148.clip, mostrado na Figura 2.2. Foi utilizada a função analisadora wavelet Chapéu Mexicano com quatro níveis de escala [1.3, 11, 45.5, 130]. A paleta de cores mostra a intensidade de cada pixel, que representa os valores dos coeficientes wavelets.

função analisadora wavelet Chapéu Mexicano, com quatro níveis de escala [1.3, 11, 45.5, 130]. O eixo horizontal do escalograma representa o tempo, em segundos, enquanto que o eixo vertical mostra a representação discreta dos níveis de escala.

A paleta de cores representa os valores dos coeficientes wavelets, onde os valores mais próximos a zero são representados por cores mais escuras e os valores mais altos por cores mais claras. No exemplo da Figura 2.6 é possível observar que em alguns instantes curtos de tempo, os valores atingem picos de intensidade em determinados valores de escala. Quando se observa um intervalo de tempo pequeno, é possível calcular o andamento local através da equação 2.8, utilizando a distância entre os picos. Porém, na maioria das vezes, a complexidade da peça musical dificulta a determinação do valor do andamento musical quando se observa um intervalo de tempo maior. Um pico em uma determinada escala pode estar periodicamente relacionado com um pico em outra escala.

Domingues *et al.* [29] discutem que a escolha da paleta de cores interfere no escalograma observado, podendo destacar diferentes informações quando alterada. Neste trabalho, optou-se por utilizar a escala de cores presente na Figura 2.6 sempre que necessária uma discussão visual do escalograma. Porém, para o treinamento e interpretação da rede neural convolucional, utilizou-se o escalograma da Figura 2.7. A escala em tons de cinza facilita o processamento do modelo e as magnitudes foram convenientemente ajustadas para 256 pixels no eixo horizontal, e 40 pixels no eixo vertical, que é a entrada da rede neural convolucional.

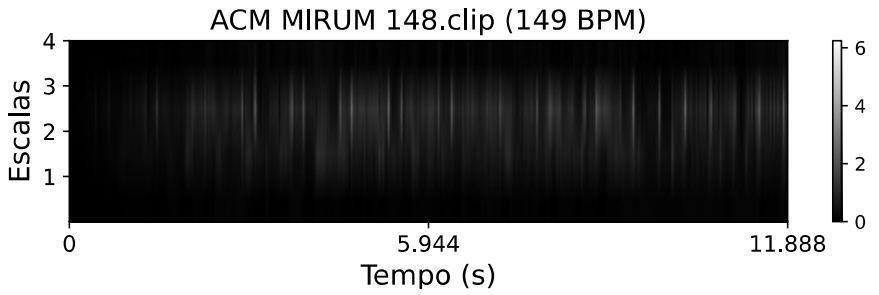


Figura 2.7 Escalograma gerado a partir da CWT aplicada ao exemplo ACM Mirum 148.clip da mesma forma que a Figura 2.6. Este escalograma possui as características das imagens que serão usadas no treinamento a rede: tons de cinza e magnitudes (40,256,1).

2.3 MÁQUINAS DE APRENDIZADO PROFUNDO

A Inteligência Artificial (IA) começou com um campo experimental na década de 1950, quando pioneiros do recente campo de pesquisa da ciência da computação começaram a se perguntar se computadores poderiam ser feitos para "pensar". Uma definição concisa do campo seria: o esforço para automatizar tarefas intelectuais normalmente realizadas por humanos. A IA é um campo geral que engloba não somente o aprendizado de máquina e o aprendizado profundo, mas também outras abordagens que não envolvem nenhum aprendizado [4]. O termo aprendizado profundo, ou *Deep Learning*, é um tipo específico de aprendizado de máquina, ou *Machine Learning* [6].

2.3.1 Aprendizado de Máquina

Um algoritmo de aprendizado de máquina nada mais é que um algoritmo que é capaz de aprender a partir de um conjunto de dados (ou *dataset*). Chollet [4] evidenciou a mudança de paradigma de programação, conforme Figura 2.8. Na programação clássica, humanos inserem regras (programa) e dados para serem processados de acordo a estas regras e gerar as respostas na saída. Com o aprendizado de máquina, humanos inserem os dados com as respostas esperadas e a saída são as regras. Estas regras podem ser aplicadas em novos dados para produzir respostas originais.

O processo de aprendizado em si não é a tarefa. Aprender é o meio de atingir a capacidade de realizar a tarefa. As tarefas geralmente são descritas em termos de como o sistema de aprendizado de máquina deve processar um exemplo. Um exemplo é um conjunto de atributos que foram medidos quantitativamente a partir de algum objeto ou evento que desejamos que o sistema de aprendizado de máquina processe. Normalmente representamos um vetor $\boldsymbol{x} \in \mathbb{R}^n$ onde cada componente x_i do vetor é um atributo. Por exemplo, os atributos de uma imagem são geralmente os valores dos pixels desta imagem [6].

Goodfellow *et al.* [6] discrimina diversos tipos de tarefas que podem ser resolvidas com o aprendizado de máquina, são elas: classificação, classificação com entradas ausentes, regressão, transcrição, máquina de tradução, saídas estruturadas, detecção de anomalia, síntese e amostragem, imputação de valores ausentes, eliminação de ruído e estimativa de

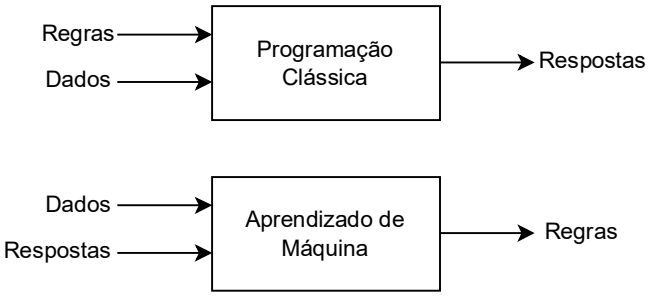


Figura 2.8 Diferenças entre os paradigmas da programação clássica e do apredizado de máquina. Na programação clássica, têm-se regras e dados como entradas e respostas como saídas, enquanto que no aprendizado de máquina, têm-se dados e respostas como entradas e regras como saídas [4].

densidade. A estimativa de andamento musical pode ser interpretada de algumas formas. Podem-se destacar:

- Classificação: neste tipo de tarefa, o programa de computador é solicitado a especificar a qual das k categorias uma entrada pertence. Para resolver esta tarefa, o algoritmo de aprendizagem é geralmente solicitado a produzir uma função $f : \mathbb{R}^n \rightarrow \{1, ..., k\}$. Quando $y = f(\boldsymbol{x})$, o modelo atribui uma entrada descrita pelo vetor $\boldsymbol{x}$ a uma categoria identificada pelo código numérico y. Existem outras variantes da tarefa de classificação, por exemplo, onde f gera uma distribuição de probabilidade sobre as classes. Um exemplo de tarefa de classificação é o reconhecimento de objeto, em que a entrada é uma imagem (geralmente descrita como um conjunto de valores de brilho do pixel) e a saída é um código numérico que identifica o objeto na imagem. O reconhecimento de objetos é a mesma tecnologia básica que permite que os computadores reconheçam rostos, que pode ser usada para marcar pessoas automaticamente em coleções de fotos e permitir que os computadores interajam mais naturalmente com seus usuários [6].

- Regressão: Neste tipo de tarefa, o programa de computador é solicitado a prever um valor numérico dado alguma entrada. Para resolver esta tarefa, o algoritmo de aprendizagem é solicitado a produzir uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Este tipo de tarefa é semelhante à classificação, exceto que o formato de saída é diferente. Em vez de um vetor que indica a categoria, a saída é apenas um número real. Um exemplo de uma tarefa de regressão é a previsão do valor de um imóvel baseado em diferentes características de outros imóveis com preços definidos [6].

2.3.2 Aprendizado Supervisionado

Algoritmos de aprendizado de máquina são geralmente divididos em quatro categorias [5]: Aprendizado Supervisionado, Aprendizado Não Supervisionado, Aprendizado Semi Supervisionado e Aprendizado por Reforço.

O foco deste trabalho é o aprendizado supervisionado. Chollet (2018) [4] diz que a maioria das aplicações que estão em evidência atualmente fazem parte desta categoria, como reconhecimento óptico, reconhecimento de fala, classificação de imagens e tradução de idiomas.

Um algoritmo de aprendizado supervisionado adquire experiência a partir de um *dataset* contendo não somente atributos, mas também rótulos (ou *labels*) para cada exemplo. O aprendizado envolve observar vários exemplos de um vetor aleatório, ou entrada, $\boldsymbol{x}$ e um valor associado, ou alvo, $\boldsymbol{y}$ e aprender a prever $\boldsymbol{y}$ a partir de $\boldsymbol{x}$, geralmente estimando $p(\boldsymbol{y}|\boldsymbol{x})$ [6].

2.3.3 Conjunto de Dados de Treinamento, Validação e Teste

Avaliar um modelo, geralmente, se resume em dividir os dados disponíveis em três conjuntos: treinamento, validação e teste. Atualmente, este é o padrão adotado pela maioria dos pesquisadores para avaliação de modelos de máquinas de aprendizado profundo. O modelo é treinado com os dados de treinamento e avaliado nos dados de validação. Após resultados satisfatórios nos dados de validação, o modelo é testado uma última vez nos dados de teste [4].

Geralmente, durante o treinamento, o desempenho do modelo nos dados de validação atinge um pico e começa a degradar, enquanto que o desempenho nos dados de treinamento continua a aumentar. Ou seja, o modelo começa a se ajustar demais aos dados de treinamento. Quando isso ocorre é chamado de *overfitting*. O *overfitting* pode ocorrer em todos os problemas de aprendizado de máquina [4].

Desenvolver um modelo envolve ajustar as suas configurações, ou *hiperparâmetros*. Um hiperparâmetro é um parâmetro do algoritmo de aprendizado que não é afetado pelo treinamento [5]. Como exemplos, pode-se citar: quantidade de camadas, quantidade de neurônios, funções de ativação, função custo, otimizadores, etc. Os valores dos hiperparâmetros precisam ser sintonizados para que o modelo atinja a melhor performance. A busca de um melhor desempenho do modelo no conjunto de dados de validação pode resultar em *overfitting* para o conjunto de validação, mesmo que o modelo nunca tenha sido diretamente treinado nele [4]. Por isso, a avaliação no conjunto de dados de teste mostra o desempenho do modelo em dados que realmente nunca foram vistos.

Dividir o conjunto de dados em um conjunto de treinamento fixo e um conjunto de teste fixo pode gerar dados inconsistentes quando o *dataset* possui uma quantidade pequena de exemplos. Um pequeno conjunto de dados de teste implica em incerteza estatística em torno do erro de teste médio estimado, tornando difícil afirmar que o algoritmo A funciona melhor do que o algoritmo B em uma dada tarefa [6]. Este tipo de validação simples é conhecida como *hold-out* [4].

Quando o conjunto de dados possui uma quantidade de exemplos muito superior à quantidade de parâmetros ajustáveis, a validação *hold-out* não é um problema grave. Isto é relativo à quantidade de graus de liberdade, do modelo a ser utilizado. Gao e Jojic [36] mostraram que para modelos simples, a quantidade de parâmetros é igual aos graus de liberdade do modelo. Entretanto, em redes profundas, o número de graus de liberdade do modelo é, geralmente, muito menor do que o número de parâmetros. Götz *et al.*

[37] mostrou empiricamente que um modelo superdimensionado, quando comparada a quantidade de parâmetros à quantidade de exemplos, geralmente atinge bons resultados em problemas de classificação.

Quando a quantidade de dados é pequena, existem métodos alternativos que permitem usar todos os exemplos na estimativa do erro médio do teste. O mais comum deles é o método de validação cruzada *k-fold* [6]. Com o método *k-fold*, divide-se o *dataset* em k partes de tamanho igual. Para cada parte i o modelo é treinado com as partes $k-1$ restantes e validado na parte i. Seu valor de desempenho final é a média dos resultados obtidos para cada valor de k. Este método é útil quando o desempenho do modelo mostra uma variância significativa com base na divisão do *dataset* entre treinamento e validação [4].

A Figura 2.9 mostra um esquema da implementação do método de validação cruzada *k-fold*, com $k = 5$. Neste caso, o conjunto é dividido em cinco partes e são realizados cinco treinamentos. Para $k = 1$, o conjunto é treinado com as partes 2, 3, 4 e 5 e validado na parte 1. Para $k = 2$, o conjunto é treinado com as partes 1, 3, 4 e 5 e validado na parte 2, e assim sucessivamente para $k = 3$, 4 e 5. O resultado final é a média entre os cinco treinamentos realizados.

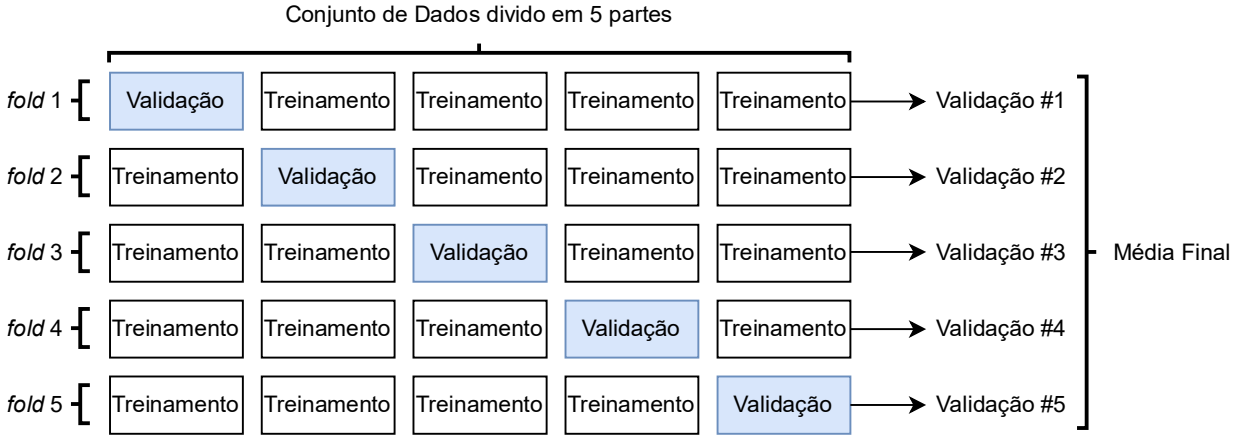


Figura 2.9 Método de validação cruzada *k-fold* dividido em 5 partes que mostra o conjunto escolhido para validação em cada treinamento. O resultado final é a média entre os resultados para cada *fold* [4].

A questão principal do aprendizado de máquina é o equilíbrio entre otimização e generalização. Otimização se refere ao processo de ajustar um modelo para obter o melhor desempenho possível nos dados de treinamento, enquanto que a generalização se refere ao desempenho do modelo treinado em dados que nunca foram vistos. O objetivo é obter uma boa generalização, mas só é possível ajustar o modelo com base nos dados de treinamento.

Para evitar que um modelo aprenda padrões enganosos, ou irrelevantes, encontrados nos dados de treinamento, uma solução é obter mais dados de treinamento. Um modelo treinado com mais dados irá, geralmente, generalizar melhor. Quando isso não for

possível, a melhor solução é modular a quantidade de informações que o modelo tem permissão para armazenar ou adicionar restrições sobre quais informações ele tem permissão para armazenar. Se uma rede só puder memorizar um pequeno número de padrões, o processo de otimização a forçará a se concentrar nos padrões mais proeminentes, que têm uma chance melhor de generalização. O processamento de mitigação ao *overfitting* desta forma é chamado de *regularização* [4].

2.3.4 Redes Neurais Profundas

Perceptrons Em 1958, Rosenblatt propôs o *perceptron* como primeiro modelo para aprendizagem supervisionada. O perceptron é a forma mais simples de uma rede neural usada para classificação de padrões ditos *linearmente separáveis*, ou seja, padrões que se encontram em lados opostos de um hiperplano [38].

Este modelo é baseado em neurônios artificiais, conforme Figura 2.10. As entradas e saídas são números e cada conexão de entrada é associada a um peso. O neurônio calcula a soma ponderada das entradas ($z = w_1x_1 + w_2x_2 + ... + w_nx_n = \boldsymbol{x}^T\boldsymbol{w}$, onde $\boldsymbol{x^T}$ é o vetor de entrada $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ transposto e $\boldsymbol{w} = [w_1, w_2, ..., w_n]$ o vetor de pesos de cada conexão) e aplica uma função de ativação ϕ, resultando na hipótese $h_{\boldsymbol{w}}(\boldsymbol{x})$, onde $h_w(\boldsymbol{x}) = \phi(\boldsymbol{x}^T\boldsymbol{w})$.

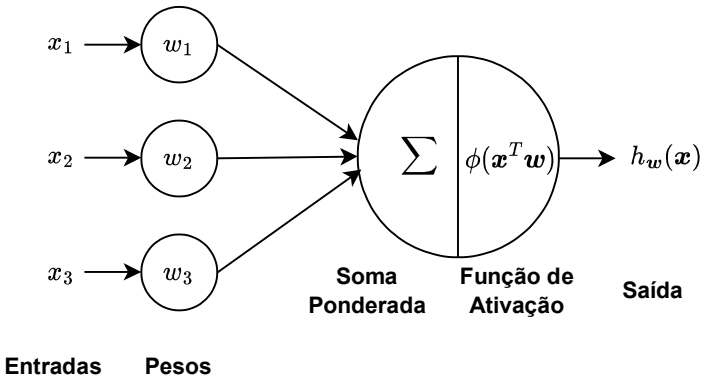


Figura 2.10 Modelo de Neurônio Artificial. Estão representadas as entradas x_i, os pesos w_i, a função de ativação ϕ e a saída, ou hipótese $h_w(x)$ [5].

Um perceptron é composto por uma única camada de neurônios, com cada neurônio conectado a todas as entradas. Quando todos os neurônios de uma camada estão conectados a todos os neurônios de uma camada anterior, esta camada é chamada de *totalmente conectada* (ou *fully connected layer*, ou *dense layer*). As entradas do Perceptron são alimentadas por neurônios de passagem, chamados *neurônios de entrada*. Todos os neurônios de entrada formam a *camada de entrada*. Além disso, um recurso de *bias* é geralmente adicionado ($x_0 = 1$). Normalmente é representado usando um tipo especial de neurônio, chamado *neurônio bias*, que produz 1 o tempo todo. Um Perceptron com duas entradas e três saídas é representado na Figura 2.11. Este Perceptron pode classifi-

car exemplos simultaneamente em três classes binárias diferentes, considerando um vetor onde apenas uma saída é ativada, o que o torna um classificador de múltiplas saídas.

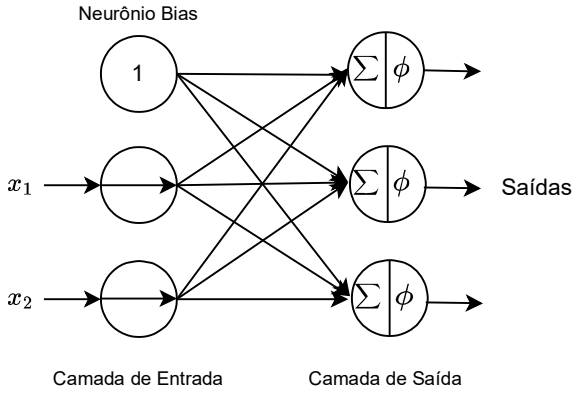


Figura 2.11 Modelo Perceptron, composto por uma única camada (camada de saída) conectada a todos os neurônios de entrada, incluindo o *bias* [5].

A equação 2.17 torna possível calcular as saídas de uma camada de neurônios artificiais para vários exemplos ao mesmo tempo.

$$h_{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{X}) = \phi(\boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}) \tag{2.17}$$

onde $\boldsymbol{X}$ é a matriz dos atributos de entrada contendo todos os exemplos do *dataset*, $\boldsymbol{W}$ a matriz contendo os pesos das conexões, exceto a conexão com o *bias* e $\boldsymbol{b}$ o vetor contendo os pesos das conexões com o *bias*.

A regra de treinamento do perceptron proposta por Rosenblatt é mostrada na equação 2.18. O perceptron é alimentado com um exemplo de treinamento por vez e, para cada exemplo, faz suas previsões. Para cada neurônio de saída que faz uma previsão errada, ele reforça os pesos de conexão das entradas deste neurônio que teriam contribuído para a previsão correta [5].

$$w_{i,j}^{(próximo\,passo)} = w_{i,j} + \eta\,(y_j - \hat{y}_j)\,x_i \tag{2.18}$$

onde $w_{i,j}$ é o peso de conexão entre o i-ésimo neurônio de entrada e o j-ésimo neurônio de saída, x_i é o i-ésimo valor de entrada do exemplo de treinamento atual, $\hat{y}_j$ é a saída do j-ésimo neurônio de saída do exemplo de treinamento atual, y_j é o rótulo, ou alvo, para o exemplo de treinamento atual e, por fim, o η é a taxa de aprendizado.

Perceptrons de Múltiplas Camadas Os perceptrons de múltiplas camadas (*Multilayer Perceptrons* - MLPs, também conhecidos como *deep feedforward networks* ou *feedforward neural networks*) são os modelos mais conhecidos de *deep learning*. O objetivo é aproximar uma função $f^{(i)}(\boldsymbol{x})$. Por exemplo, para um classificador, $y = f^{(i)}(\boldsymbol{x})$ mapeia uma entrada $\boldsymbol{x}$ para uma categoria y. Uma rede MLP define um mapeamento

$y = f^{(i)}(\boldsymbol{x}; \boldsymbol{\theta})$ e aprende o valor dos parâmetros $\boldsymbol{\theta}$ que resultam na melhor função de aproximação [6].

As *redes neurais profundas* são normalmente representadas pela composição de muitas funções diferentes. O modelo está associado a um grafo acíclico direcionado que descreve como as funções são compostas juntas. Por exemplo, pode-se ter três funções $f^{(1)}$, $f^{(2)}$ e $f^{(3)}$ conectadas em uma cadeia, para formar $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$. Estas estruturas em cadeias são as mais comumente usadas em redes neurais artificiais. Nesse caso, $f^{(1)}$ é chamada de *primeira camada* da rede, $f^{(2)}$ é chamada de *segunda camada* e assim por diante. O comprimento total da cadeia de camadas fornece a profundidade do modelo. É dessa terminologia que surge o nome *aprendizado profundo* (*Deep Learning*), ou seja, quando um modelo possui muitas camadas ele pode ser considerado como um modelo de aprendizado profundo.

A camada final é chamada de *camada de saída*. Durante o treinamento da rede neural, dirigimos $f(\boldsymbol{x})$ para corresponder a $f^{(i)}(\boldsymbol{x})$. Os dados de treinamento nos fornecem exemplos aproximados e ruidosos de $f^{(i)}(\boldsymbol{x})$ avaliados em diferentes pontos de treinamento. Cada exemplo x é acompanhado por um rótulo $y \approx f^{(i)}(\boldsymbol{x})$. Os exemplos de treinamento especificam diretamente o que a camada de saída deve fazer em cada ponto x; deve produzir um valor próximo de y. O comportamento das outras camadas não é especificado diretamente pelos dados de treinamento. O algoritmo de aprendizado deve decidir como usar essas camadas para produzir a saída desejada, mas os dados de treinamento não dizem o que cada camada individual deve fazer. Em vez disso, o algoritmo de aprendizagem deve decidir como usar essas camadas para melhor implementar uma aproximação de $f^{(i)}$. Como os dados de treinamento não mostram a saída desejada para cada uma dessas camadas, essas camadas são chamadas de *camadas ocultas* [6].

A Figura 2.12 mostra a arquitetura de um MLP com duas entradas, uma camada oculta com quatro neurônios e uma camada de saída com três neurônios. Cada camada, exceto a camada de saída, possui um neurônio *bias*.

O algoritmo de retropropagação (*backpropagation* - BP) tem sido o principal método para realizar o aprendizado dos parâmetros de um MLP. O desenvolvimento do algoritmo BP na literatura foi bastante gradual e a invenção deste método não é atribuída a nenhum artigo único ou grupo de autores. No entanto, um artigo no final da década de 1980, dos autores David Rumelhart, Geoffrey Hinton e Ronald Williams, chamou a atenção para a importância e significado desse algoritmo. A ideia central do algoritmo BP para o aprendizado de parâmetros, baseado no gradiente descendente (GD), é bastante simples. Basicamente, ele se baseia na regra da cadeia de diferenciação para fazer uma conexão entre a perda calculada na camada de saída e os nós ocultos. Essa conexão ajuda a transmitir a perda final da rede de volta para as camadas anteriores na rede, para que os pesos dessas camada possam ser ajustados proporcionalmente (na direção que reduz a perda) [39].

Uma das variações do GD é o gradiente descendente em mini lotes (*Mini-batch Gradient Descent*). Este é o tipo de GD que tende a convergir mais rápido e utiliza um lote de exemplos do banco de dados a cada iteração. Considerando o gradiente descendente em mini lotes, o algoritmo de retropropagação pode ser resumido da seguinte forma [5]:

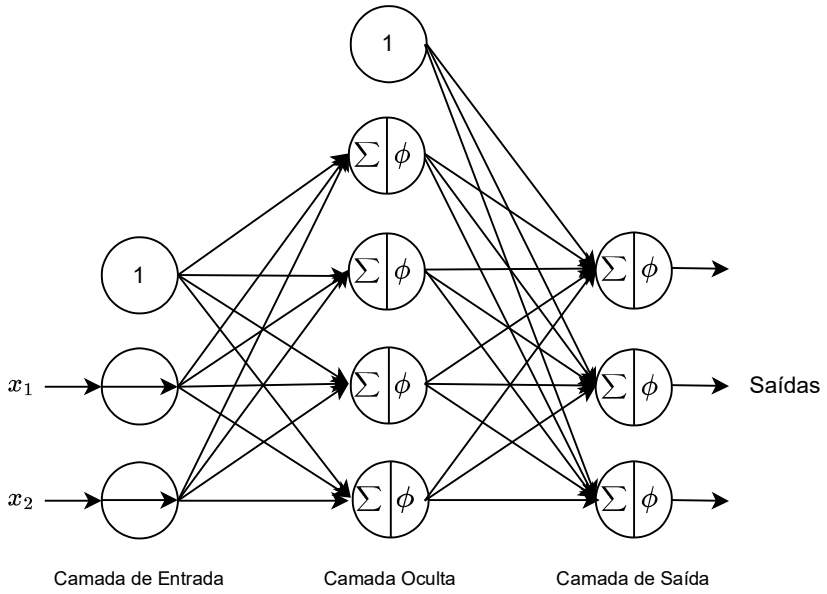


Figura 2.12 *Multilayer* Perceptron com duas camadas, uma camada de saída com três neurônios e uma camada oculta com quatro neurônios mais o *bias*, representado pelo neurônio com valor constante 1 [5].

- Ele trabalha com um *mini-batch* por vez (contendo 32 exemplos, por exemplo) e passa por todo o conjunto de treinamento várias vezes. Cada passagem é chamada de época (*epoch*).
- Cada *mini-batch* é inserido na camada de entrada da rede, que o envia para a primeira camada oculta. O algoritmo então calcula a saída de todos os neurônios nesta camada. O resultado é passado para a próxima camada, e assim por diante, até se obter a saída da camada de saída. Esse é o *passo para frente*: é exatamente como fazer previsões, exceto que todos os resultados intermediários são preservados, pois são necessários para o *passo para trás*.
- Em seguida, o algoritmo mede o erro de saída da rede, ou seja, ele usa uma função custo que compara a saída desejada e a saída real da rede.
- Então, ele calcula o quanto uma variação infinitesimal de cada conexão de saída perturba o critério de desempenho, que contribui para o erro. Isso é feito analiticamente aplicando a regra da cadeia.
- O algoritmo mede então quanto dessas contribuições de erro vieram de cada conexão na camada anterior, novamente usando a regra da cadeia, trabalhando para trás até que o algoritmo alcance a camada de entrada.
- Finalmente, o algoritmo executa uma etapa do gradiente descendente para ajustar todos os pesos de conexão na rede, usando os gradientes de erro que acabou de

calcular.

Para o algoritmo funcionar corretamente é necessário que a função de ativação tenha derivada diferente de zero, como a função sigmoide, exposta na equação 2.19. Isto permite que o gradiente descendente faça progresso a cada iteração.

$$\sigma(t) = \frac{1}{1 + e^{-t}} \tag{2.19}$$

O algoritmo de retropropagação funciona bem com outras funções de ativação, como:

- A função *Rectified Linear Unit*: $ReLU(z) = max(0, z)$.

  A função ReLU é contínua, mas infelizmente não diferenciável em $z = 0$ (a inclinação muda abruptamente, o que pode fazer o gradiente descendente saltar), e sua derivada é 0 para $z < 0$. Na prática, no entanto, funciona muito bem e tem a vantagem de ser computada rapidamente e por isso se tornou o padrão [5].

- A função *Exponential Linear Unit* ELU [40]: $ELU(z) = \begin{cases} \alpha(e^z - 1) & se\ z < 0 \\ z & se\ z \geq 0 \end{cases}$.

  A função ELU se parece muito com a ReLU, com algumas diferenças. Ela assume valores negativos quando $z < 0$, o que permite que a unidade tenha uma saída média mais próxima de 0 e ajuda a aliviar o problema de *vanishing gradients* [5]. O hiperparâmetro α define o valor que a função ELU se aproxima quando z é um grande número negativo. A principal desvantagem da função de ativação ELU é que ela é mais lenta para ser computada do que a função ReLU e suas variantes (devido ao uso da função exponencial). Sua taxa de convergência mais rápida durante o treinamento compensa essa computação lenta, mas ainda assim, no momento do teste, uma rede ELU será mais lenta do que uma rede ReLU [**?**].

Otimizadores Treinar uma rede neural profunda com muitos parâmetros pode ser extremamente lento. Existem algumas maneiras de acelerar o treinamento e alcançar uma melhor solução como uma boa estratégia de inicialização, uma boa função de ativação, normalização por lote, reutilizar partes de uma rede já treinada (*transfer learning*). Outra enorme melhoria de velocidade vem de utilizar um otimizador mais rápido do que o Gradiente Descendente. Alguns algoritmos populares são: *Momentum Optimization*, Nesterov *Accelerated Gradient*, AdaGrad, RMSProp, Adam e Nadam [5].

O algoritmo Adam [41] (*Adaptive moment estimation*) combina as ideias do *Momentum Optimization* e do RMSProp. Assim como o *Momentum Optimization*, ele acompanha uma média exponencialmente decrescente dos passos do gradientes. E assim como o RMSProp, ele acompanha uma média exponencialmente decrescentes dos passos da raiz quadrada dos gradientes.

Dropout O *Dropout* é uma das técnicas mais populares de regularização para redes neurais profundas. Foi proposta por Srivastava *et al.* [42], e provou ser altamente bem

sucedida. Até mesmo redes neurais estado-da-arte conseguem uma melhoria de 1-2% simplesmente adicionando o *dropout* [5].

O *dropout* é um algoritmo bastante simples: a cada etapa de treinamento, cada neurônio (incluindo os neurônios de entrada, exceto os neurônios de saída) tem uma probabilidade p de ser temporariamente "descartado", o que significa que será totalmente ignorado durante esta etapa de treinamento, mas pode estar ativo durante a próxima etapa. O hiperparâmetro p é chamado de taxa de abandono e é normalmente definido entre 10% e 50%. Geralmente entre 40-50% nas redes neurais convolucionais. Após o treinamento, os neurônios não são mais descartados. A Figura 2.13 mostra um exemplo da aplicação do *dropout*, onde alguns neurônios são descartados na camada de entrada e na camada oculta.

Uma forma de entender o poder do *dropout* é perceber que uma rede neural única é gerada a cada passo do treinamento. Como cada neurônio pode estar presente ou ausente, há um total de 2^N redes possíveis (em que N é o número total de neurônios que podem ser descartados). Esse é um número tão grande que é virtualmente impossível que a mesma rede neural seja amostrada duas vezes. Depois de executar 10.000 etapas de treinamento, são treinadas 10.000 redes neurais diferentes (cada uma com apenas um exemplo de treinamento). Essas redes neurais obviamente não são independentes porque compartilham muitos de seus pesos, mas, apesar disso, são todas diferentes. A rede neural resultante pode ser vista como um conjunto de média de todas essas redes neurais menores [5].

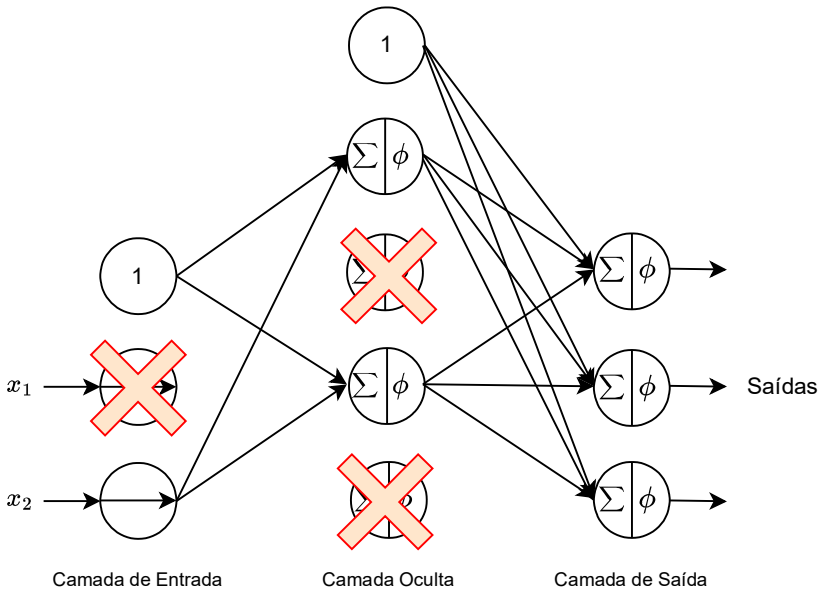


Figura 2.13 Regularização *Dropout*. Cada neurônio tem uma probabilidade p de ser descartado em cada operação realizada [5].

Early Stopping Ao treinar grandes modelos com capacidade de representação suficiente para gerar *overfitting* sobre a tarefa, muitas vezes observa-se que o erro de treina-

mento diminui constantemente ao longo do tempo, mas o erro do conjunto de validação começa a aumentar. Isso significa que pode-se obter um modelo com melhor erro de validação e, portanto, melhor erro de conjunto de teste, retornando à configuração dos parâmetros no ponto do tempo com o menor erro de conjunto de validação. Esta estratégia é conhecida como *early stopping* e é, provavelmente, a forma de regularização mais comum em *deep learning*. Sua popularidade se deve tanto à sua eficácia quanto à sua simplicidade [6].

2.3.5 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNNs, também conhecidas como redes convolucionais) são um tipo de arquitetura de rede neural que possuem operações de convolução e *pooling* em pelo menos uma de suas camadas. Este tipo de arquitetura é especializado em processar dados de imagens de duas e três dimensões, embora seja possível também processar outros tipos de dados, com diferentes dimensões, como áudio, vídeo, etc. Embora a ideia construtiva de uma CNN seja básica, é possível optar por construir a rede de maneiras diferentes, resultando em diferente arquiteturas que são frequentemente motivadas por problemas específicos [39].

Camada Convolucional O bloco construtivo mais importante de uma CNN é a camada convolucional. Neurônios em uma camada convolucional não estão conectados a todos os neurônios da camada anterior, como ocorre em um MLP. Esta arquitetura possibilita que a rede se concentre em pequenas características de baixo nível da camada anterior e permite que as camadas posteriores lidem com atributos de mais alto nível.

A convolução é uma operação matemática que desliza uma função sobre outra e mede a integral da multiplicação ponto a ponto entre elas. No entanto, as camadas convolucionais utilizam correlação cruzada, que são muito similares às convoluções. Esta questão é alertada com mais detalhes em Goodfellow *et al.* [6], Géron [5] e Venkatesan e Li [39]. Na terminologia de rede convolucional, a operação realizada na camada convolucional é representada por um asterisco ($*$). Considerando $z[i,j,k] = x[i,j,k] * w[i,j,k]$, a saída ou mapa de atributos z, é o resultado da operação entre a entrada x e o filtro (ou núcleo, ou *kernel*) w. A equação 2.20 mostra como ocorre a operação na camada convolucional para um neurônio [5].

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k} \qquad onde \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases} \tag{2.20}$$

Na equação 2.20, $z_{i,j,k}$ é a saída do neurônio localizado na linha i, coluna j, no mapa de atributos k, da camada convolucional l. Os *strides*, que é o espaçamento entre os pontos do filtro ao longo da operação, são representados por s_h (vertical) e s_w (horizontal). A altura e largura do campo receptivo, que é a região da entrada mapeada pelo filtro, é respectivamente, f_h e f_w, e $f_{n'}$ é a quantidade de mapa de atributos da camada anterior $(l-1)$. O termo $x_{i',j',k'}$ é a saída do neurônio localizado na camada anterior $(l-1)$, na linha i', coluna j' e no mapa de atributos k'. O *bias* é representado pelo termo b_k. O

termo $w_{u,v,k',k}$ é o peso de conexão entre um neurônio no mapa de atributos k, da camada l, e sua entrada localizada na linha u, coluna v (relativo ao campo receptivo do neurônio), e o mapa de atributos k'.

A Figura 2.14 mostra uma operação de duas dimensões na camada convolucional. Neste caso, a saída foi restringida a apenas posições onde o filtro está inteiramente dentro da imagem, chamada de convolução "válida" [6]. Foram desenhadas caixas com setas para indicar como o elemento superior esquerdo do tensor de saída é formado pela aplicação do filtro à região superior esquerda correspondente da entrada.

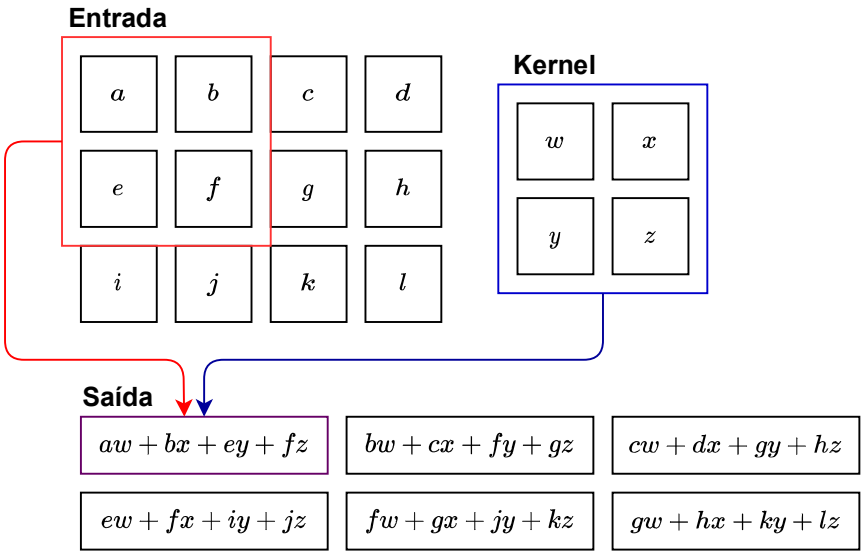


Figura 2.14 Operação Convolucional de um filtro 2D com dimensão (2, 2), com uma entrada com dimensão (3, 4), gerando uma saída com dimensão (3, 4) [6].

Pooling Uma camada típica de uma rede convolucional consiste em três estágios, conforme Figura 2.15. No primeiro estágio, a camada realiza diversas convoluções em paralelo para produzir um conjunto de ativações lineares. No segundo estágio, cada ativação linear é executada por meio de uma função de ativação não linear, como a ReLU por exemplo. Esta etapa é chamada de "estágio de detecção". No terceiro estágio, usa-se uma função *pooling* para modificar ainda mais a saída [6].

Uma função *pooling* substitui a saída da rede em um determinado local por uma estatística resumida das saídas próximas. Por exemplo, a operação *max pooling* resulta na saída máxima em uma região retangular. Outras funções *pooling* populares são a média de uma região retangular, a norma L2 de uma região retangular ou uma média ponderada com base na distância do pixel central.

O objetivo da camada *pooling* é subamostrar a matriz de entrada para reduzir o processamento computacional, o uso de memória e o número de parâmetros, limitando assim o risco de *overfitting* [5]. Goodfellow *et al.* [6] enfatiza que a função *pooling* permite que a camada convolucional seja invariante à pequenas translações da matriz de entrada,

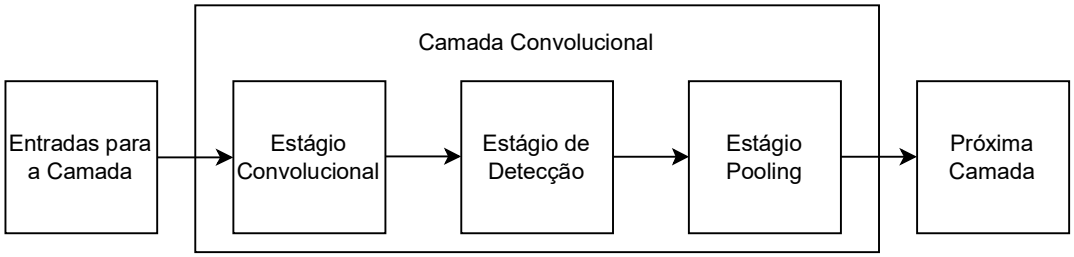


Figura 2.15 Etapas mais comuns de uma Camada Convolucional. As entradas passam pelo estágio convolucional, após isso passam pela função de ativação ou estágio de detecção, e por fim um estágio de *pooling* [6].

uma característica desejável nas aplicações de visão computacional. A Figura 2.16 mostra uma operação *max pooling* em uma entrada 5x5, com um *kernel* 3x3, resultando em uma saída 3x3.

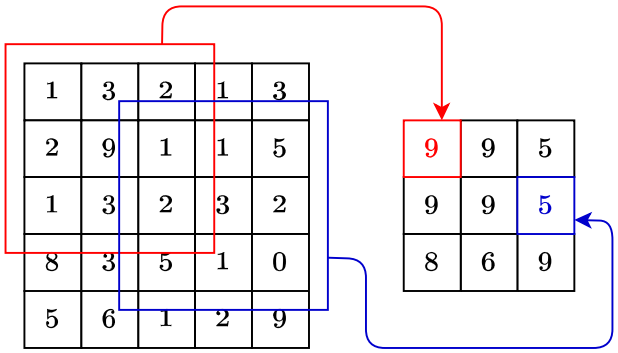


Figura 2.16 Operação *Max Pooling* utilizando filtro com dimensão (3, 3) em uma entrada com dimensão (5, 5), resultando em uma saída com dimensão (3, 3).

Capítulo 3

METODOLOGIA

Neste capítulo será mostrada a metodologia utilizada neste trabalho. Serão apresentados o modelo proposto e os bancos de dados utilizados no treinamento e avaliação do modelo. O método de geração dos escalogramas wavelet é detalhado, assim como a arquitetura e treinamento da rede neural convolucional. Por fim, são apresentados os métodos utilizados para aumento artificial de dados e de previsão por janelas aleatórias.

3.1 MODELO PROPOSTO

O modelo proposto para estimativa de andamento consiste em gerar um escalograma wavelet a partir de um sinal de áudio de uma peça musical com o andamento em BPM pré-definido. O objetivo é realizar um aprendizado supervisionado treinando a Rede Neural Convolucional com as imagens geradas, conforme figura 3.1.

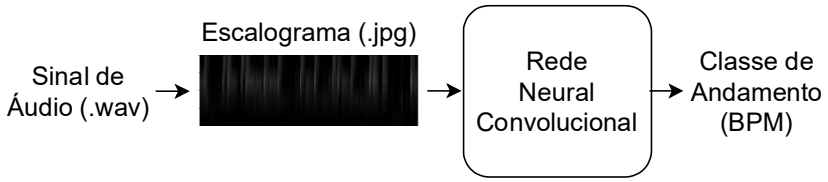


Figura 3.1 Modelo proposto simplificado. O sinal de áudio é transformado em um escalograma wavelet. O escalograma é utilizado para o treinamento da rede neural convolucional que irá detectar o valor do andamento musical em BPM.

Intuitivamente, o problema da estimativa de andamento aparenta ser um problema de regressão para um valor inteiro. Baseado na abordagem de Schreiber e Müller [7], este trabalho optou por tratar como um problema de classificação. A justificativa é que

Tabela 3.1 Quantidade de exemplos dos bancos de dados para treinamento do modelo.

Banco de Dados	Quantidade de Exemplos
LMD Tempo	3611
MTG Tempo	1159
Eball	3826
Total	8596

a distribuição de probabilidade entre diversas classes nos permite julgar o quão confiável é aquela estimativa.

Além disso, o problema de classificação de imagens foi amplamente estudado na última década e diversas arquiteturas de rede foram desenvolvidas especificamente para este tipo de aplicação. Uma boa forma de medir este progresso é observar a taxa de erro em competições como o *ILSVRC Challenge* [43]. Nesta competição a taxa de erro é menor que 2,3%. As imagens são grandes, 256 x 256 pixels, e estão divididas em 1.000 classes [5]. Por isso, esperou-se que a rede consiguisse interpretar o escalograma e classificá-lo em uma classe de andamento.

Observando todos os bancos de dados disponíveis, é possível determinar que os valores de andamento variam entre 23 e 257 BPM. Por isso, podem-se definir classes de andamento variando entre 23 e 257 BPM com passos de 1 BPM, ou seja, 235 classes diferentes. Porém, na seção 3.4.1 serão discutidos os chamados "erros de oitava", em que o modelo pode prever um múltiplo ou submúltiplo do valor de andamento original e ainda assim este ser considerado um acerto. Devido a este tipo de erro, optou-se por concentrar o treinamento no intervalo onde ocorre a maioria dos exemplos, resultando em classes entre 60 e 199 BPM, reduzindo para 140 classes diferentes. Este tipo de intervalo é chamado por alguns autores como *sweet octave* [44], que é o intervalo que possui mais peças musicais do que qualquer outro.

3.2 BANCOS DE DADOS

3.2.1 Bancos de Dados para Treinamento

Para que o modelo seja capaz de generalizar o problema da estimativa de andamento musical, é necessário treiná-lo com bancos de dados que contenham exemplos de diversos estilos musicais e diversas classes de andamento. Foram escolhidos os bancos *Lakh MIDI dataset Tempo* (LMD Tempo) [45] [7], *GiantSteps MTG key dataset Tempo* (MTG Tempo) [46] [7] e *The Extended Ballroom dataset* (EBall) [47]. A quantidade de exemplos em cada um desses bancos está exposta na tabela 3.1.

O LMD é originalmente composto por arquivos *MIDI*. Os arquivos *MIDI* possuem uma anotação de andamento pré-definida, porém não há nenhuma garantia que seja o andamento da música. Por isso, os arquivos foram convertidos em *.wav* e, utilizando o algoritmo de Schreiber e Müller [44], as anotações de andamento foram definidas, formando o LMD Tempo [7]. Este banco de dados possui diversos estilos em sua representação, rock, pop, música eletrônica, etc. Com bateria ou percussão sempre presente.

O MTG *Key* é um banco de dados criado por Faraldo [46] para estimar a tonali-

dade de músicas eletrônicas (edm - *eletronic dance music*). Schreiber *et al.* [7] fizeram manualmente as anotações de andamento resultando no banco de dados MTG Tempo.

O banco de dados *The Extended Ballroom* [47] é uma extensão do banco original Ballroom, com um número maior de exemplos com anotações de andamento. Como o banco Ballroom será usado para avaliação do modelo, as canções que fazem parte dele foram eliminadas do The Extended Ballroom resultando no banco de dados Eball. Os estilos musicais representados são de música de salão como foxtrote, salsa, tango, rumba entre outros. Destaca-se também a presença constante de instrumentos percussivos. Combinando todos os bancos de dados de treinamento, têm-se 8596 exemplos para treinamento do modelo.

Na Figura 3.2 pode-se ver o gráfico violino (*violinplot*) dos bancos de dados utilizados no treinamento do modelo, sendo a quarta coluna a junção de todos os bancos. Naturalmente, o ser humano tende a compor canções em valores de andamento específicos, e isso pode ser observado nos bancos LMD Tempo e MTG Tempo. As medianas dos dois conjuntos estão próximas a 125 BPM. Já o banco Eball possui uma distribuição mais equilibrada ao longo dos andamentos, porém com uma quantidade pequena de exemplos próximo a 150 BPM. A mediana do Eball está entre 100 BPM e 125 BPM. Todos os bancos de dados combinados apresentam uma distribuição mais equilibrada, também com a mediana próxima a 125 BPM, com a maioria dos exemplos concentrados entre 100 BPM e 150 BPM.

A fim de realizar o treinamento do modelo com o *sweet octave* foi necessário revisar as anotações de alguns exemplos do banco de dados. Esta revisão se dá apenas alterando a sua anotação para um valor de múltiplo ou submúltiplo que esteja no intervalo desejado. 453 exemplos tiveram suas anotações revisadas (5,27 %). Com isso, o *sweet octave* escolhido (60 a 199 BPM) possui aproximatamente 95% das peças musicais. A distribuição resultante pode ser vista no *violinplot* da última coluna da Figura 3.2.

3.2.2 Bancos de Dados para Avaliação

Os bancos de dados escolhidos para a avaliação do modelo são amplamente utilizados na literatura. Com isso, será possível comparar os resultados alcançados com outras publicações. Logicamente, exemplos utilizados no treinamento não estarão nos conjuntos de avaliação. Foram utilizados os conjuntos ACM Mirum[48], Ballroom[13], GiantSteps Tempo [49], GTzan [50], Hainsworth[51], ISMIR04[13] e SMC Mirum[52]. Todos possuem um conjunto de obras musicais em arquivos *.wav* com uma anotação de andamento em BPM.

O ACM Mirum representa músicas populares como rock, pop, entre outros. A maioria das músicas possui instrumentos percussivos, porém em alguns casos apenas piano ou violão são utilizados. O Ballroom é composto por música de salão similar ao *The Extended Ballroom*. O GiantSteps Tempo é composto por música eletrônica, assim como o MTG Tempo. O GTzan possui estilos variados como blues, clássico, country, disco, hip hop, jazz, metal, pop, reggae e rock. O Hainsworth também é variado com rock, clássico, disco, entre outros. O ISMIR04 também possui estilos variados como samba, jazz, rock, entre outros. Por fim, o SMC Mirum é exclusivamente clássico, com poucos instrumentos

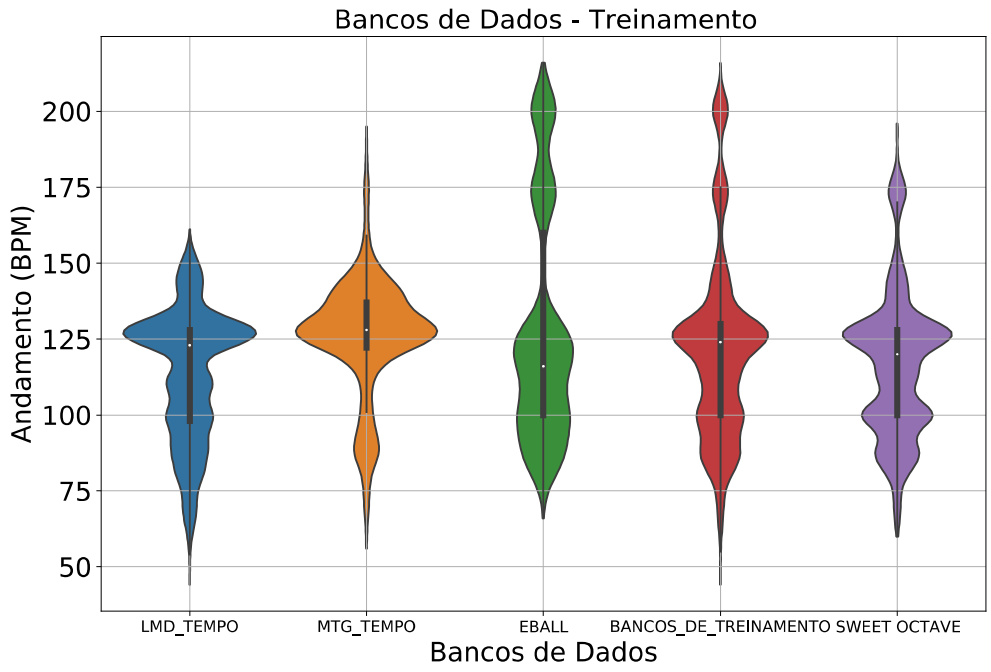


Figura 3.2 *Violinplot* mostrando a distribuição de andamento musical dos Bancos de Dados de Treinamento. A quarta coluna mostra a distribuição de todos os bancos de treinamento juntos e a última coluna apresenta a distribuição do *sweet octave*.

percussivos.

Cada banco de dados possui uma quantidade de exemplos específica, conforme Tabela 3.2. Estes exemplos estão distribuídos nos mais diversos valores de andamento, como mostrado nos *violinplots* da Figura 3.3. Como já mencionado anteriormente, a maioria das obras musicais são compostas em intervalos de andamento específicos. Isto pode ser observado no Ballroom, coluna 2 da Figura 3.3. Ou seja, a distribuição igualitária dos conjuntos não pressupõe que deva existir a mesma quantidade de exemplo em todos os andamentos, pois alguns andamentos ocorrem com pouca frequência em bancos de dados reais. A última coluna da Figura 3.3 mostra a distribuição de andamento musical da junção entre todos os bancos utilizados para avaliação, denominada "Combinados".

É importante enfatizar que os bancos de dados de avaliação não sofreram nenhum tipo de revisão nas anotações de andamento para que a comparação com o estado da arte seja fidedigna.

3.3 REPRESENTAÇÃO DO SINAL DE ÁUDIO COMO IMAGEM

Sempre que são utilizados algoritmos de aprendizado profundo para solucionar problemas que envolvam sinais de áudio, a representação do sinal é um ponto importante a ser definido. O sinal de áudio pode ser representado de diversas formas, independentemente

Tabela 3.2 Quantidade de exemplos dos bancos de dados para avaliação do modelo.

Banco de Dados	Quantidade de Exemplos
ACM Mirum	1410
Ballroom	698
GiantSteps Tempo	660
GTzan	999
Hainsworth	222
ISMIR04	465
SMC	217
Total	4671

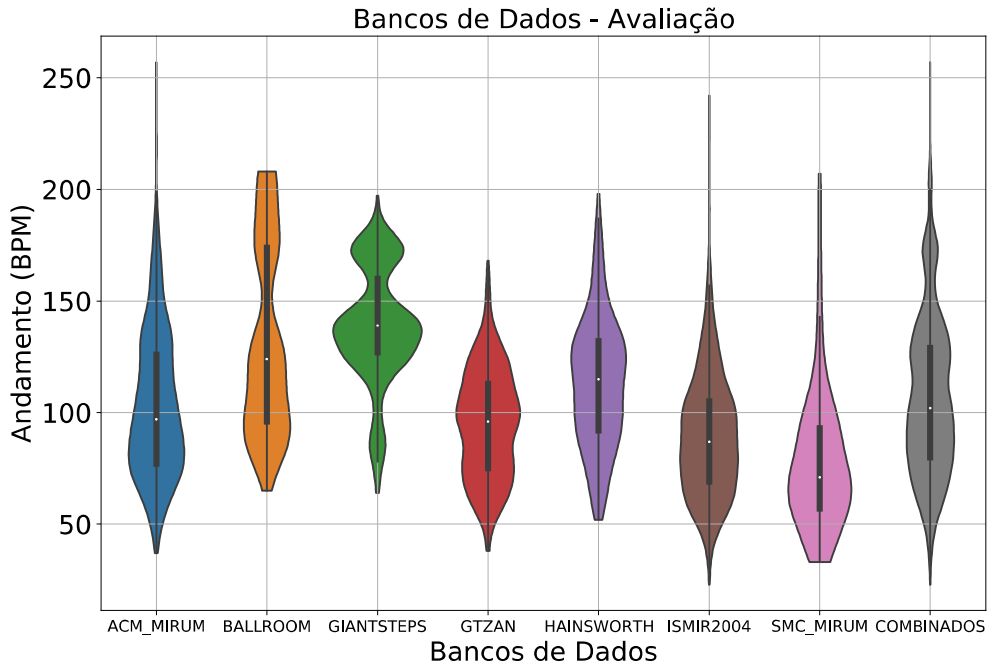


Figura 3.3 *Violinplot* mostrando a distribuição de andamento musical dos Bancos de Dados de Avaliação. A última coluna mostra a distribuição de todos os bancos de avaliação juntos, denominado *Combinados*.

do tipo de problema. Dieleman *et. al.* [53] e Lee *et. al.* [54] utilizaram o próprio sinal de áudio bruto para alimentar a rede neural aritificial. Schreiber e Müller [7] e Wu e Lee [55] utilizaram espectrogramas-mel, a forma mais popular para aplicações de áudio. Esta popularidade ocorre devido à especialidade de arquiteturas de redes convolucionais para problemas de visão computacional, conforme já discutido.

Recentemente, alguns trabalhos começaram a utilizar outras formas de representar os sinais de áudio como imagem, como os escalogramas. Copiaco *et. al.* [56], Ren *et. al.* [57] e Abbasi *et. al.* [58] são exemplos de trabalhos utilizando escalogramas wavelet para

treinar redes neurais convolucionais.

Neste trabalho, para gerar os escalogramas, todos os sinais de áudio foram convertidos para mono e subamostrados para 11.025 Hz, valor suficiente para detectar andamentos acima de 646 BPM [7]. Como o andamento musical não é uma característica instantânea, é necessário que o escalograma represente um espaço de tempo suficiente. As dimensões finais dos escalogramas que serão utilizados para treinar a rede são de 256 pixels no eixo horizontal e 40 pixels no eixo vertical. Por isso, foi escolhido o valor de 11,888 segundos para que o comprimento do vetor fique representado na base 2, otimizando as operações. O vetor resultante do áudio, após a subamostragem, possui 131072 amostras. Desta forma, cada pixel irá representar uma janela de 512 amostras do sinal, valor suficiente para que seja detectado o andamento de todos os exemplos dos bancos de dados.

A partir do vetor do sinal de áudio, pode-se gerar o escalograma wavelet após definir a função wavelet e as escalas a serem utilizadas na CWT. Não há definição de qual a função wavelet é a melhor para gerar os escalogramas e por isso foram realizados experimentos para verificar a partir de qual escalograma a rede interpreta melhor o andamento.

3.3.1 Geração dos Escalogramas

Para realizar o treinamento da rede neural convolucional, todos os exemplos dos bancos de dados, inicialmente em formato *.wav*, são convertidos em imagens com formato *.jpg*. É importante que em um experimento, todos os exemplos sejam convertidos utilizando a mesma técnica. A Figura 3.4 mostra o processo de geração do escalograma wavelet, onde a entrada é o sinal de áudio, que passa por três etapas diferentes, e a saída é o escalograma.

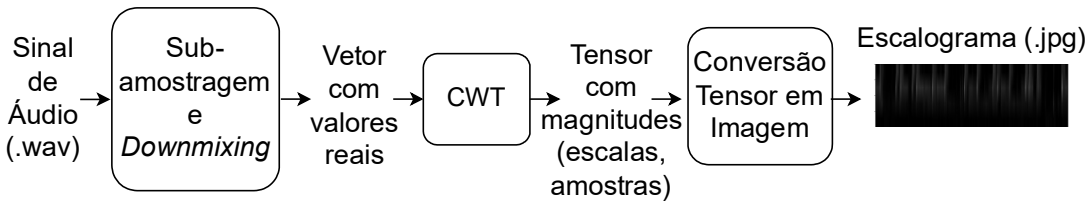


Figura 3.4 Processo de geração do escalograma wavelet. O sinal de áudio passa pelos processos de subamostragem e *downmixing*, aplicação da CWT e por fim a conversão do tensor em imagem, resultando no escalograma.

Cada etapa possui parâmetros que quando modificados influenciam no resultado final do escalograma. Na primeira etapa, os parâmetros para conversão do sinal de áudio em um vetor com valores reais são mantidos fixos, conforme já mencionado anteriormente. São eles: a taxa de amostragem e o comprimento do sinal, com valores 11.025 Hz e 11,888 s respectivamente. Caso o sinal seja *stereo*, ele é convertido em *mono*, processo conhecido como *downmixing*. O único parâmetro que pode ser modificado é o *offset*, ou seja, ponto inicial que o sinal de áudio será convertido.

Realizar a conversão com um valor de *offset* significa desprezar os segundos iniciais da música, o que pode ser um fator determinante para que o modelo generalize melhor o valor

de andamento. Grande parte das músicas possuem um trecho inicial de introdução em um andamento diferente, ou com volume menor. Isso pode ser visualizado nos exemplos da Figura 3.5. Os três escalogramas mostram músicas com pouca informação nos cinco segundos iniciais. Por isso, para todos os exemplos foi considerado um *offset* de 5.0 segundos, e este valor somente foi alterado para realizar um aumento de dados.

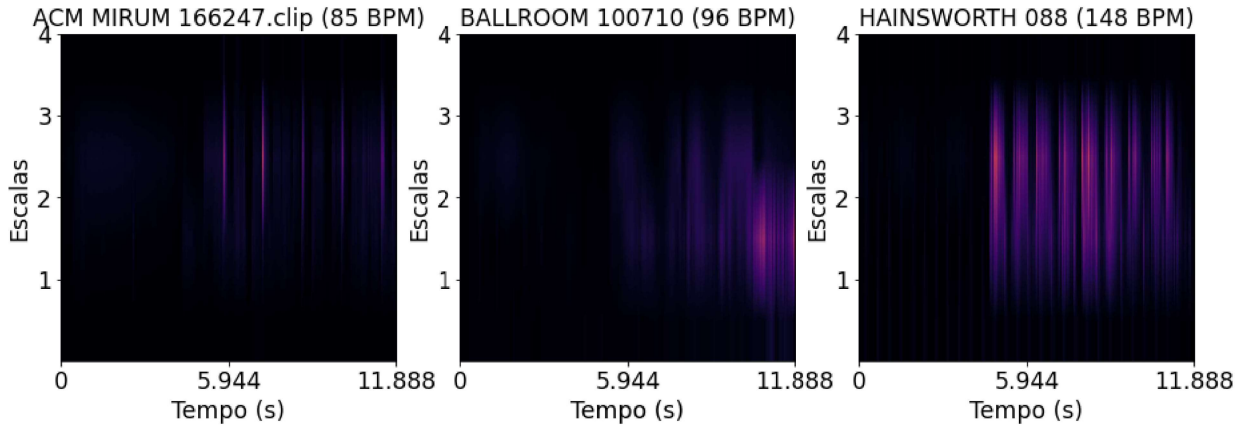


Figura 3.5 Escalogramas com *offset*= 0. São mostrados os segundos iniciais da música, onde geralmente a intensidade do sinal é menor e começa aumentar ao longo do tempo. Este comportamente não é desejável durante o treinamento da CNN.

A partir do vetor com valores reais é possível aplicar a CWT ao sinal. Nesta etapa, alguns parâmetros são importantes para realizar a geração dos escalogramas. O primeiro deles é o tipo de função analisadora a ser utilizada. Para cada tipo de problema um tipo de função wavelet analisadora pode ser mais adequado, realçando informações no sinal que ajudem o modelo a conseguir melhores resultados. Kumar e Kumar [59] aplicaram a CWT para estimar a tonalidade musical de canções polifônicas e detectaram que a wavelet Morlet era mais adequada para o problema do que as wavelets Shannon e Chapéu Mexicano. Azizi *et al.* [60] utilizou apenas a Chapéu Mexicano para o problema da transcrição musical automática.

Ainda não se sabe qual função contínua wavelet se comporta melhor para o problema de estimativa de andamento musical. Por isso, neste trabalho foram investigadas algumas funções wavelets conhecidas na literatura, que são utilizadas nas mais diferentes aplicações, para verificar se existem variações significativas no resultado. Foram escolhidas as wavelets: Chapéu Mexicano, Morlet e Shannon.

O segundo parâmetro para aplicar a CWT são as escalas a a serem utilizadas. Elas podem variar em quantidade e em valores do parâmetro a da equação 2.9. Cada valor escolhido para o parâmetro a refere-se a uma frequência na qual o sinal está sendo observado. Esta frequência de observação também varia a depender da wavelet utilizada e também da taxa de amostragem do sinal. Assim como Fernandes Junior [2], as escalas foram escolhidas buscando um equilíbrio entre as regiões do espectro comumente utilizado para equalização, conforme Tabela 3.3.

A Tabela 3.4 mostra os diferentes parâmetros para geração dos escalograma que serão

Tabela 3.3 Espectro de Equalização

Nome da Faixa	Faixa de Frequências (Hz)
Subgraves	20-60
Baixas	60-250
Médias-baixas	250-2000
Médias-altas	2000-6000
Altas	6000-20000

utilizados para treinar, validar e testar o modelo em cada experimento. Na primeira coluna está o número do experimento e na segunda o rótulo utilizado. As Figuras 3.6, 3.7 e 3.8 mostram escalogramas gerados a partir de um mesmo arquivo *.wav*, considerando os parâmetros de cada experimento.

Para exemplificar as diferenças entre os escalogramas, foram geradas imagens a partir do exemplo ACM MIRUM 1196613.clip, que possui 120 BPM. A Figura 3.6 mostra os escalogramas gerados pela função analisadora wavelet Chapéu Mexicano, que representam os experimentos 1, 2 e 3. O experimento 1 considera as faixas subgraves, baixas e médias-baixas, onde foi observado a maior parte da informação do sinal na maioria dos exemplos, e utiliza quatro escalas. O experimento 2 utiliza 6 escalas buscando observar todo o espectro de equalização. O experimento 3 busca observar as mesmas faixas de frequência do experimento 1, porém aumentando o número de escalas gerando assim um escalograma com maior resolução e mais informação. A estratégia é a mesma para os experimentos seguintes, a Figura 3.7 mostra os escalogramas gerados pelos experimentos 4, 5 e 6, utilizando a função wavelet Morlet. A Figura 3.8 mostra os escalogramas gerados pelos experimentos 7, 8 e 9, utilizando a função wavelet Shannon.

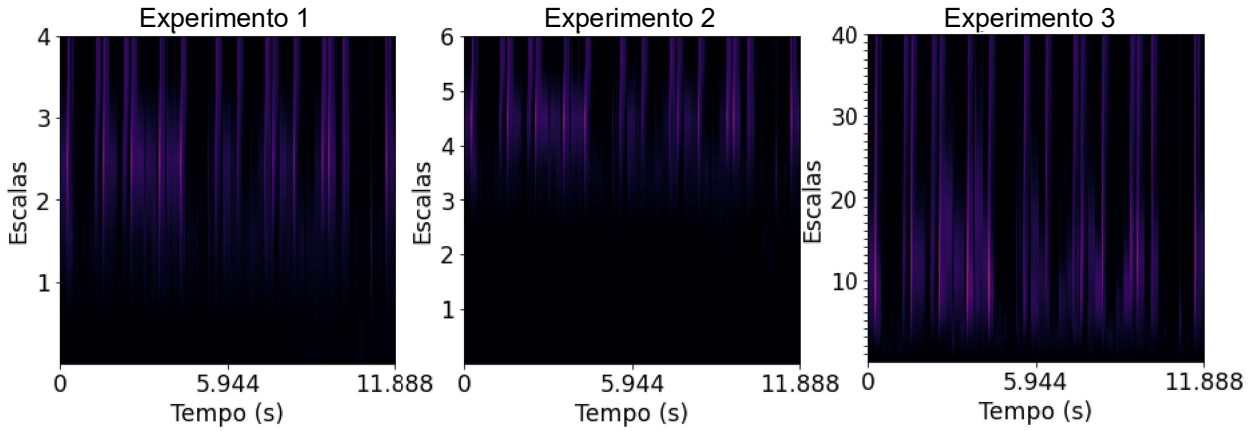


Figura 3.6 Escalogramas aplicando a função analisadora wavelet Chapéu Mexicano. Para o experimento 1 foram utilizados 4 níveis de escalas, para o experimento 2 foram utilizados 6 níveis de escalas e para o experimento 3 foram utilizados 40 níveis de escala.

Após aplicar a CWT em um vetor com n amostras, a matriz resultante possui magnitudes (nº de escalas, n). Esta matriz é então transformada em uma imagem com 256 pixels no eixo horizontal e 40 pixels no eixo vertical utilizando o método de interpolação

Tabela 3.4 Parâmetros de Geração dos Escalogramas

Exp.	**Escalograma**	**Função Analisadora**	**Escalas**	**Frequências de Observação (Hz)**
1	Mexh_4	Chapéu Mexicano	[1.3, 11, 45.5, 130]	[2120, 251, 61, 21]
2	Mexh_6	Chapéu Mexicano	[0.13, 0.45, 1.3, 11, 45.5, 130]	[21202, 6125, 2120, 251, 61, 21]
3	Mexh_40	Chapéu Mexicano	[1.3, ..., 130] (40 escalas)	[2120, ..., 21]
4	Morl_4	Morlet	[4.4, 35.5, 149, 400]	[2036, 252, 60, 22]
5	Morl_6	Morlet	[0.44, 1.49, 4.4, 35.5, 149, 400]	[20358, 6012, 2036, 252, 60, 22]
6	Morl_40	Morlet	[4.4, ..., 400] (40 escalas)	[2036, ..., 22]
7	Shan_4	Shannon	[1.51, 12, 50.5, 149]	[2008, 253, 60, 20]
8	Shan_6	Shannon	[0.15, 0.50, 1.51, 12, 50.5, 149]	[20213, 6064, 2008, 253, 60, 20]
9	Shan_40	Shannon	[1.51, ..., 149] (40 escalas)	[2008, ..., 20]

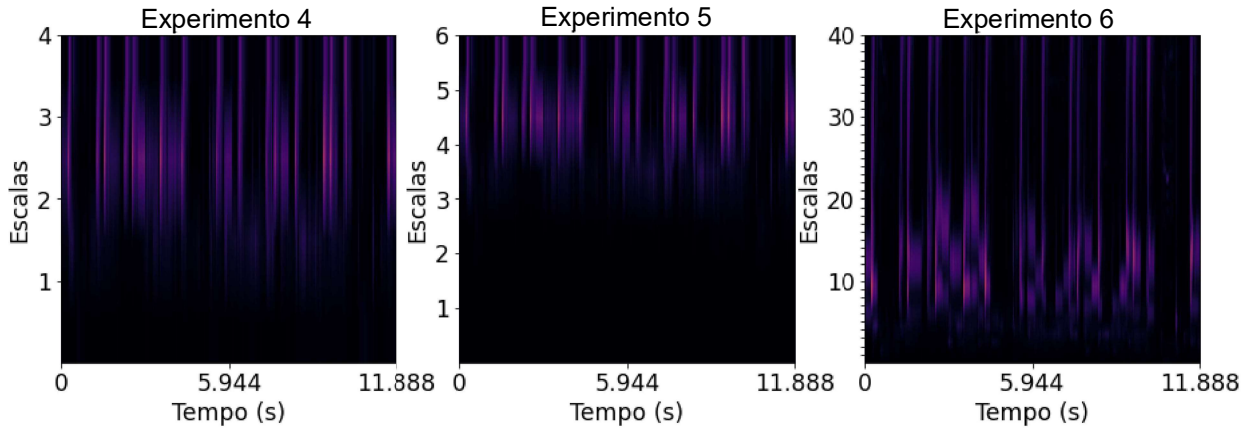


Figura 3.7 Escalogramas aplicando a função analisadora wavelet Morlet. Para o experimento 4 foram utilizados 4 níveis de escalas, para o experimento 5 foram utilizados 6 níveis de escalas e para o experimento 6 foram utilizados 40 níveis de escala.

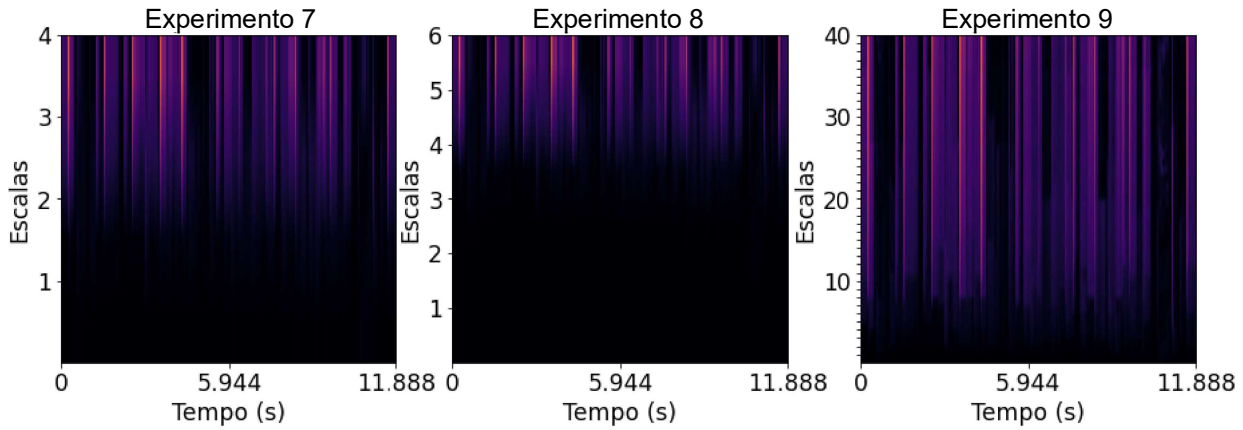


Figura 3.8 Escalogramas aplicando a função analisadora wavelet Shannon. Para o experimento 7 foram utilizados 4 níveis de escalas, para o experimento 8 foram utilizados 6 níveis de escalas e para o experimento 9 foram utilizados 40 níveis de escala.

bilinear e uma paleta de cores em tons de cinza, gerando uma imagem com duas dimensões (40, 256, 1).

3.4 ARQUITETURA DA REDE NEURAL CONVOLUCIONAL

Cada hiperparâmetro escolhido em uma rede neural convolucional pode gerar alterações significativas no resultado final. Como há um grande número de hiperparâmetros, é comum utilizar arquiteturas que já performaram bem anteriormente. Neste trabalho, testes preliminares mostraram que redes clássicas para classificação de imagens, como ResNet50[61], VGG16[62], Xception[63] e InceptionV3 [64], não tiveram um bom desempenho para classificar os escalogramas. Em todos os casos, o modelo se especializava nos exemplos de treinamento e não conseguia generalizar os exemplos de validação e teste.

Isto ocorre porque as redes clássicas de classificação de imagens possuem filtros com duas dimensões. A informação mais importante do escalograma está ao longo do eixo horizontal, que representa os pulsos do sinal de áudio ao longo do tempo. Com esses filtros, as camadas convolucionais acabam relacionando pulsos em diferentes escalas, em momentos diferentes, o que não é desejável para a detecção do andamento.

Por isso, optou-se por utilizar a CNN utilizada por Schreiber e Müller [7]. Esta rede conseguiu bons resultados utilizando espectrogramas-mel, e por isso, espera-se que ela apresente um bom desempenho ao ser treinada com os escalogramas wavelet. O diferencial desta arquitetura é que todas as convoluções são do tipo *"same"*, ou seja, o *padding* é utilizado para que a imagem permaneça com a mesma dimensão, e o *stride* igual a um. Como os filtros possuem dimensão unitária ao longo do eixo vertical, o formato do tensor permanece inalterado ao longo do eixo do tempo, que é o principal para detecção do andamento. Isto pode ser observado na Tabela 3.5 onde o formado do tensor é (y, 256, z) até a camada de achatamento (*flatten*).

A arquitetura CNN pode ser observada na Figura 3.9. A entrada é um escalograma com dimensões (40,256,1). A geração do escalograma será discutida na seção 3.2.2.1. Após a entrada, tem-se em sequência três camadas convolucionais com filtros curtos, com função de ativação ELU. Essas camadas são inspiradas na abordagem tradicional em criar um OSS (*Onset Strengh* - Intensidade do *Onset*) e após isso analisar a sua periodicidade [7].

Após as camadas convolucionais com filtros curtos têm-se os módulos multifiltros, na qual a estrutura pode ser observada na Figura 3.10. Esses módulos tem como objetivo reduzir a dimensionalidade ao longo do eixo das escalas, resumindo a informação e combinando o sinal com uma variedade de filtros que são capazes de detectar dependências temporais [7]. A utilização de camadas convolucionais com filtros de diferentes comprimentos foi inspirada pelos trabalhos de Ferreira *et al.* [65] e Szegedy *et al.* [66].

Para classificar os atributos gerados pelas camadas convolucionais são adicionadas duas camadas densas com 64 neurônios cada, com função de ativação ELU. A camada de saída possui 140 neurônios, representando as 140 classes de andamento e com função de ativação *softmax*. Esta arquitetura resulta em uma CNN com um total 2.840.392 parâmetros, sendo 2.839.750 treináveis e 642 não treináveis.

3.4.1 Treinamento da Rede Neural Convolucional

Para o treinamento, os bancos de dados de treinamento são unificados e aleatoriamente divididos em cinco partes, para utilização da validação cruzada k-fold, com k=5. Destas cinco partes, quatro são usadas para o treinamento, e uma parte dividida entre validação e teste, ou seja, 80% treinamento, 10% validação e 10% teste, conforme Figura 3.11.

Todos os tensores são normalizados antes de iniciar o treinamento do modelo. Seja $\mathbf{X}'$ o tensor $\mathbf{X}$ normalizado, a normalização é realizada conforme equação 3.1.

$$\mathbf{X}' = \frac{\mathbf{X} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{3.1}$$

onde $\boldsymbol{\mu}$ é o vetor contendo a média de cada atributo e $\boldsymbol{\sigma}$ o vetor contendo o desvio padrão

Tabela 3.5 Arquitetura da Rede Neural Convolucional Proposta

Camada	Filtro	Função de Ativação	Formato Tensor de Saída	Número de Parâmetros
Entrada	-	-	(40,256,1)	0
BatchNormalization	-	-	(40,256,1)	4
Conv2D	(1,5,16)	ELU	(40,256,16)	96
BatchNormalization	-	-	(40,256,16)	64
Conv2D	(1,5,16)	ELU	(40,256,16)	1296
BatchNormalization	-	-	(40,256,16)	64
Conv2D	(1,5,16)	-	(40,256,16)	1296
AVGpooling2D	(5,1)	-	(8,256,16)	0
BatchNormalization	-	-	(8,256,16)	64
Conv2D	(1,32,24)	ELU	(8,256,24)	12312
Conv2D	(1,64,24)	ELU	(8,256,24)	24600
Conv2D	(1,96,24)	ELU	(8,256,24)	36888
Conv2D	(1,128,24)	ELU	(8,256,24)	49176
Conv2D	(1,192,24)	ELU	(8,256,24)	73752
Conv2D	(1,256,24)	ELU	(8,256,24)	98328
Concatenate	-	-	(8,256,144)	0
Conv2D	(1,1,36)	-	(8,256,36)	5220
AVGpooling2D	(2,1)	-	(4,256,36)	0
BatchNormalization	-	-	(4,256,36)	144
Conv2D	(1,32,24)	ELU	(4,256,24)	27672
Conv2D	(1,64,24)	ELU	(4,256,24)	55320
Conv2D	(1,96,24)	ELU	(4,256,24)	82968
Conv2D	(1,128,24)	ELU	(4,256,24)	110616
Conv2D	(1,192,24)	ELU	(4,256,24)	165912
Conv2D	(1,256,24)	ELU	(4,256,24)	221208
Concatenate	-	-	(4,256,144)	0
Conv2D	(1,1,36)	-	(4,256,36)	5220
AVGpooling2D	(2,1)	-	(2,256,36)	0
BatchNormalization	-	-	(2,256,36)	144
Conv2D	(1,32,24)	ELU	(2,256,24)	27672
Conv2D	(1,64,24)	ELU	(2,256,24)	55320
Conv2D	(1,96,24)	ELU	(2,256,24)	82968
Conv2D	(1,128,24)	ELU	(2,256,24)	110616
Conv2D	(1,192,24)	ELU	(2,256,24)	165912
Conv2D	(1,256,24)	ELU	(2,256,24)	221208
Concatenate	-	-	(2,256,144)	0
Conv2D	(1,1,36)	-	(2,256,36)	5220
AVGpooling2D	(2,1)	-	(1,256,36)	0
BatchNormalization	-	-	(1,256,36)	144
Conv2D	(1,32,24)	ELU	(1,256,24)	27672
Conv2D	(1,64,24)	ELU	(1,256,24)	55320
Conv2D	(1,96,24)	ELU	(1,256,24)	82968
Conv2D	(1,128,24)	ELU	(1,256,24)	110616
Conv2D	(1,192,24)	ELU	(1,256,24)	165912
Conv2D	(1,256,24)	ELU	(1,256,24)	221208
Concatenate	-	-	(1,256,144)	0
Conv2D	(1,1,36)	-	(1,256,36)	5220
BatchNormalization	-	-	(1,256,36)	144
Flatten	-	-	(9216)	0
Dropout 0.5	-	-	(9216)	0
Densa	-	ELU	(64)	589888
BatchNormalization	-	-	(64)	256
Densa	-	ELU	(64)	4160
BatchNormalization	-	-	(64)	256
Densa (Saída)	-	Softmax	(140)	9100

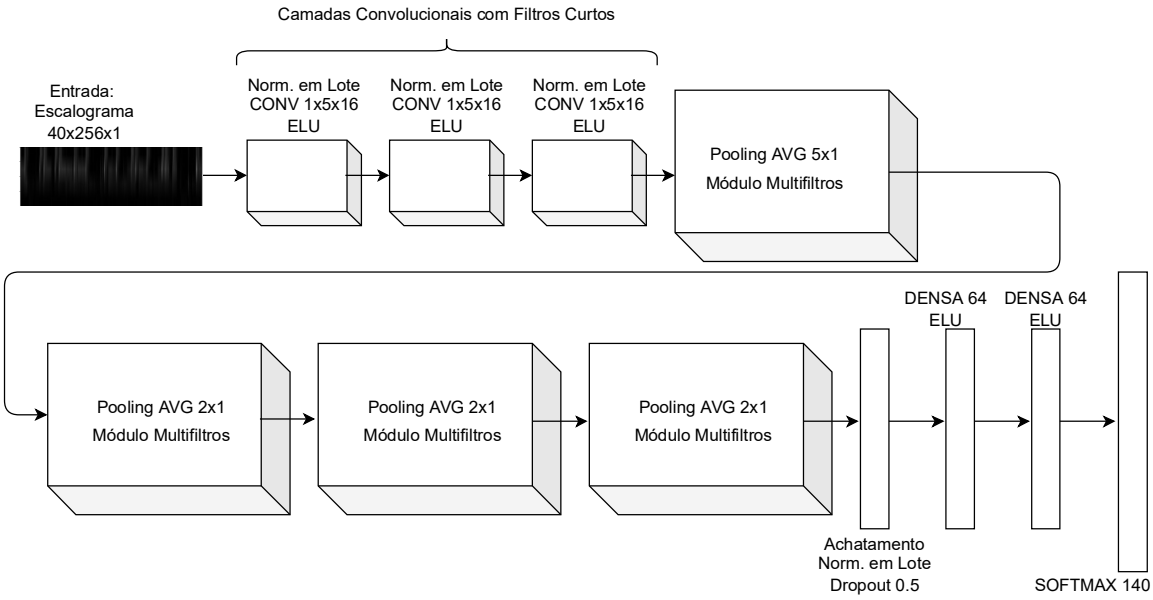


Figura 3.9 Arquitetura Rede Neural Convolucional. Adaptado de [7]. É compostada pela camada de entrada seguida por três camadas convolucionais com filtros curtos, quatro módulos multifiltros e por fim o tensor é achatado e conectado a duas camadas totalmente conectadas. A camada de saída é uma softmax com 140 classes.

de cada atributo. A operação aritmética é realizada através da técnica de *broadcasting* aplicada ao vetor $\boldsymbol{\mu}$ [6].

Após a definição da arquitetura da rede, é necessário definir os hiperparâmetros que irão modificar a forma de como a CNN é treinada. O tipo de perda (ou *loss*) é a *categorical crossentropy*, o otimizador é o Adam [41], as métricas são as Acurácias 0 e 2, foram definidas 50 épocas e o tamanho do lote é 64 exemplos em cada passo do treinamento. Previamente foram testados outros otimizadores, porém o Adam apresentou os melhores resultados.

O desempenho da CNN ao longo do treinamento é acompanhado através dos resultados das métricas escolhidas após cada época. Com isso, é possível traçar o gráfico da acurácia ao longo das épocas e observar o comportamento do modelo. Após finalizar o treinamento, também é gerado o gráfico de valores reais *versus* previsões. Através desse gráfico é possível ver o comportamento das previsões da CNN.

A avaliação do desempenho de modelos de estimativa de andamento possui uma particularidade. É comum que mesmo um humano treinado estime um andamento de uma peça musical com valores múltiplos ou submúltiplos de 2 ou 3 do andamento definido. Isto porque uma peça musical pode ser compatível com diferentes valores absolutos de andamento, a depender da forma dos elementos rítmicos que a compõe. Por isso, será utilizada a forma de avaliação escolhida por Schreiber e Müller [7], Wu *et. al.* [55],

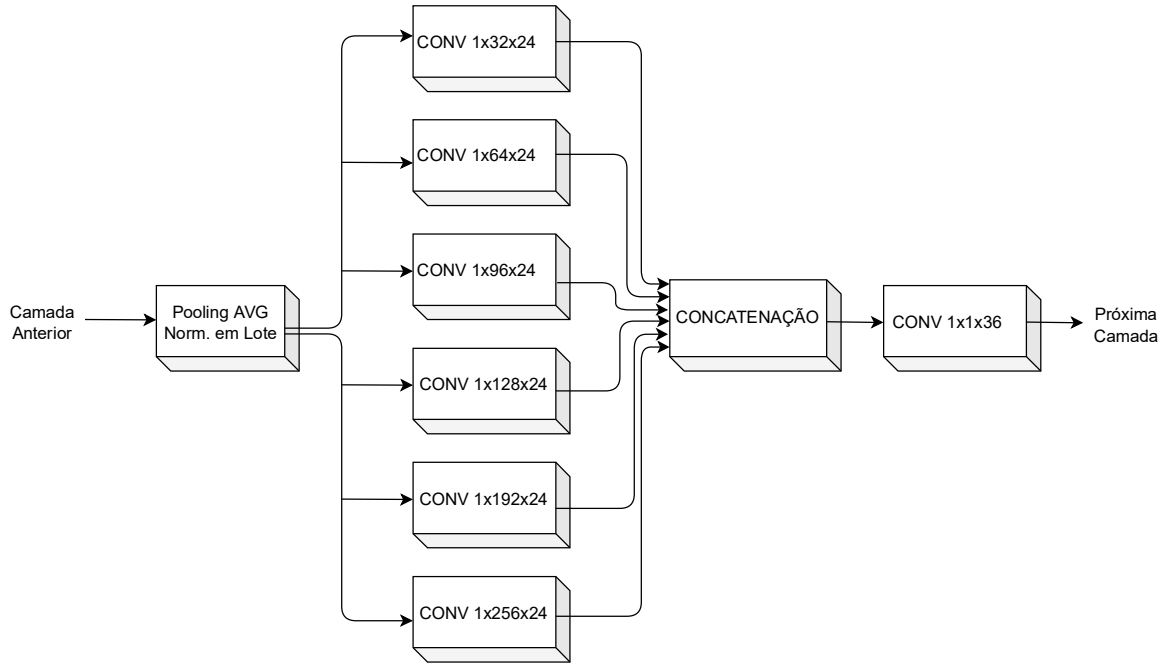


Figura 3.10 Módulo Multifiltros. Sua característica principal são as convoluções em paralelo com diferentes dimensões de filtros. Todas as funções de ativação são ELU. Adaptado de [7].

Fernandes Júnior [2] e também em diversos outros trabalhos.

A Métrica 0, ou Acurácia 0, é definida como a acurácia real do modelo, quando a rede neural convolucional consegue prever exatamente o andamento (Γ) da peça musical, conforme Equação 3.2.

$$\hat{\Gamma} = \Gamma \tag{3.2}$$

A Métrica 1, ou Acurácia 1, considera valores dentro de uma janela de precisão de 4%, conforme a Equação 3.3. Este critério leva em consideração que esta diferença mínima é imperceptível ao ouvido humano, e mesmo pessoas bem treinadas podem prever andamentos com a mesma margem de erro.

$$\hat{\Gamma} = \Gamma \pm 4\% \tag{3.3}$$

Por fim, a Métrica 2, ou Acurácia 2, considera os submúltiplos(1/2 e 1/3) e múltiplos (2 e 3) para o valor real do andamento, dentro de uma janela de precisão de 4%, conforme Equação 3.4.

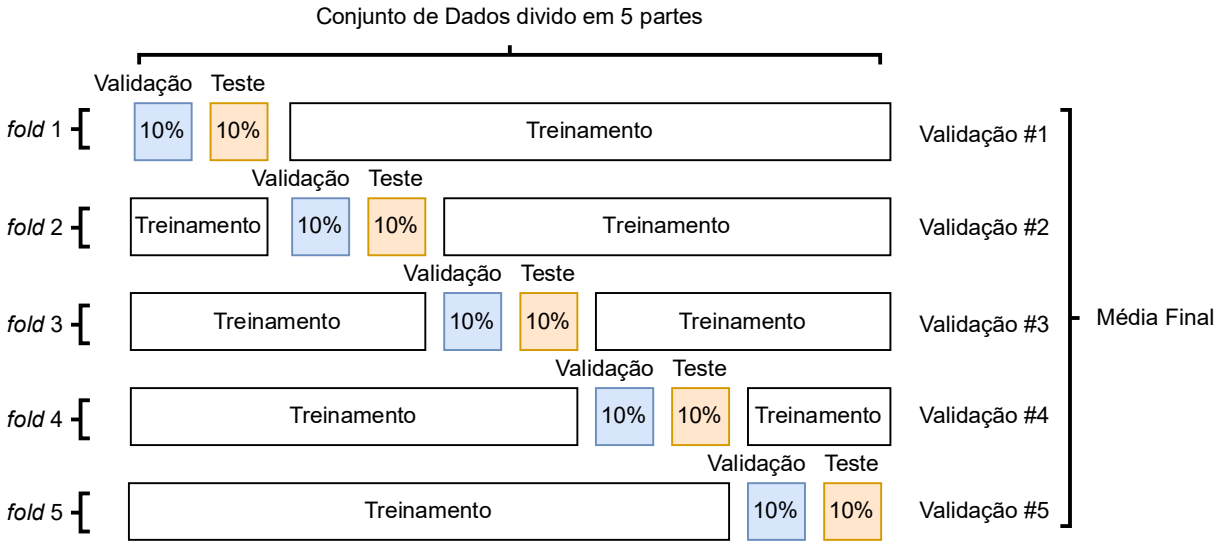


Figura 3.11 Método de validação cruzada *k-fold* dividido em 5 partes, proposto para o trabalho. Diferencia apenas que para cada parte separada para validação, metade dos exemplos são destinados ao teste. O resultado final é a média entre os resultados para cada *fold*.

$$\hat{\Gamma} = (\Gamma \pm 4\%)\, M \qquad M = \frac{1}{3}, \frac{1}{2}, 1, 2, 3 \tag{3.4}$$

onde $\hat{\Gamma}$ é o valor de andamento estimado e Γ o valor de andamento real.

Conforme mencionado, para cada experimento será utilizado o método de validação k-*fold*, com k=5. Ao longo do treinamento serão acompanhadas as acurácias 0 e 2. O desempenho final do modelo é a média dos cinco treinamentos. Com este método de validação cruzada será possível gerar uma maior confiabilidade estatística e definir o melhor resultado para as escolhas envolvendo os parâmetros de geração dos escalogramas wavelet.

Para comparação dos resultados dos experimentos, serão avaliados os indicadores:

- "ACC0 - treinamento": Acurácia 0 do conjunto de treinamento, ao final do treinamento;
- "ACC1 - treinamento": Acurácia 1 do conjunto de treinamento, ao final do treinamento;
- "ACC2 - treinamento": Acurácia 2 do conjunto de treinamento, ao final do treinamento;
- "Valor máximo ACC2 - treinamento": Valor máximo da acurácia 2 do conjunto de treinamento, ao longo do treinamento;
- "ACC0 - validação": Acurácia 0 do conjunto de validação, ao final do treinamento;

- "ACC1 - validação": Acurácia 1 do conjunto de validação, ao final do treinamento;
- "ACC2 - validação": Acurácia 2 do conjunto de validação, ao final do treinamento;
- "Valor máximo ACC2 - validação": Valor máximo da acurácia 2 do conjunto de validação, ao longo do treinamento;
- "ACC0 - teste": Acurácia 0 do conjunto de teste, ao final do treinamento;
- "ACC1 - teste": Acurácia 1 do conjunto de teste, ao final do treinamento;
- "ACC2 - teste": Acurácia 2 do conjunto de teste, ao final do treinamento;

Desta relação de indicadores, os mais importantes são "ACC2 - validação", "ACC2 - teste" e "Valor máximo ACC2 - validação". Com os dois primeiros, conseguimos observar o valores de acurácia 2 para os conjuntos de teste e validação após 50 épocas e verificar se o modelo consegue generalizar e estimar o andamento de músicas que não fizeram parte do treinamento. Já o "Valor máximo ACC2 - validação" é importante porque permite a possibilidade de melhoria dos resultados ao utilizar o *early stopping* e interromper o treinamento antes da 50ª época.

3.5 AUMENTO DE DADOS

O Aumento de Dados (ou *Data Augmentation*) é a técnica para aumentar artificialmente a quantidade de exemplos do conjunto de treinamento, gerando variantes realistas de cada exemplo. Isto reduz o *overfitting* o que a torna uma técnica de regularização [5].

Quando o problema envolve imagens é comum utilizar técnicas como rotação, deslocamento, redimensionamento, entre outras. Quando envolve alterações de iluminação, pode-se utilizar técnicas de variação de contraste ou variações de cores. Porém estas técnicas não se aplicam ao escalograma, pois ele deve ser gerado de um arquivo de áudio. Aplicar alguma destas técnicas iria distorcer a informação contida no escalograma.

Para realizar o aumento de dados com arquivos de áudio, pode-se utilizar técnicas de mudanças de frequências, mudança de velocidade, inserção de ruído, porém algumas delas não fazem sentido para o problema da estimativa de andamento musical. Por isso, inspirado pelo trabalho de Schreiber e Müller [7], optou-se por fazer compressões e expansões do escalograma ao longo do eixo horizontal, mantendo o eixo vertical sem alteração e ajustando o valor de anotação do andamento após a modificação. Ao expandir ou comprimir um escalograma, a velocidade de execução está variando proporcionalmente. Expandindo-o, a música fica mais lenta e o andamento diminui, enquanto que ao comprimir o escalograma, a música fica mais rápida e o andamento aumenta.

Este aumento de dados foi implementado de forma online. Foi definido um hiperparâmetro DA onde pode ser variado de 0% a 100%. Este valor é a probabilidade do exemplo sofrer uma modificação antes de ser utilizado no treinamento da rede. Se $DA = 30\%$, cada exemplo utilizado no treinamento tem 30% de chance de sofrer uma modificação e ter o seu valor de andamento alterado. A cada época novos exemplos podem ser alterados expondo a rede a uma quantidade muito maior de escalogramas.

O método de compressão e expansão está exemplificado nos quatro escalogramas da Figura 3.12. Inicialmente é gerado um escalograma de toda a música, conforme Figura 3.12a utilizando o exemplo ACM MIRUM 169108.clip (124 BPM). Foi definido um fator de alteração (F_a) que é o valor utilizado para comprimir ou expandir o escalograma. O fator de alteração é escolhido aleatoriamente em um grupo de valores pré-definidos $F_a \in \{0.8, 0.85, 0.9, 0.95, 1, 1.05, 1.1, 1.15, 1.2\}$. Se $F_a = 1$, o escalograma permanece com magnitude horizontal inalterada e uma janela de 256 pixels por 40 pixels, com *offset* aleatório, é selecionada. O escalograma da Figura 3.12b mostra esta janela selecionada.

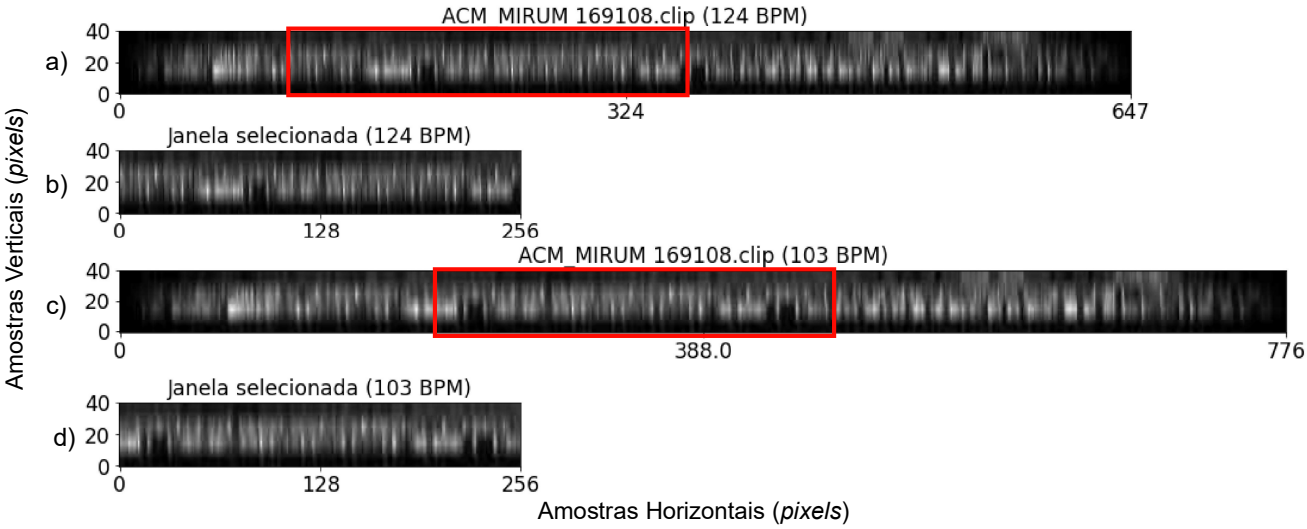


Figura 3.12 Escalogramas do exemplo ACM MIRUM 169108.clip (124 BPM). A figura (a) mostra a música completa e uma janela escolhida aleatoriamente para o treinamento da rede. A figura (b) mostra o detalhe da janela escolhida no escalograma de 124 BPM. A figura (c) mostra a música completa expandida com $F_a = 1.2$ e uma janela escolhida aleatoriamente. A figura (d) mostra o detalhe da janela escolhida no escalograma de 103 BPM.

Para o caso de $F_a = 1.2$, toda a música sofre uma expansão, conforme o escalograma da Figura 3.12c. A classe do exemplo ACM MIRUM 169108.clip passa a ser 103 BPM. Da mesma forma, uma janela de 256 pixels por 40 pixels, com *offset* aleatório, é selecionada conforme apresentado na Figura 3.12d. Com o objetivo de observar os impactos desta técnica de aumento de dados foram realizados experimentos variando o valor de $DA \in \{0, 25, 50, 75, 100\}$.

3.6 PREVISÃO POR JANELAS ALEATÓRIAS

Ao estimar um valor de andamento em uma janela com 256 pixels horizontais, o modelo está prevendo um valor de andamento apenas para aquele intervalo de tempo específico, de 11.888 s. A maioria das músicas dos bancos de dados utilizados possuem um andamento constante ao logo de todo o sinal de áudio, porém em alguns momentos podem ocorrer variações. Caso a janela se posicione exatamente em uma destas variações, a previsão pode ser errada.

Para estimar o andamento global da música e evitar erros pelo posicionamento do local observado, utilizou-se uma estratégia de janelas aleatórias ao longo da música. Esta estratégia consiste em fazer previsões de 30 janelas escolhidas aleatoriamente em uma peça musical. A Figura 3.13 mostra o exemplo GTZAN GENRES metal.00034 (80 BPM). Esta música possui uma variação significativa na metade do escalograma. Foi exemplificado a escolha de 3 janelas, variando o *offset*, mostrando que a janela escolhida pode ser qualquer região do escalograma. Após a escolha aleatória das 30 janelas e todas as previsões, o modelo escolhe o valor de andamento com maior ocorrência e o considera como a estimativa de andamento global.

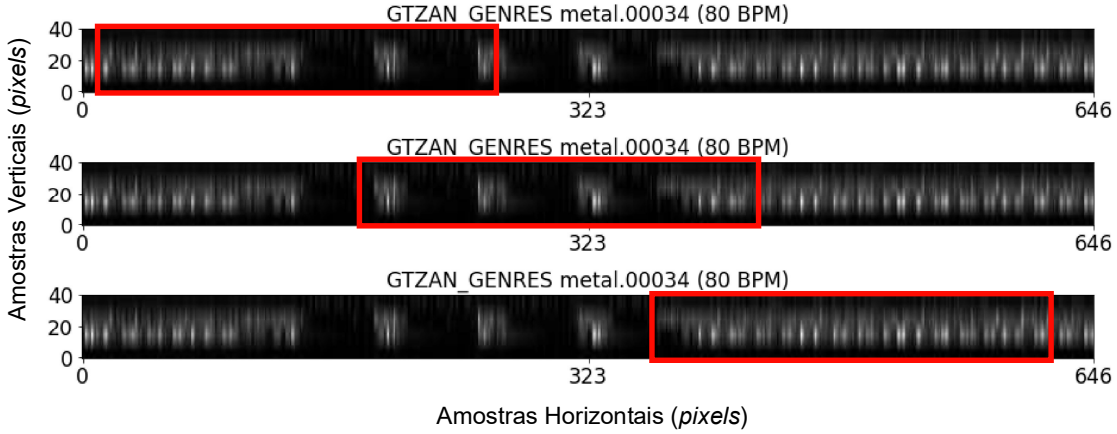


Figura 3.13 Janelas escolhidas aleatoriamente no exemplo GTZAN GENRES metal.00034 (80 BPM). O modelo faz uma previsão para cada janela e escolhe o valor de andamento com mais ocorrências.

Após a definição do melhor modelo, a partir dos experimentos de variação dos parâmetros de geração dos escalogramas e também da variação do parâmetro de aumento de dados (DA), foi realizado um novo treinamento com aplicação do *early stopping* com o objetivo de interromper o treinamento com um valor de acurácia 2 próximo ao valor máximo atingido pelo modelo ao longo de 50 épocas. A partir daí, foi realizada a previsão por janelas aleatórias nos bancos de dados de avaliação e os resultados foram comparados com o estado da arte.

Capítulo

4

RESULTADOS

Neste capítulo serão apresentados os resultados obtidos neste trabalho. Inicialmente, nas seções 4.1, 4.2 e 4.3 são mostrados os resultados dos experimentos realizados com as funções wavelet analisadoras Chapéu Mexicano, Morlet e Shannon, respectivamente. Os resultados são apresentados de forma padronizada em cada uma dessas seções. Os resultados são comparados e discutidos na seção 4.4, para se obter o método de geração do escalograma que melhor se adequou ao problema da estimativa de andamento. Na seção 4.5 foram apresentados os resultados após a aplicação da técnica de aumento de dados. Por fim, os resultados da avaliação do modelo são apresentados na seção 4.6 que são comparados ao estado da arte.

4.1 EXPERIMENTOS COM WAVELET CHAPÉU MEXICANO

4.1.1 Experimento 1 - Mexh_4

A Tabela 4.1 apresenta os resultados do primeiro experimento, utilizando a wavelet Chapéu Mexicano e quatro níveis de escala. É possível observar que, como esperado, os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 50,8%, a acurácia 1 de 85,7% e a acurácia 2 de 89%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,8%. O modelo k1 atingiu um valor máximo 91,7%. Para o conjunto de teste a acurácia 0 foi de 52,7%, a acurácia 1 de 85,0% e a acurácia 2 de 88,3%.

Na Figura 4.1 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de validação fica estagnado. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas para melhor entendimento.

Tabela 4.1 Resultados do Treinamento (k-*fold*) - Mexh_4 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	94,2	94,6	93,7	93,9	94,3	94,2 ± 0,3
ACC1 - treinamento	99,6	99,7	99,6	99,7	99,8	99,7 ± 0,1
ACC2 - treinamento	99,7	99,8	99,8	99,7	99,9	99,8 ± 0,1
Valor máximo ACC2 - treinamento	99,9	99,9	99,8	99,9	99,9	99,9 ± 0,0
ACC0 - validação	53,0	49,7	49,8	52,9	48,9	50,8 ± 1,8
ACC1 - validação	88,1	86,4	84,4	84,9	84,9	85,7 ± 1,4
ACC2 - validação	90,8	89,6	88,3	88,0	88,3	89,0 ± 1,1
Valor máximo ACC2 - validação	91,7	89,9	90,2	91,6	90,5	90,8 ± 0,7
ACC0 - teste	52,3	54,4	52,4	52,9	51,4	52,7 ± 1,0
ACC1 - teste	84,7	87,0	85,8	85,3	82,2	85,0 ± 1,6
ACC2 - teste	87,9	89,0	89,6	88,9	86,2	88,3 ± 1,2

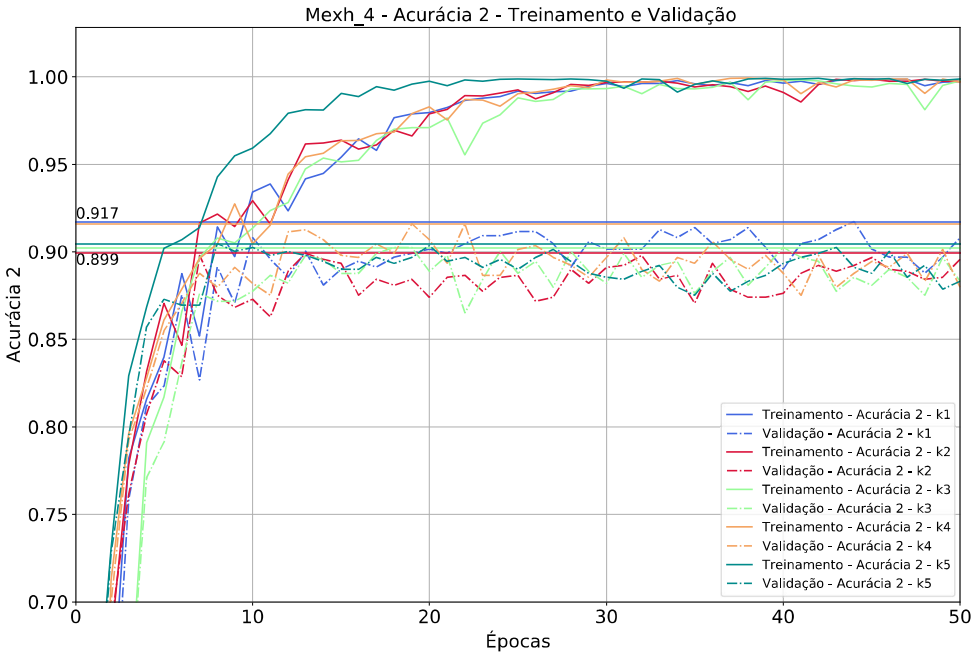


Figura 4.1 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 1 (mexh_4). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

Observando os resultados da Tabela 4.1 e da Figura 4.1 pode-se assumir que o modelo k1 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.2 mostra as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

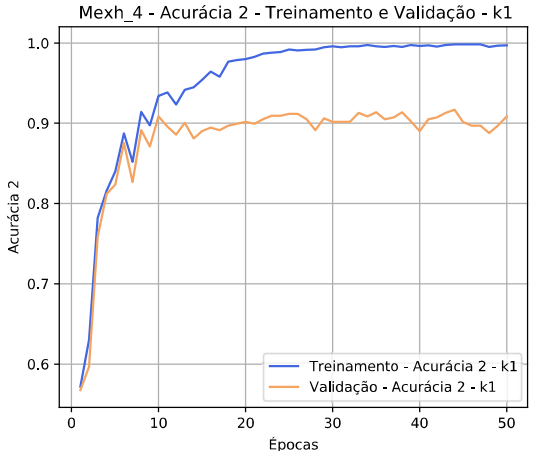


Figura 4.2 Melhor modelo do Experimento 1 (mexh_4), k=1. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

4.1.2 Experimento 2 - Mexh_6

Os resultados do experimento 2, utilizando a wavelet Chapéu Mexicano e com seis níveis de escala, são apresentados de maneira similar ao experimento anterior. Este padrão irá permanecer em todos os experimentos.

A Tabela 4.2 apresenta os resultados deste segundo experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 49,3%, a acurácia 1 de 85,4% e a acurácia 2 de 88,6%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,8%. O modelo k5 atingiu um valor máximo 91,7%. Para o conjunto de teste a acurácia 0 foi de 51,6%, a acurácia 1 de 84,4% e a acurácia 2 de 87,5%.

Tabela 4.2 Resultados do Treinamento (k-*fold*) - Mexh_6 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	93,7	93,3	92,6	93,5	92,8	93,2 ± 0,4
ACC1 - treinamento	99,4	99,5	99,2	99,6	99,0	99,4 ± 0,2
ACC2 - treinamento	99,4	99,6	99,3	99,7	99,1	99,4 ± 0,2
Valor máximo ACC2 - treinamento	99,5	99,6	99,8	99,8	99,6	99,7 ± 0,1
ACC0 - validação	50,3	49,0	48,4	51,2	47,5	49,3 ± 1,3
ACC1 - validação	85,8	86,3	84,6	85,7	84,8	85,4 ± 0,6
ACC2 - validação	89,7	89,1	87,3	88,0	88,8	88,6 ± 0,8
Valor máximo ACC2 - validação	91,1	90,6	90,1	90,7	91,7	90,8 ± 0,5
ACC0 - teste	54,7	50,7	51,4	52,3	49,1	51,6 ± 1,9
ACC1 - teste	84,7	85,1	85,7	84,5	81,7	84,4 ± 1,4
ACC2 - teste	88,0	88,2	88,2	88,7	84,5	87,5 ± 1,5

Na Figura 4.3 as curvas de acurácia 2, em percentual, de treinamento e validação

são apresentadas para todos os k modelos treinados, para o experimento 2. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de validação fica estagnado. Destaca-se uma variação abrupta que o modelo k2 obteve antes da 10ª época e após a 40ª época. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas.

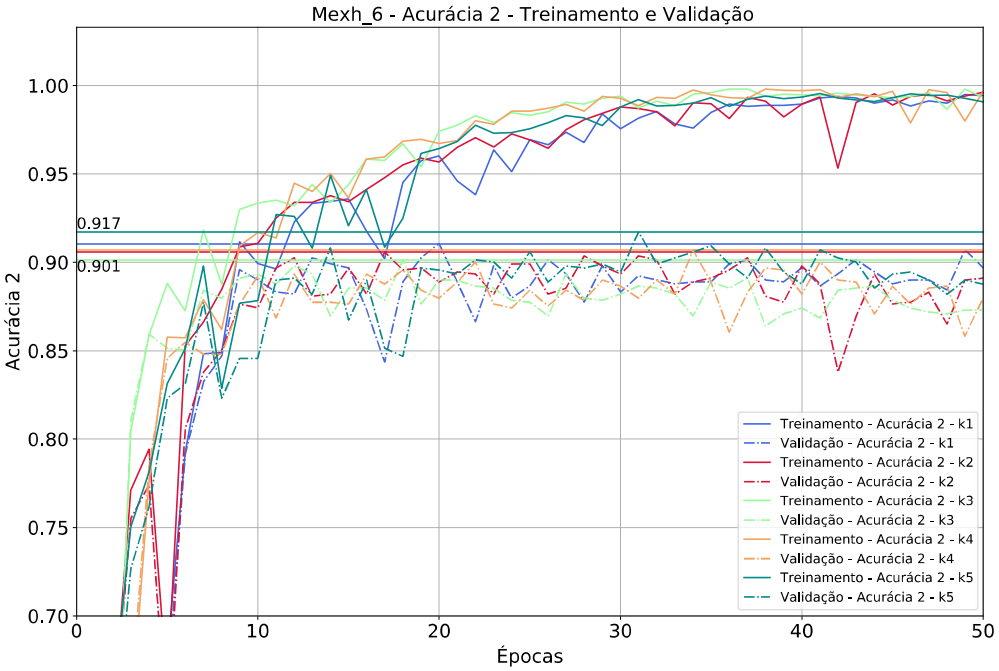


Figura 4.3 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 2 (mexh_6). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

Observando os resultados da Tabela 4.2 e da Figura 4.3 pode-se assumir que o modelo k5 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.4 mostra as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

4.1.3 Experimento 3 - Mexh_40

Os resultados do experimento 3, utilizando a wavelet Chapéu Mexicano e com quarenta níveis de escala, são apresentados de maneira similar ao experimento anterior. A Tabela 4.3 apresenta os resultados deste terceiro experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 50,0%, a acurácia 1 de

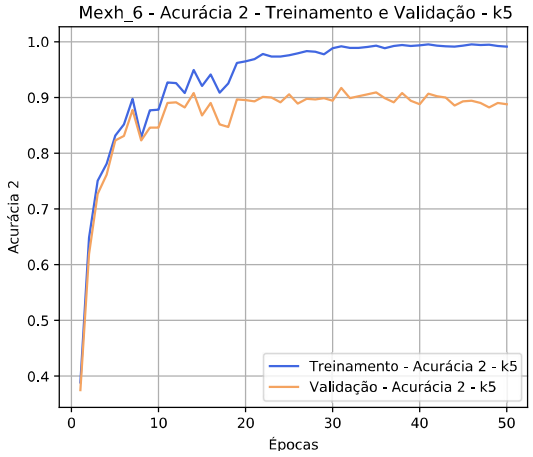


Figura 4.4 Melhor modelo do Experimento 2 (mexh_6), k=5. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

84,9% e a acurácia 2 de 88,0%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,3%. O modelo k5 atingiu um valor máximo 91,4%. Para o conjunto de teste a acurácia 0 foi de 51,6%, a acurácia 1 de 84,9% e a acurácia 2 de 88,3%.

Tabela 4.3 Resultados do Treinamento (k-*fold*) - Mexh_40 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	93,6	93,4	92,4	94,3	93,6	93,5 ± 0,6
ACC1 - treinamento	99,4	99,5	99,1	99,8	99,5	99,5 ± 0,2
ACC2 - treinamento	99,6	99,6	99,5	99,9	99,6	99,6 ± 0,1
Valor máximo ACC2 - treinamento	99,8	99,6	99,7	99,9	99,6	99,7 ± 0,1
ACC0 - validação	48,8	50,2	49,7	52,6	48,9	50,0 ± 1,4
ACC1 - validação	81,9	85,4	86,3	86,1	84,8	84,9 ± 1,6
ACC2 - validação	86,3	88,7	89,3	88,3	87,4	88,0 ± 1,1
Valor máximo ACC2 - validação	89,9	89,5	90,0	90,7	91,4	90,3 ± 0,7
ACC0 - teste	52,2	52,3	51,6	51,7	50,0	51,6 ± 0,8
ACC1 - teste	84,4	85,3	85,6	84,8	84,6	84,9 ± 0,4
ACC2 - teste	87,5	89,0	88,1	88,7	88,1	88,3 ± 0,5

Na Figura 4.5 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados, para o experimento 3. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de validação fica estagnado. Destaca-se uma variação abrupta que o modelo k2 obteve entre a 10ª época e a 20ª época. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas.

Observando os resultados da Tabela 4.3 e da Figura 4.5 pode-se assumir que o modelo k5 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.6 mostra as

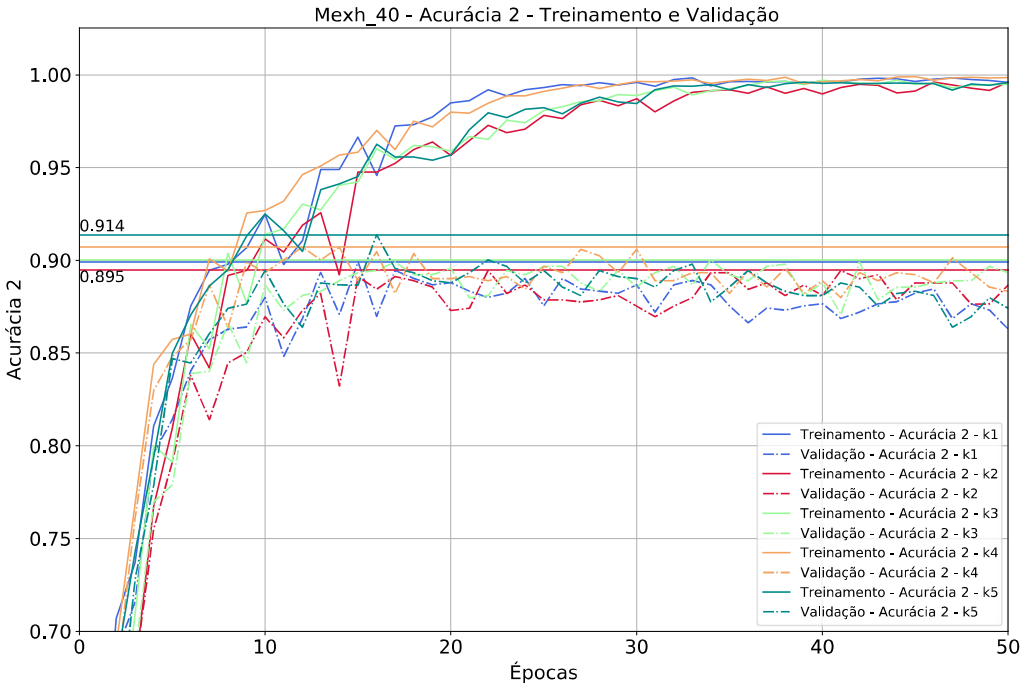


Figura 4.5 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 3 (mexh_40). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

curvas de acurácia 2 de treinamento e validação apenas para este modelo.

4.2 EXPERIMENTOS COM WAVELET MORLET

4.2.1 Experimento 4 - Morl_4

Os resultados do experimento 4, utilizando a wavelet Morlet e com quatro níveis de escala, são apresentados de maneira similar aos experimentos anteriores. A Tabela 4.4 apresenta os resultados deste quarto experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 50,2%, a acurácia 1 de 84,8% e a acurácia 2 de 88,3%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,8%. O modelo k4 atingiu um valor máximo 91,4%. Para o conjunto de teste a acurácia 0 foi de 51,4%, a acurácia 1 de 84,3% e a acurácia 2 de 87,8%.

Na Figura 4.7 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados, para o experimento 4. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10^{a} época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de

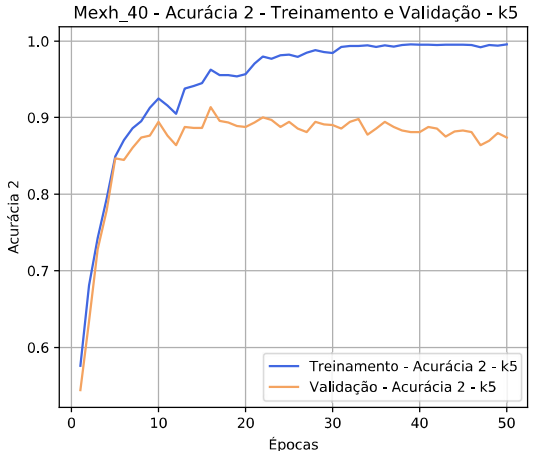


Figura 4.6 Melhor modelo do Experimento 3 (mexh_40), k=5. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

Tabela 4.4 Resultados do Treinamento (k-*fold*) - Morl_4 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	94,1	94,3	93,9	92,8	94,1	93,8 ± 0,5
ACC1 - treinamento	99,6	99,7	99,5	99,2	99,7	99,5 ± 0,2
ACC2 - treinamento	99,7	99,7	99,7	99,3	99,8	99,6 ± 0,2
Valor máximo ACC2 - treinamento	99,8	99,9	99,9	99,9	99,8	99,9 ± 0,1
ACC0 - validação	52,0	48,2	50,5	52,4	48,2	50,2 ± 1,8
ACC1 - validação	84,4	85,1	85,1	84,5	84,9	84,8 ± 0,3
ACC2 - validação	87,9	88,7	88,5	87,4	88,8	88,3 ± 0,5
Valor máximo ACC2 - validação	90,0	90,2	90,8	91,4	91,3	90,8 ± 0,5
ACC0 - teste	53,2	51,7	51,5	50,3	50,2	51,4 ± 1,1
ACC1 - teste	84,4	84,8	85,4	82,0	84,9	84,3 ± 1,2
ACC2 - teste	87,2	88,0	88,4	86,8	88,4	87,8 ± 0,6

validação fica estagnado. Destaca-se uma variação abrupta que o modelo k5 obteve entre a 40ª época e a 50ª época. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas.

Observando os resultados da Tabela 4.4 e da Figura 4.7 pode-se assumir que o modelo k4 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.8 mostra as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

4.2.2 Experimento 5 - Morl_6

Os resultados do experimento 5, utilizando a wavelet Morlet e com seis níveis de escala, são apresentados de maneira similar aos experimentos anteriores. A Tabela 4.5 apresenta os resultados deste quinto experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão

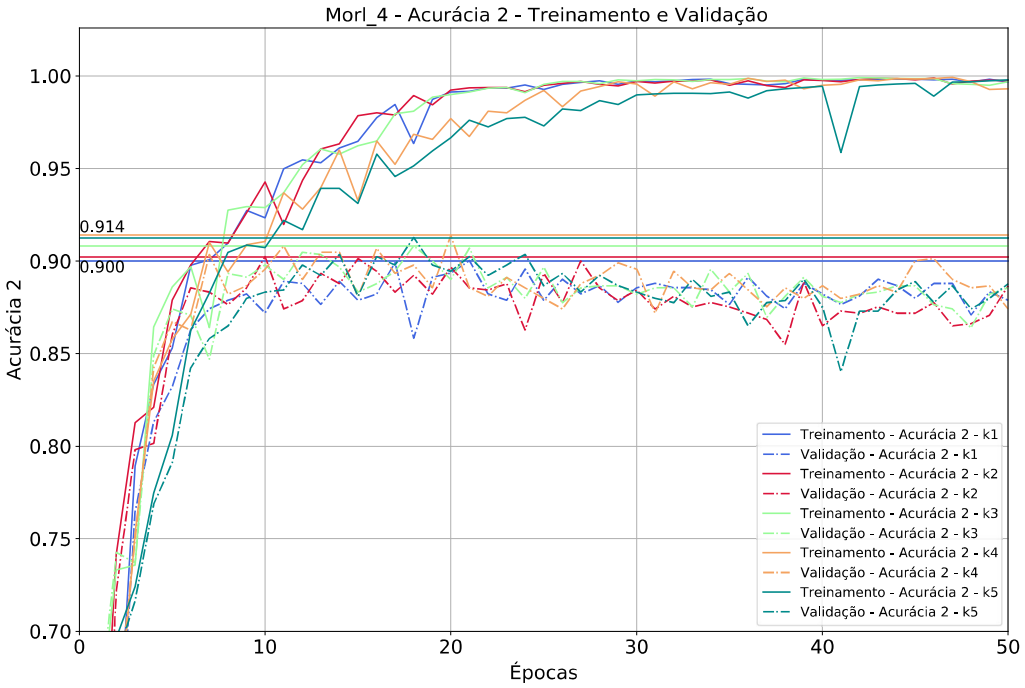


Figura 4.7 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 4 (morl_4). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

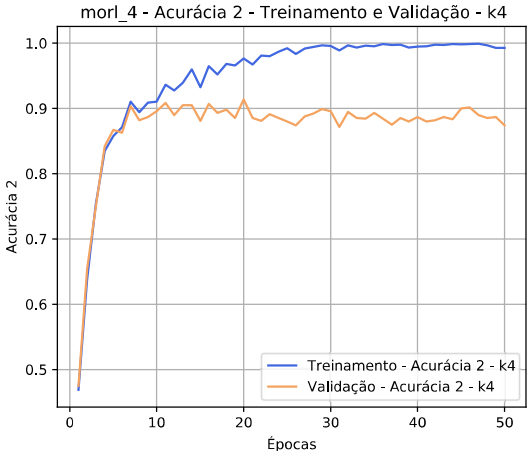


Figura 4.8 Melhor modelo do Experimento 4 (morl_4), k=4. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

baixo. Para o conjunto de validação, a acurácia 0 foi de 49,9%, a acurácia 1 de 84,3% e a acurácia 2 de 87,7%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,0%. Os modelos k3 e k5 atingiram valores máximos de 90,7%. Para o conjunto de teste a acurácia 0 foi de 50,3%, a acurácia 1 de 82,8% e a acurácia 2 de 86,3%.

Tabela 4.5 Resultados do Treinamento (k-*fold*) - Morl_6 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	88,5	93,8	93,1	93,5	92,7	92,3 ± 1,9
ACC1 - treinamento	97,0	99,6	99,3	99,6	99,1	98,9 ± 1,0
ACC2 - treinamento	97,7	99,6	99,5	99,7	99,2	99,1 ± 0,7
Valor máximo ACC2 - treinamento	99,5	99,6	99,9	99,7	99,4	99,6 ± 0,1
ACC0 - validação	46,0	52,0	50,9	51,9	48,8	49,9 ± 2,3
ACC1 - validação	80,3	84,6	84,8	86,1	85,6	84,3 ± 2,1
ACC2 - validação	84,3	88,3	88,8	88,9	88,4	87,7 ± 1,8
Valor máximo ACC2 - validação	89,2	89,0	90,7	90,6	90,7	90,0 ± 0,8
ACC0 - teste	48,1	53,4	50,6	51,1	48,1	50,3 ± 2,0
ACC1 - teste	80,4	85,4	84,7	81,9	81,6	82,8 ± 1,9
ACC2 - teste	83,0	88,4	87,5	86,1	86,3	86,3 ± 1,8

Na Figura 4.9 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados, para o experimento 5. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de validação fica estagnado. Destaca-se uma variação abrupta que o modelo k2 obteve antes da 10 ª época e entre a 10ª época e a 20ª época. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas.

Observando os resultados da Tabela 4.5 e da Figura 4.9 pode-se assumir que o modelo k3 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.10 mostra as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

4.2.3 Experimento 6 - Morl_40

Os resultados do experimento 6, utilizando a wavelet Morlet e com quarenta níveis de escala, são apresentados de maneira similar aos experimentos anteriores. A Tabela 4.6 apresenta os resultados deste sexto experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 52,6%, a acurácia 1 de 86,8% e a acurácia 2 de 90,0%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 91,0%. O modelo k4 atingiu um valor máximo 91,7%. Para o conjunto de teste a acurácia 0 foi de 54,4%, a acurácia 1 de 85,8% e a acurácia 2 de 89,1%.

Na Figura 4.11 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados, para o experimento 6. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de

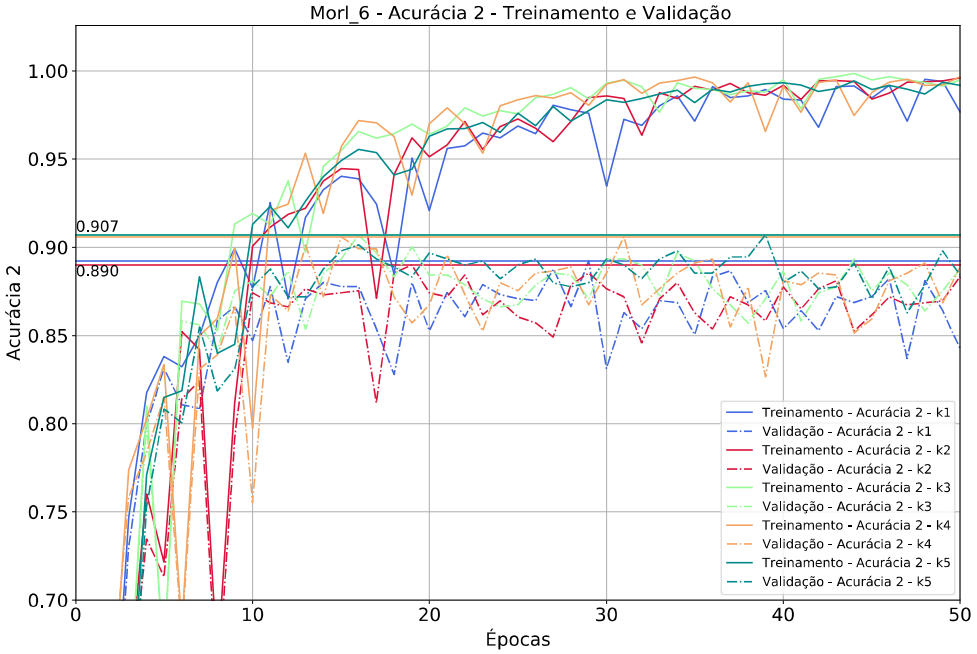


Figura 4.9 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 5 (morl_6). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

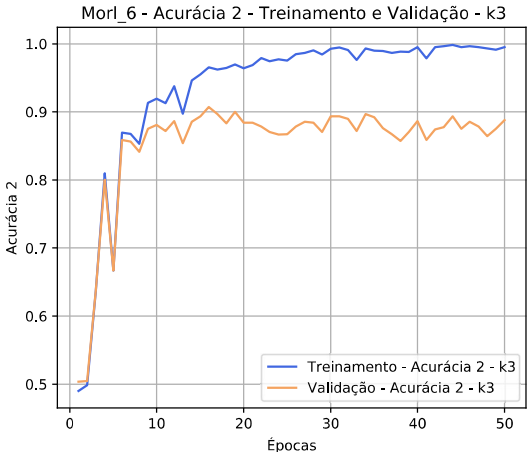


Figura 4.10 Melhor modelo do Experimento 5 (morl_6), k=3. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

validação fica estagnado. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados

Tabela 4.6 Resultados do Treinamento (k-*fold*) - Morl_40 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	93,5	93,7	93,7	93,5	94,4	93,7 ± 0,3
ACC1 - treinamento	99,1	99,3	99,4	99,7	99,8	99,5 ± 0,2
ACC2 - treinamento	99,3	99,5	99,5	99,7	99,8	99,6 ± 0,2
Valor máximo ACC2 - treinamento	99,8	99,9	99,6	99,8	99,8	99,8 ± 0,1
ACC0 - validação	50,5	50,2	54,0	56,5	51,9	52,6 ± 2,3
ACC1 - validação	84,9	86,7	88,2	88,1	86,1	86,8 ± 1,2
ACC2 - validação	89,1	89,8	90,8	91,4	89,0	90,0 ± 0,9
Valor máximo ACC2 - validação	91,5	90,4	91,2	91,7	90,1	91,0 ± 0,6
ACC0 - teste	53,8	53,5	53,2	57,9	53,6	54,4 ± 1,8
ACC1 - teste	86,6	85,1	86,5	86,6	84,1	85,8 ± 1,0
ACC2 - teste	90,5	87,9	89,7	89,7	87,8	89,1 ± 1,1

à esquerda das retas.

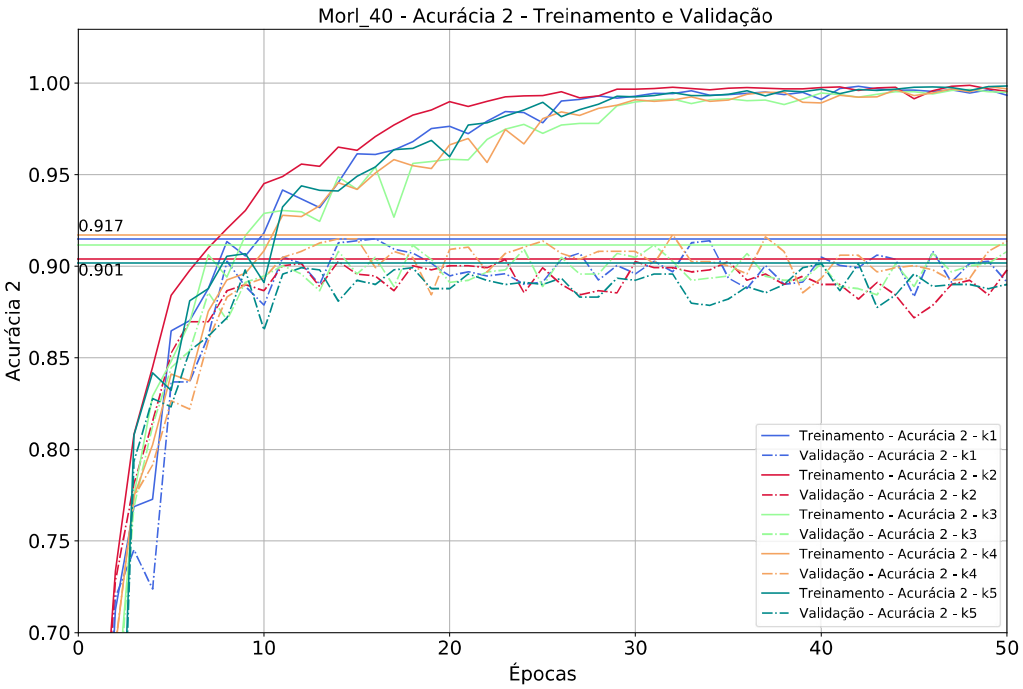


Figura 4.11 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 6 (morl_40). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

Observando os resultados da Tabela 4.6 e da Figura 4.11 pode-se assumir que o modelo k4 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.12 mostra

as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

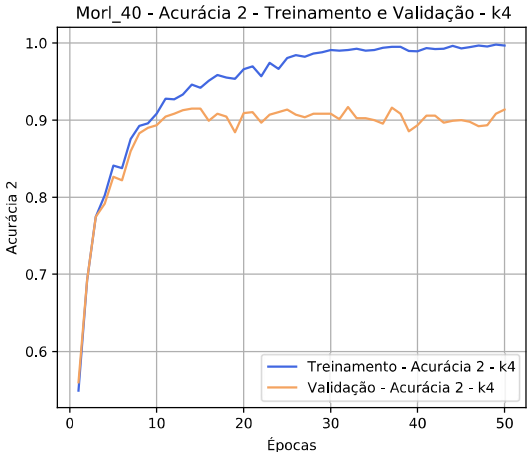


Figura 4.12 Melhor modelo do Experimento 6 (morl_40), k=4. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

4.3 EXPERIMENTOS COM WAVELET SHANNON

4.3.1 Experimento 7 - Shan_4

Os resultados do experimento 7, utilizando a wavelet Shannon e com quatro níveis de escala, são apresentados de maneira similar aos experimentos anteriores. A Tabela 4.7 apresenta os resultados deste sétimo experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 50,9%, a acurácia 1 de 85,8% e a acurácia 2 de 88,6%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,7%. O modelo k5 atingiu um valor máximo 91,2%. Para o conjunto de teste a acurácia 0 foi de 52,1%, a acurácia 1 de 85,4% e a acurácia 2 de 88,7%.

Na Figura 4.13 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados, para o experimento 7. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de validação fica estagnado. Destaca-se uma variação abrupta que o modelo k2 obteve entre a 10ª época e a 20ª época. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas.

Observando os resultados da Tabela 4.7 e da Figura 4.13 pode-se assumir que o modelo k5 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.14 mostra as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

Tabela 4.7 Resultados do Treinamento (k-*fold*) - Shan_4 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	92,2	91,9	93,5	93,7	93,3	92,9 ± 0,7
ACC1 - treinamento	99,1	99,1	99,5	99,6	99,5	99,3 ± 0,2
ACC2 - treinamento	99,2	99,3	99,6	99,7	99,6	99,5 ± 0,2
Valor máximo ACC2 - treinamento	99,7	99,5	99,8	99,8	99,7	99,7 ± 0,1
ACC0 - validação	49,2	48,9	52,2	54,3	50,0	50,9 ± 2,1
ACC1 - validação	84,5	85,7	86,1	85,8	86,7	85,8 ± 0,7
ACC2 - validação	88,3	88,7	88,5	88,0	89,6	88,6 ± 0,5
Valor máximo ACC2 - validação	91,1	90,6	90,0	90,6	91,2	90,7 ± 0,4
ACC0 - teste	50,3	51,4	53,7	53,7	51,6	52,1 ± 1,4
ACC1 - teste	84,3	85,9	86,3	85,8	84,8	85,4 ± 0,8
ACC2 - teste	87,9	88,4	89,0	89,9	88,4	88,7 ± 0,7

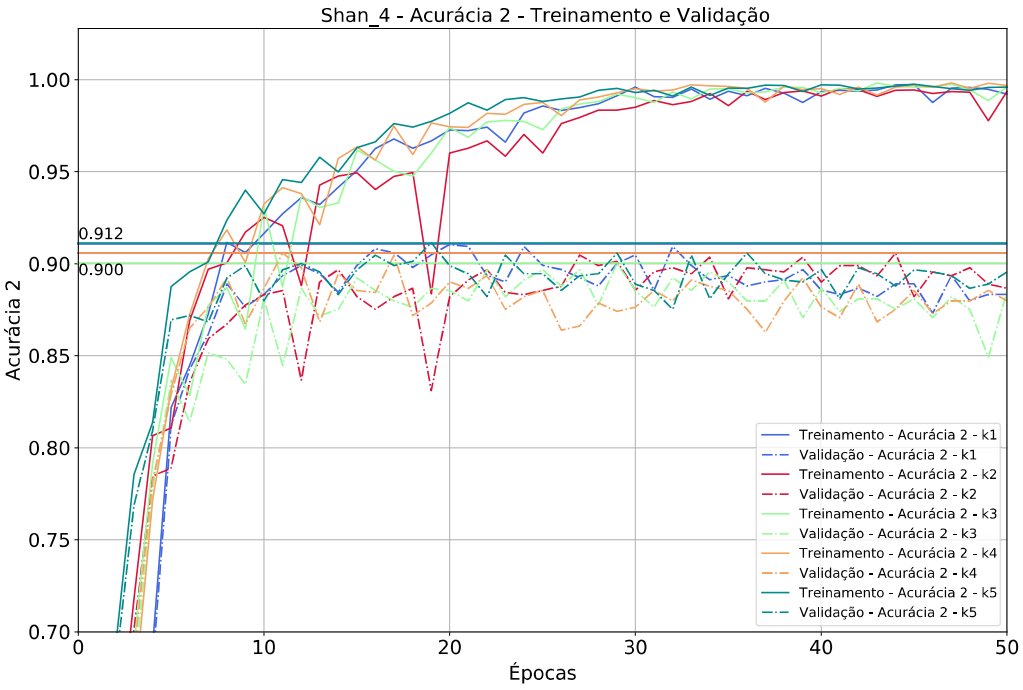


Figura 4.13 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 7 (shan_4). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

4.3.2 Experimento 8 - Shan_6

Os resultados do experimento 8, utilizando a wavelet Shannon e com seis níveis de escala, são apresentados de maneira similar aos experimentos anteriores. A Tabela 4.8

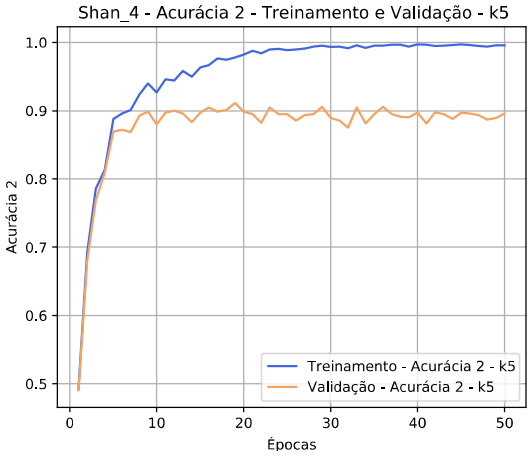


Figura 4.14 Melhor modelo do Experimento 7 (shan_4), k=5. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

apresenta os resultados deste oitavo experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 49,1%, a acurácia 1 de 85,0% e a acurácia 2 de 88,1%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,5%. O modelo k1 atingiu um valor máximo 92,4%. Para o conjunto de teste a acurácia 0 foi de 49,9%, a acurácia 1 de 84,2% e a acurácia 2 de 87,5%.

Tabela 4.8 Resultados do Treinamento (k-*fold*) - Shan_6 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	92,2	93,4	91,5	92,0	92,0	92,2 ± 0,6
ACC1 - treinamento	99,4	99,1	99,0	98,7	98,9	99,0 ± 0,2
ACC2 - treinamento	99,5	99,4	99,4	99,0	99,2	99,3 ± 0,2
Valor máximo ACC2 - treinamento	99,7	99,8	99,8	99,9	99,5	99,7 ± 0,1
ACC0 - validação	48,6	50,6	48,1	51,0	47,1	49,1 ± 1,5
ACC1 - validação	86,2	86,2	83,6	84,9	84,4	85,0 ± 1,0
ACC2 - validação	90,0	88,7	86,5	87,9	87,3	88,1 ± 1,2
Valor máximo ACC2 - validação	92,4	90,1	89,3	90,5	89,9	90,5 ± 1,0
ACC0 - teste	50,1	52,2	47,7	49,7	50,1	49,9 ± 1,4
ACC1 - teste	84,5	85,8	84,7	83,4	82,5	84,2 ± 1,1
ACC2 - teste	87,8	88,8	87,8	87,2	85,9	87,5 ± 0,9

Na Figura 4.15 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados, para o experimento 8. É possível observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de validação fica estagnado. Destaca-se uma variação abrupta que o modelo k2 obteve entre a 10ª época e a 20ª época. O modelo k5 também obteve uma variação significativa entre a 20ª e 30ª época. Foram traçadas retas horizontais onde ocorreram os valores máximos de

cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas.

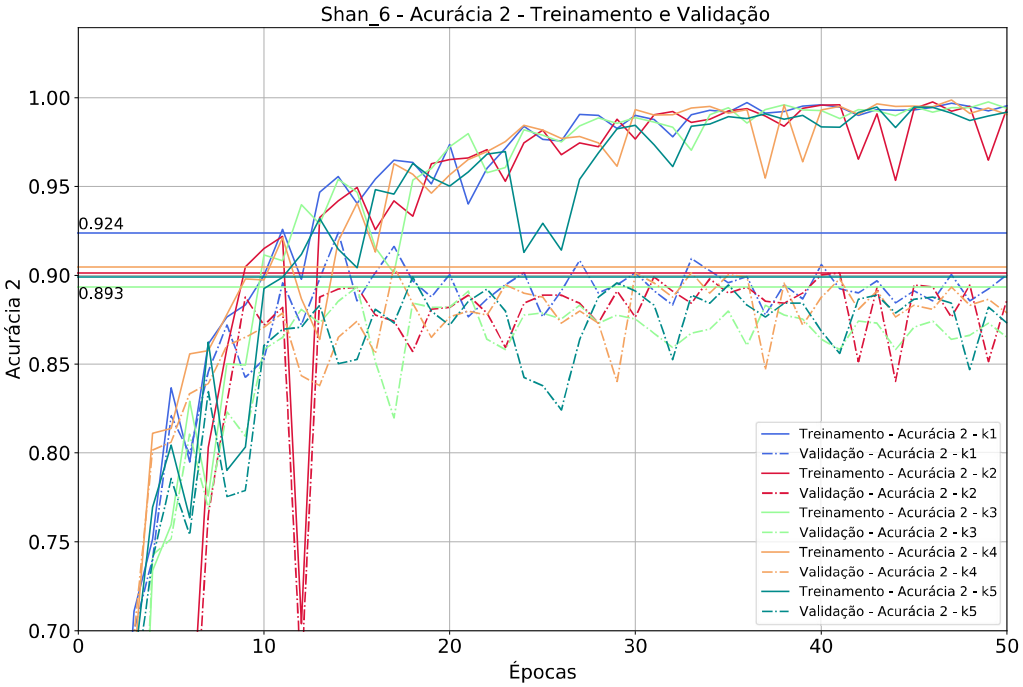


Figura 4.15 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 8 (shan_6). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

Observando os resultados da Tabela 4.8 e da Figura 4.15 pode-se assumir que o modelo k1 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.16 mostra as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

4.3.3 Experimento 9 - Shan_40

Os resultados do experimento 9, utilizando a wavelet Shannon e com quarenta níveis de escala, são apresentados de maneira similar aos experimentos anteriores. A Tabela 4.9 apresenta os resultados deste nono experimento. Os resultados de Acurácia 0, 1 e 2 no conjunto de treinamento atingiram altos valores, acima de 90% de acerto e desvio padrão baixo. Para o conjunto de validação, a acurácia 0 foi de 52,8%, a acurácia 1 de 86,0% e a acurácia 2 de 89,1%. A média dos valores máximos de acurácia 2 ao longo do treinamento foi 90,6%. O modelo k5 atingiu um valor máximo 91,1%. Para o conjunto de teste a acurácia 0 foi de 53,5%, a acurácia 1 de 87,5% e a acurácia 2 de 90,0%.

Na Figura 4.17 as curvas de acurácia 2, em percentual, de treinamento e validação são apresentadas para todos os k modelos treinados, para o experimento 9. É possível

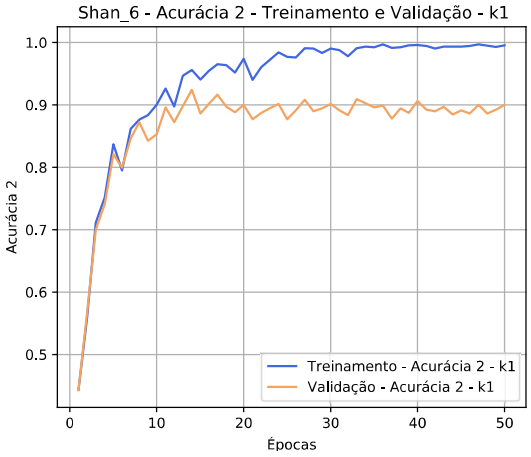


Figura 4.16 Melhor modelo do Experimento 8 (shan_6), k=1. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

Tabela 4.9 Resultados do Treinamento (k-*fold*) - Shan_40 (%)

	k1	**k2**	**k3**	**k4**	**k5**	**Resultado Final**
ACC0 - treinamento	94,7	92,4	93,4	94,6	94,9	94,0 ± 1,0
ACC1 - treinamento	99,8	99,2	99,4	99,9	99,8	99,6 ± 0,3
ACC2 - treinamento	99,9	99,3	99,6	99,9	99,9	99,7 ± 0,2
Valor máximo ACC2 - treinamento	99,9	99,8	99,9	99,9	99,9	99,9 ± 0,0
ACC0 - validação	51,5	54,1	52,4	52,3	53,8	52,8 ± 1,0
ACC1 - validação	87,0	86,1	83,6	86,4	87,0	86,0 ± 1,3
ACC2 - validação	89,6	88,5	87,7	89,8	90,0	89,1 ± 0,9
Valor máximo ACC2 - validação	91,0	90,1	90,6	90,4	91,1	90,6 ± 0,3
ACC0 - teste	52,7	49,8	54,1	56,0	55,0	53,5 ± 2,2
ACC1 - teste	88,1	87,6	87,3	87,3	87,2	87,5 ± 0,3
ACC2 - teste	89,5	90,6	90,6	89,3	89,8	90,0 ± 0,5

observar o comportamento similar para todos os valores de k, onde a partir da 10ª época o modelo se aprimora no conjunto de dados de treinamento e o conjunto de dados de validação fica estagnado. Foram traçadas retas horizontais onde ocorreram os valores máximos de cada modelo. O melhor valor máximo e o pior valor máximo estão sinalizados à esquerda das retas.

Observando os resultados da Tabela 4.9 e da Figura 4.17 pode-se assumir que o modelo k5 atingiu os melhores resultados. Para uma melhor visualização, a Figura 4.18 mostra as curvas de acurácia 2 de treinamento e validação apenas para este modelo.

4.4 COMPARATIVO ENTRE FUNÇÕES WAVELET

Os resultados obtidos nos experimentos realizados foram bastante próximos entre si, tanto em comportamento das curvas de treinamento e validação, como valores finais e máximos dos modelos. Por isso, a análise para definição do melhor modelo foi minuciosa,

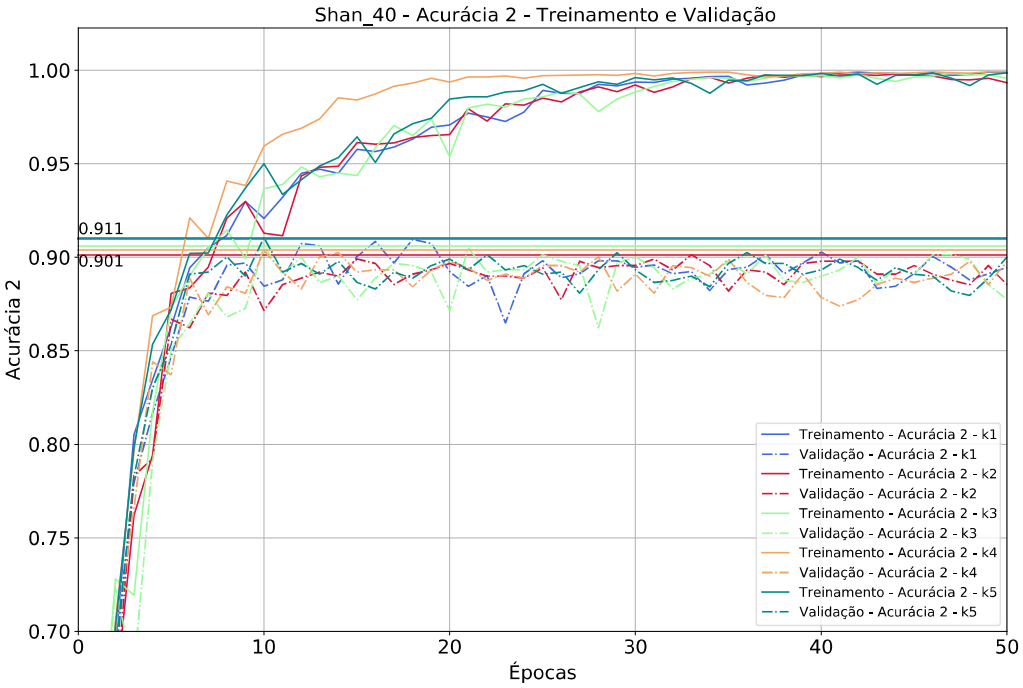


Figura 4.17 Resultados de treinamento e validação para todos os modelos k-fold do Experimento 9 (shan_40). As curvas contínuas representam a acurácia 2 de treinamento de cada modelo. As curvas traço-ponto representam a acurácia 2 de validação de cada modelo. Retas horizontais foram traçadas para visualizar o melhor valor da acurácia 2 ao longo da validação. O valores do melhor modelo e do pior modelo estão expostos no canto esquerdo das retas.

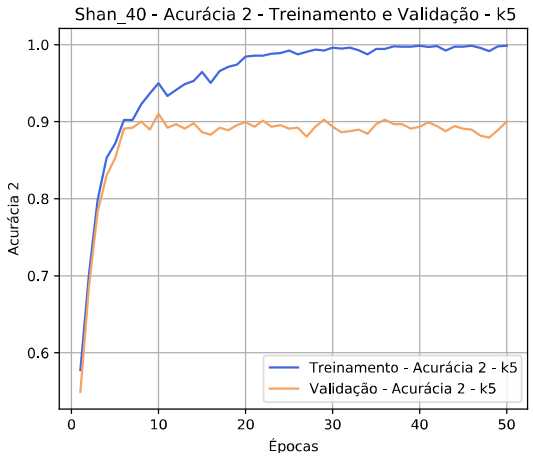


Figura 4.18 Melhor modelo do Experimento 9 (shan_40), k=5. São apresentadas as curvas de acurácia 2 para o conjunto de treinamento e validação.

observando a média e desvio padrão dos principais indicadores. A Tabela 4.10 mostra os resultados do valor máximo de validação atingido durante o treinamento. Este indicador é importante, pois através dele pode-se retreinar a rede utilizando a técnica de *early stopping*, interrompendo o treinamento próximo ao valor máximo. Desta forma o modelo irá atingir melhores resultados nos conjuntos de avaliação. O melhor resultado foi conseguido com a wavelet Morlet, utilizando 40 níveis de escala (Morl_40).

Tabela 4.10 Comparativo entre os valores máximos de Acurácia 2 no conjunto de validação, ao longo do treinamento

Experimento	**Escalograma**	**Resultado Final (%)**
1	Mexh_4	90,8 ± 0,7
2	Mexh_6	90,8 ± 0,5
3	Mexh_40	90,3 ± 0,7
4	Morl_4	90,8 ± 0,5
5	Morl_6	90,0 ± 0,8
6	Morl_40	**91,0 ± 0,6**
7	Shan_4	90,7 ± 0,4
8	Shan_6	90,5 ± 1,0
9	Shan_40	90,6 ± 0,3

Na Figura 4.19 é apresentando um gráfico *box plot*, onde estão sintetizados os resultados (*k-fold*) de validação e teste ao final de 50 épocas. Cada experimento está representado em cores diferentes e as colunas estão em ordem, primeiramente o conjunto de validação e posteriormente o conjunto de teste. Analisando este gráfico é possível afirmar que a wavelet Morlet, utilizando 40 níveis de escala (Morl_40), atingiu melhores resultados para o conjunto de validação. Enquanto a wavelet Shannon, também utilizando 40 níveis de escala (Shan_40), obteve melhores resultados no conjunto de teste.

Estes resultados levam a concluir que o modelo se comportou melhor com os escalogramas gerados a partir de 40 níveis de escala. Isto era previsto, pois estes são os escalogramas que apresentam melhor resolução, e consequentemente mais informação. Visualmente analisando, a wavelet analisadora Chapéu Mexicano foi a que apresentou menos variações de intensidade, e isto pode ter relação com os resultados do experimento Mexh_40, que foram um pouco abaixo que os demais com 40 níveis de escala.

É interessante notar que os resultados mais baixos foram os com escalogramas gerados a partir de 6 níveis de escala. Isto também pode ser entendido ao observar visualmente os escalogramas gerados desta forma. Eles apresentam uma grande parte sem informação, visto que a maioria das músicas não possui sinais naquelas faixas de frequência.

A Figura 4.20 mostra os gráficos de valores reais *versus* previsões realizadas pelo modelo. Os valores estão em BPM. Os gráficos da coluna da esquerda, Figuras 4.20 (a), 4.20 (c) e 4.20 (e), representam os resultados dos conjuntos de treinamento validação e teste, respectivamente, para o modelo Morl_40. Já os gráficos da coluna da direita, Figuras 4.20 (b), 4.20 (d) e 4.20 (f), representam os resultados dos conjuntos de treinamento validação e teste, respectivamente, para o modelo Shan_40.

Nestes gráficos, a reta em vermelho representa a acurácia 0, quando o modelo estima exatamente o valor de andamento real. A zona sombreada em vermelho representa a

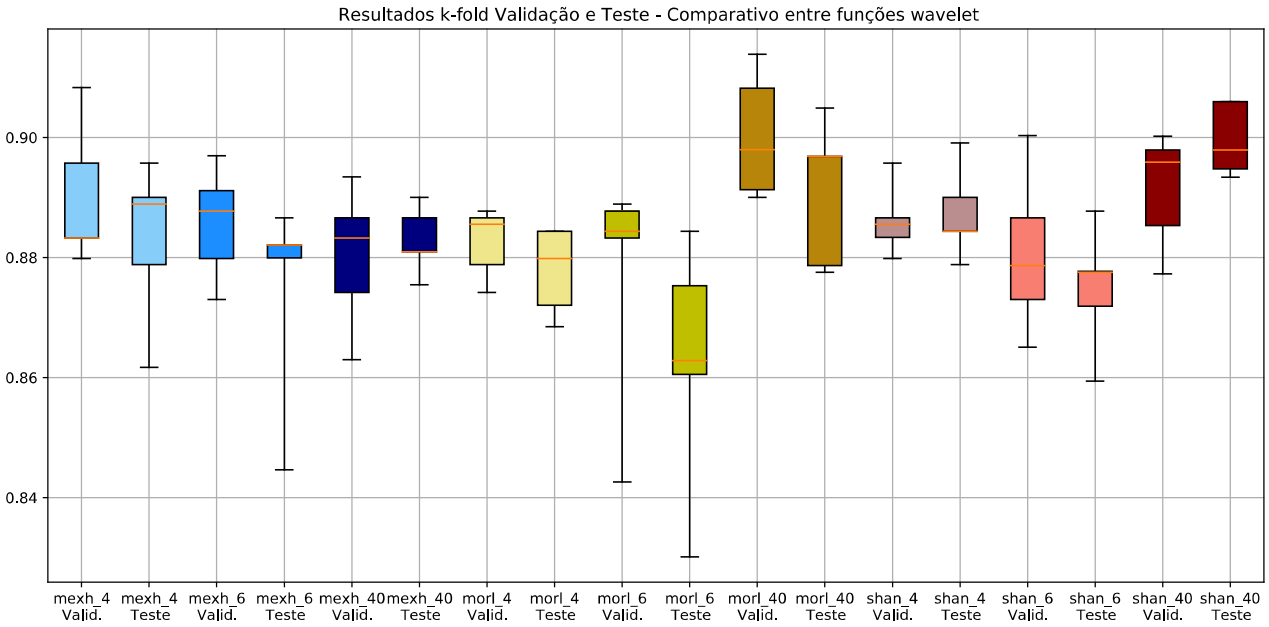


Figura 4.19 *Box plot* comparativo entre os resultados do treinamento k-*fold* para os conjuntos de dados de validação e teste, para todos os experimentos.

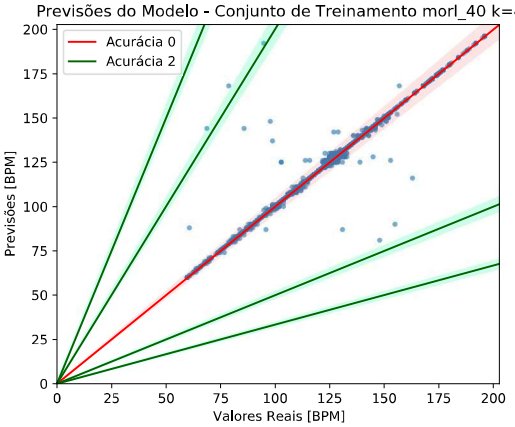
acurácia 1, considerando a margem de 4%. As retas em verde representam os múltiplos e submúltiplos e as zonas sombreadas em verde quando considerada a margem de 4% para estes casos. Para o conjunto de treinamento, Figuras 4.20 (a) e 4.20 (b), os gráficos se comportaram como esperado, concentrando a maioria dos exemplos sobre a linha em vermelho, e alguns poucos erros. Para a validação, Figuras 4.20 (c) e 4.20 (d), os erros aumentam e alguns exemplos aparecem nas retas de acurácia 2. O conjunto de teste apresenta comportamento similar ao conjunto de validação para ambos os casos, Figuras 4.20 (e) e 4.20 (f).

4.5 DEFINIÇÃO DO PARÂMETRO DO AUMENTO ARTIFICIAL DE DADOS

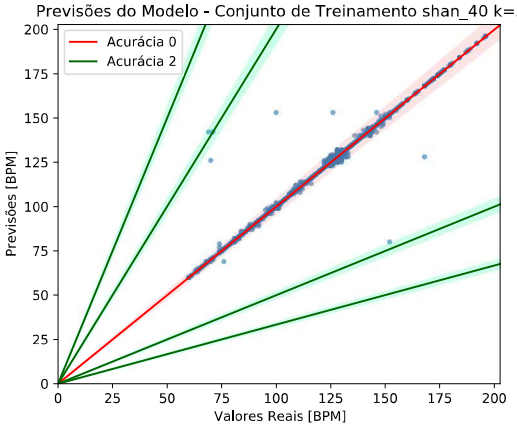
Após a definição dos dois melhores métodos de geração dos escalogramas, Morl_40 e Shan_40, a estratégia foi alterada visando os melhores resultados no banco de dados de avaliação. O dois modelos foram treinados com todos os exemplos dos bancos de dados de treinamento, sem a divisão do método de validação *k-fold*. Para a validação do modelo, foi utilizado a junção dos bancos de dados de avaliação, "Combinados".

A Tabela 4.11 mostra os valores máximos de acurácia 2 para o conjunto de validação, que neste caso é o "Combinados", para os dois melhores métodos de geração dos escalogramas. Os valores vão aumentando à medida que o parâmetro DA é aumentado. Ou seja, quanto mais dados artificias são inserido na rede neural convolucional, mais ela atinge melhores resultados nos dados reais que ela não foi treinada.

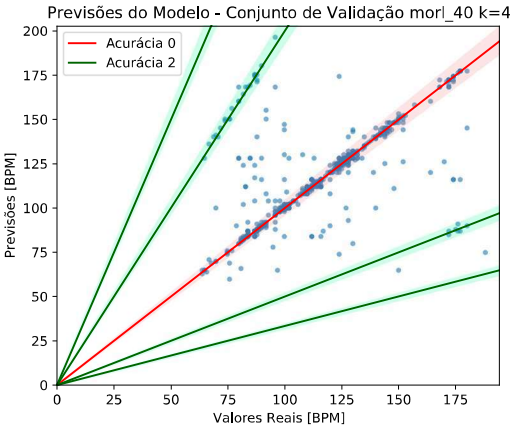
A variação do parâmetro DA significa a quantidade de exemplos que irão sofrer in-



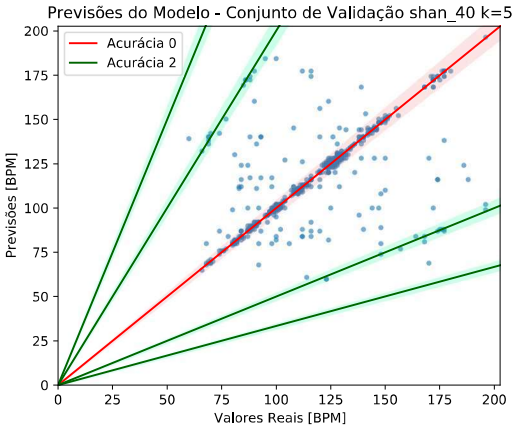
(a) Morl_40 - Treinamento.



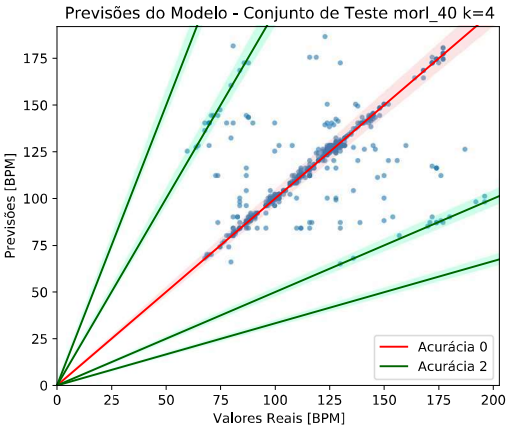
(b) Shan_40 - Treinamento.



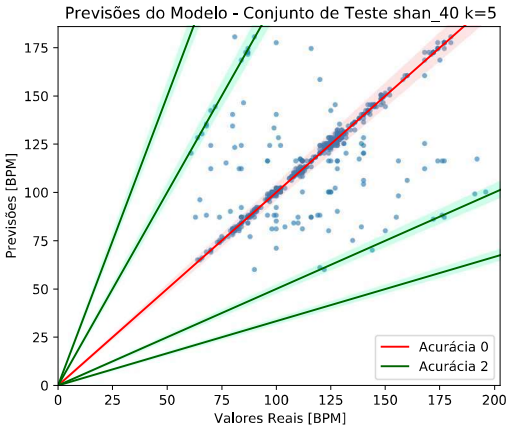
(c) Morl_40 - Validação.



(d) Shan_40 - Validação.



(e) Morl_40 - Teste.



(f) Shan_40 - Teste.

Figura 4.20 Valores Reais *versus* Previsões dos conjuntos de treinamento, validação e teste dos modelos propostos Morl_40 (k=4) e Shan_40 (k=5).

fluência do processo de aumento artificial de dados. Para $DA = 100\%$, ou seja, todos os exemplo passarão por um processo de expansão ou compressão e posteriormente uma janela aleatória será escolhida para realizar o treinamento, os resultados foram 87,8% para o modelo Morl_40 e 88,4% para o modelo Shan_40.

As Figuras 4.21 e 4.22 mostram as curvas de validação para os valores de DA. Percebe-se que a melhoria da performance no conjunto de validação é bastante significante variando de $DA = 0\%$ para $DA = 25\%$. A partir daí, a variação vai ficando cada vez mais sutil, porém com melhorias, até chegar a $DA = 100\%$.

Tabela 4.11 Comparativo entre a variação do parâmetro DA. Valores máximos de acurácia 2 ao longo do treinamento para o conjunto de validação.

	Acurácia 2 (%)	
DA(%)	**Morl_40**	**Shan_40**
0	81,3	81,6
25	86,0	85,7
50	86,3	86,5
75	87,3	87,7
100	87,8	88,4

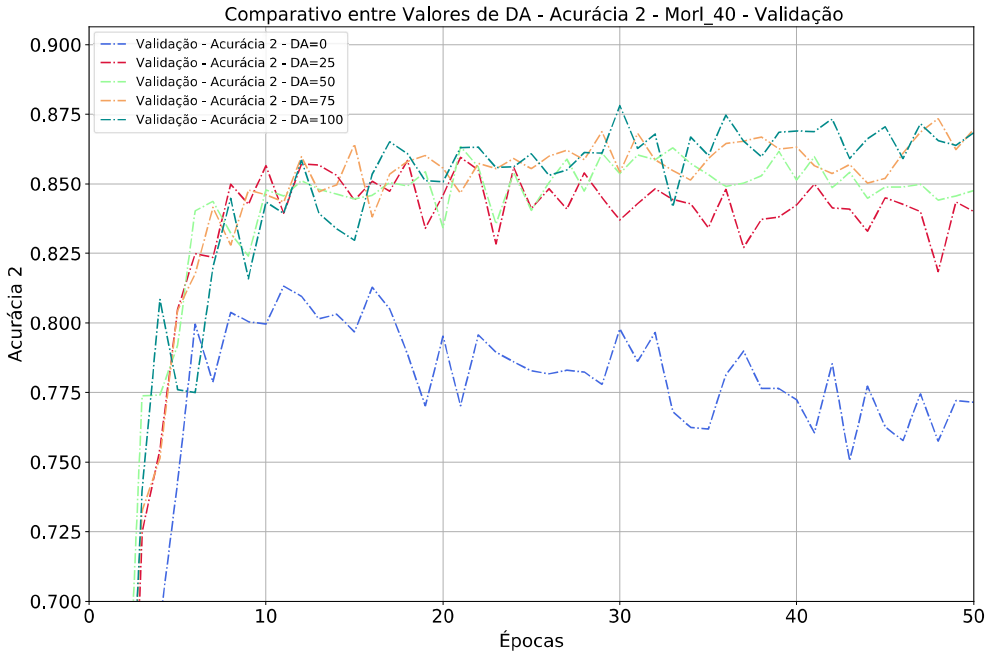


Figura 4.21 Curvas de validação para o treinamento do modelo Morl_40 com variação de valores de DA.

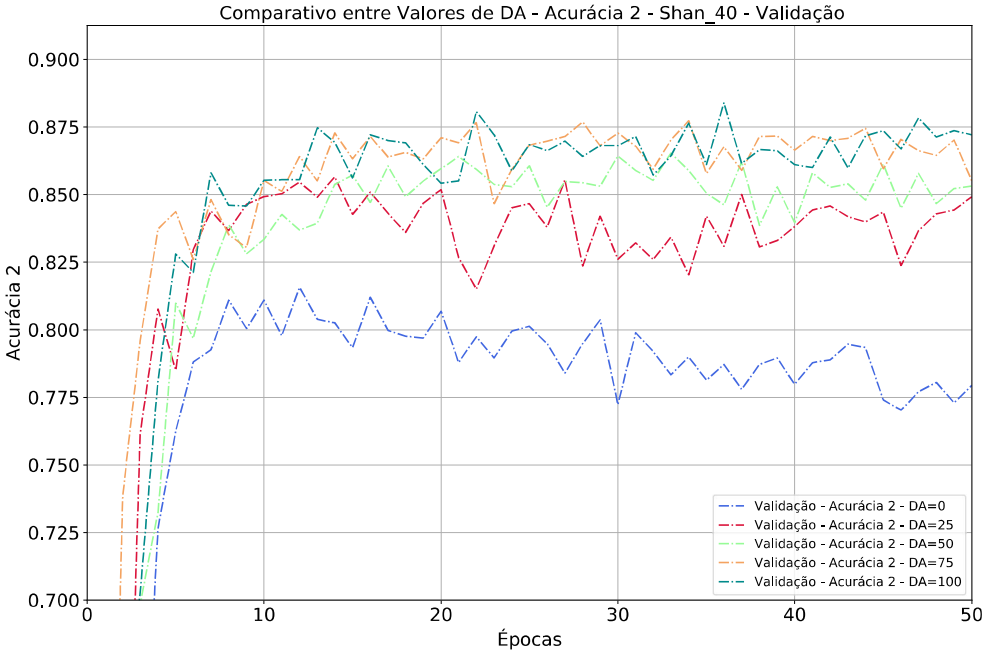


Figura 4.22 Curvas de validação para o treinamento do modelo Shan_40 com variação de valores de DA.

4.6 AVALIAÇÃO DO MODELO PROPOSTO E COMPARATIVO COM O ESTADO DA ARTE

Em 2020, Schreiber *et al.* [12] apontaram os trabalhos de Schreiber e Müller [7] e Böck *et al.*[9] como os principais trabalhos de estimativa de andamento musical. Até o momento, não há registros de um trabalho que tenha superado a performance destes algoritmos em todos os bancos de dados estudados. Por isso, estes trabalhos foram escolhidos para serem comparados com os modelos propostos.

Para que os melhores resultados do modelo proposto sejam atingidos, foi definido o parâmetro $DA = 100\%$ e o método de previsão por janelas aleatórias foi utilizado. Foram considerados dois modelos com métodos de geração do escalograma diferentes, Morl_40 e Shan_40. Ambos os modelos foram treinados com todos os bancos de dados de treinamento. Para o Morl_40, foi utilizado um *early stopping* para interromper o treinamento quando a acurácia 2 para o conjunto de validação fosse maior que 87,5%, enquanto que para o Shan_40 o treinamento seria interrompido quando a acurácia 2 para o conjunto de validação fosse maior que 88,0%. Para ambos os modelos o limite máximo de épocas foi 50.

No caso do modelo Morl_40 o treinamento durou as 50 épocas e a acurácia 2 para o conjunto de validação não ultrapassou 87,5%. Para o modelo Shan_40, o treinamento foi interrompido na 31ª época, com 88,02%. É importante enfatizar que durante o treinamento o método de previsão por janelas aleatórias não é utilizado.

As Tabelas 4.12, 4.13 e 4.14 mostram os resultados para acurácia 0, 1 e 2 dos modelos Morl_40 e Shan_40 comparados ao estado da arte, respectivamente. Para a acurácia 0, era esperado que os resultados fossem inferiores ao estado da arte devido ao treinamento ter sido realizado com o conjunto *sweet octave*. Porém, o modelo conseguiu resultados próximos com o banco de dados Ballroom. Isso ocorreu porque a maioria dos exemplos deste banco possui um valor de andamento dentro do intervalo que a rede é capaz de prever.

Este também é o motivo do modelo ter atingido resultados baixos para acurácia 0 no banco de dados SMC Mirum. A maioria dos exemplos deste banco estão próximos ou fora do limite do intervalo de treinamento. Além disso, o SMC Mirum é composto por música clássica, estilo com poucos instrumentos percussivos e pouco representado nos bancos de dados de treinamento, o que justifica o resultado ruim em todas as acurácias. A qualidade das anotações dos bancos também pode ser outro motivo pelo qual o modelo proposto obteve baixos valores de acurácia para este banco de dados.

Para a acurácia 1, exposta na Tabela 4.13, destaca-se o resultado atingido no banco de dados GiantSteps, que superou o estado da arte com 75,6% de acerto com o modelo Shan_40. O GiantSteps é composto por músicas eletrônicas, o que permite que o andamento da música seja rigorosamente constante. A maior dificuldade que um modelo encontra ao estimar um andamento com este estilo são os efeitos sonoros inseridos nestas peças musicais que pode interferir na percepção do andamento. O escalograma se mostrou eficaz em abstrair estes efeitos e permitir que o modelo classificasse corretamente o andamento.

Analisando os resultados de acurácia 2 na Tabela 4.14 pode-se observar que, mais uma vez, os modelos propostos atingiram bons resultados com o banco de dados GiantSteps chegando a um resultado de 92,6% com o Morl_40 e 92,3% com o Shan_40, superando o estado da arte. Para o banco de dados "Combinados" o modelo Shan_40 conseguiu resultados próximos ao estado da arte, com uma acurácia 2 de 90,1%. Para o banco SMC Mirum, os resultados foram consideravelmente inferiores ao estado da arte.

A Figura 4.23 mostram os gráficos de valores reais *versus* previsões dos modelos Morl_40 e Shan_40 para todos os bancos de dados de avaliação. São apresentados oito duplas de gráficos, uma para cada banco de dados. Analisando as Figuras 4.23 (a) e 4.23 (b), percebe-se que, para o ACM Mirum, as previsões se concentraram nas regiões de acurácia 0 e 1, mas também o erro ocorreu exatamente em uma das retas da acurácia 2, onde a rede estima o dobro do valor real. As Figuras 4.23 (c) e 4.23 (d) mostram os gráficos para o Ballroom, na qual ocorre uma concentração de pontos na área de acurácia 1, com alguns exemplos nas retas $y = x/2$ e $y = 2x$. Nas Figuras 4.23 (e) e 4.23 (f) estão apresentados os resultados para o banco de dados GiantSteps, pode-se notar erros mais distribuídos pelo gráfico, mas com a maioria dos pontos concentrados nas áreas de acurácia 1.

As Figuras 4.23 (g) e 4.23 (h) mostram os gráficos pra o banco de dados GTzan, que apresenta comportamento similar ao ACM Mirum. Pode-se notar também uma concentração de pontos entre as retas $y = x$ e $y = 2x$. É possível visualizar os resultados para o banco de dados Hainsworth nas figuras 4.23 (i) e 4.23 (j). A maioria dos pontos estão na área de acurácia 1 para este caso. Nas Figuras 4.23 (k) e 4.23 (l) são mostrados os

gráficos para o banco de dados ISMIR04, com comportamento parecido ao ACM Mirum. Os resultados para o banco de dados SMC Mirum estão nas Figuras 4.23 (m) e 4.23 (n). Para este banco o erro foi o maior e percebe-se o surgimento de retas horizontais no comportamento de previsão dos modelos propostos. Isto significa que a rede está prevendo um mesmo valor de andamento para uma série de exemplos. Por fim, a figura 4.23 (o) e 4.23 (p) mostram os gráficos para o banco de dados combinados e é possível ver o comportamento de todos os bancos em conjunto. A maioria dos exemplos se concentram nas áreas de acurácia 2 e a maior parte dos erros ocorre entre $y = x/2$ e $y = 2x$.

Tabela 4.12 Comparação com o estado da arte - Acurácia 0 (%)

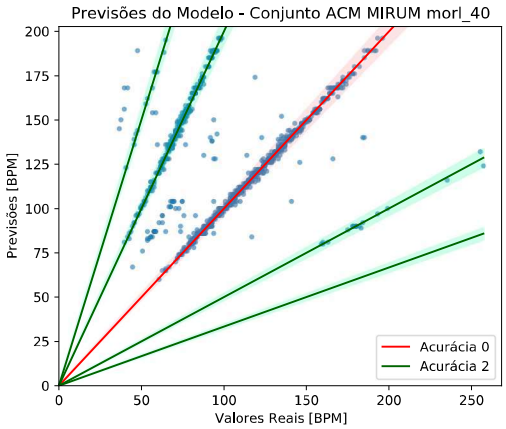
Banco de Dados de Avaliação	Schr	Böck	Morl_40	Shan_40
ACM Mirum	40,6	29,4	28,2	23,7
Ballroom	67,9	33,8	60,6	61,7
GiantSteps	59,8	37,2	17,3	19,5
GTzan	36,9	32,2	28,6	27,1
Hainsworth	43,2	33,8	31,7	30,3
ISMIR04	34,1	27,2	20,6	21,5
SMC	12,4	17,1	7,8	6,5
Combinados	44,8	31,2	30,4	29,2

Tabela 4.13 Comparação com o estado da arte - Acurácia 1 (%)

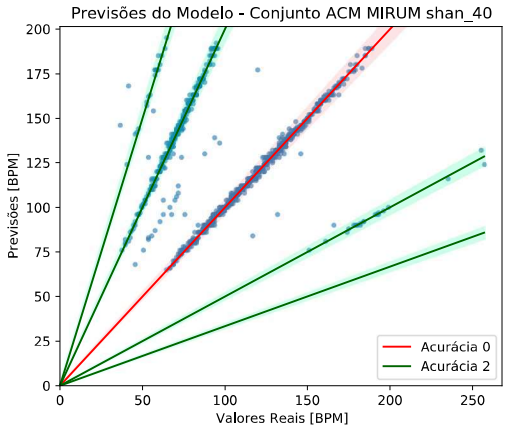
Banco de Dados de Avaliação	Schr	Böck	Morl_40	Shan_40
ACM Mirum	79,5	74,0	67,4	64,4
Ballroom	92,0	84,0	79,2	79,8
GiantSteps	73,0	58,9	**73,3**	**75,6**
GTzan	69,4	69,7	59,2	59,4
Hainsworth	77,0	80,6	72,9	72,9
ISMIR04	60,6	55,0	47,7	48,4
SMC	33,6	44,7	23,0	21,1
Combinados	74,2	69,5	64,8	64,1

Tabela 4.14 Comparação com o estado da arte - Acurácia 2 (%)

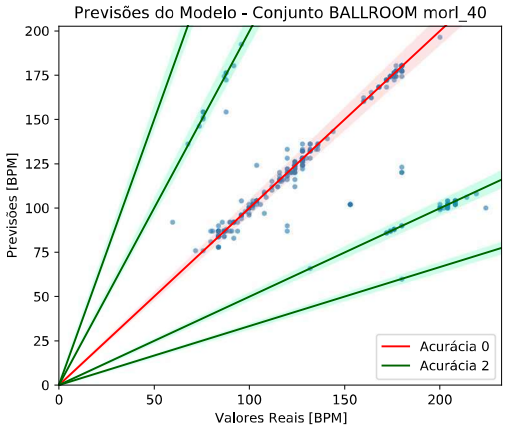
Banco de Dados de Avaliação	Schr	Böck	Morl_40	Shan_40
ACM Mirum	97,4	97,7	95,5	96,7
Ballroom	98,4	98,7	93,6	93,8
GiantSteps	89,3	86,4	**92,6**	**92,3**
GTzan	92,6	95,0	88,6	90,4
Hainsworth	84,2	89,2	83,7	81,4
ISMIR04	92,2	95,0	85,2	87,3
SMC	50,2	67,3	45,2	44,2
Combinados	92,1	93,6	89,6	90,1



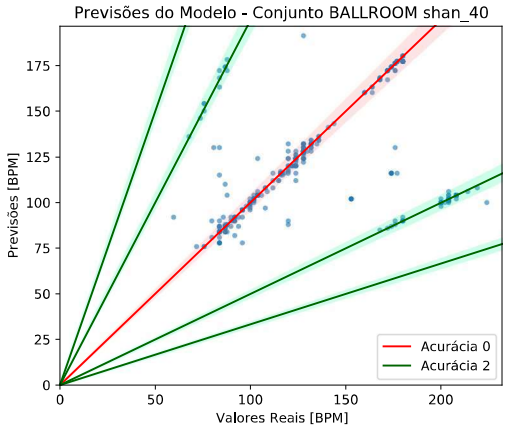
(a) Morl_40 - ACM Mirum.



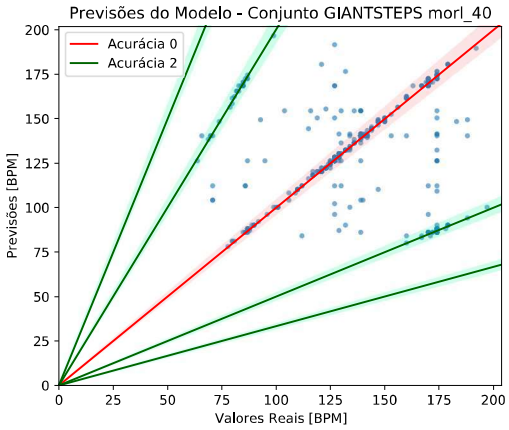
(b) Shan_40 - ACM Mirum.



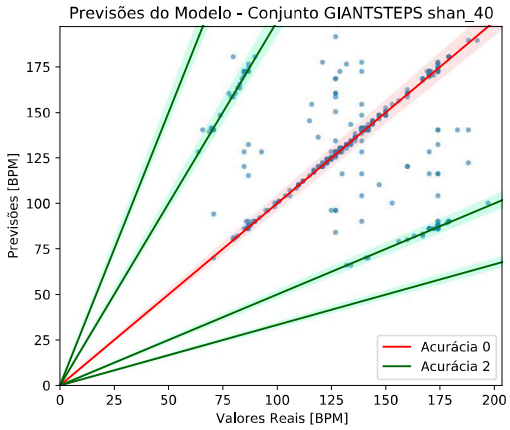
(c) Morl_40 - Ballroom.



(d) Shan_40 - Ballroom.

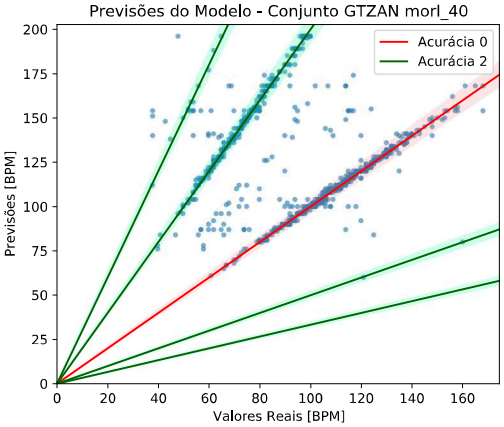


(e) Morl_40 - GiantSteps.

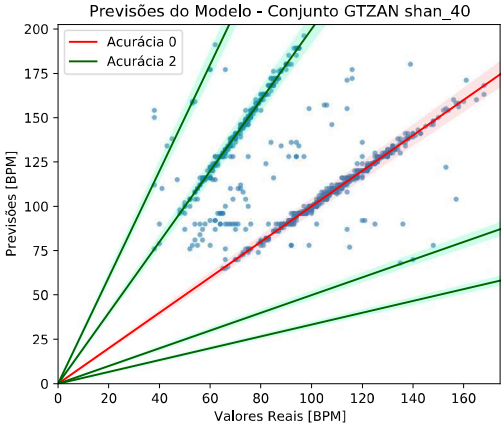


(f) Shan_40 - GiantSteps.

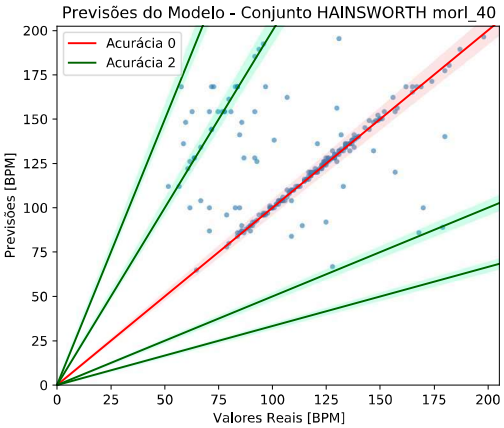
Figura 4.23 Valores Reais *versus* Previsões dos modelos propostos Morl_40 e Shan_40 gerados a partir dos bancos de dados de avaliação (continua).



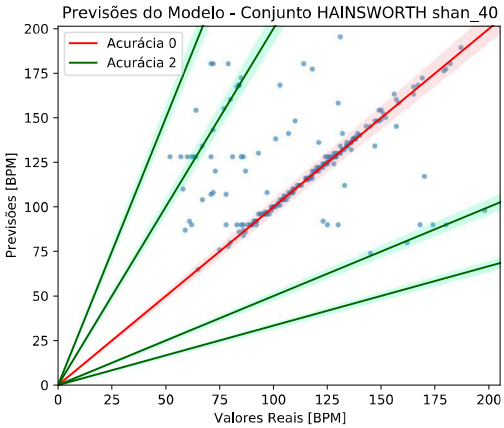
(g) Morl_40 - GTzan.



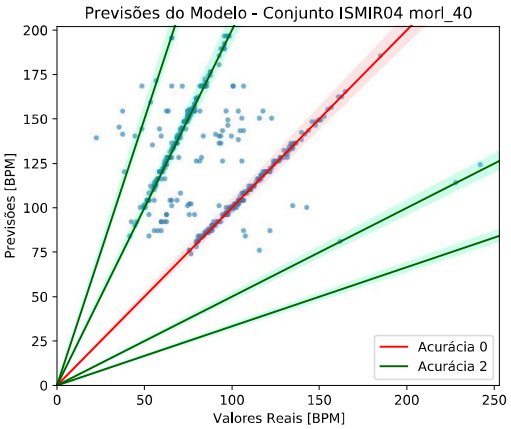
(h) Shan_40 - GTzan.



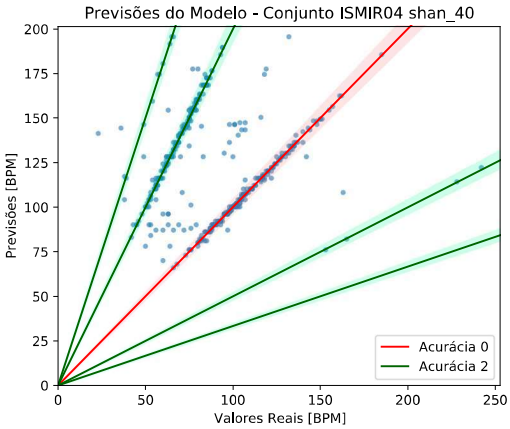
(i) Morl_40 - Hainsworth.



(j) Shan_40 - Hainsworth.

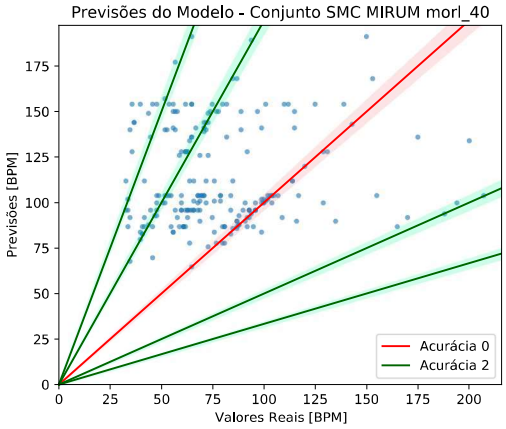


(k) Morl_40 - ISMIR04.

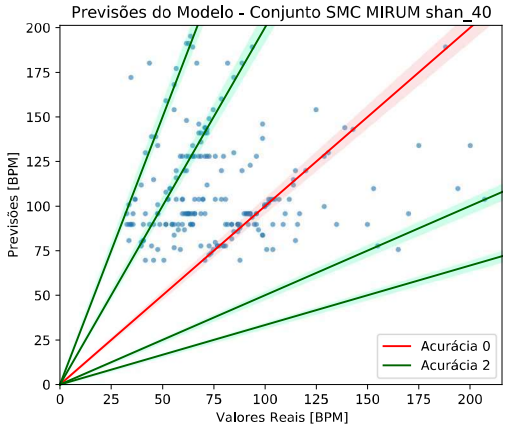


(l) Shan_40 - ISMIR04.

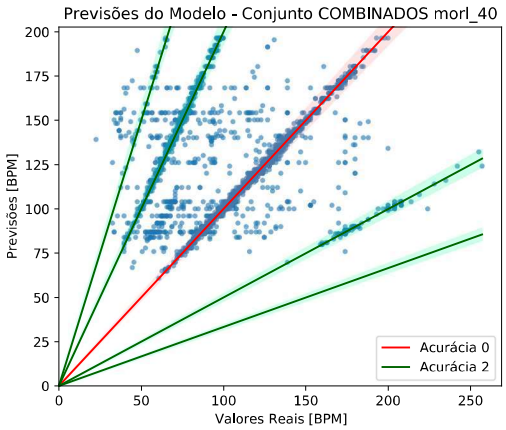
Figura 4.23 Valores Reais *versus* Previsões dos modelos propostos Morl_40 e Shan_40 gerados a partir dos bancos de dados de avaliação (continua).



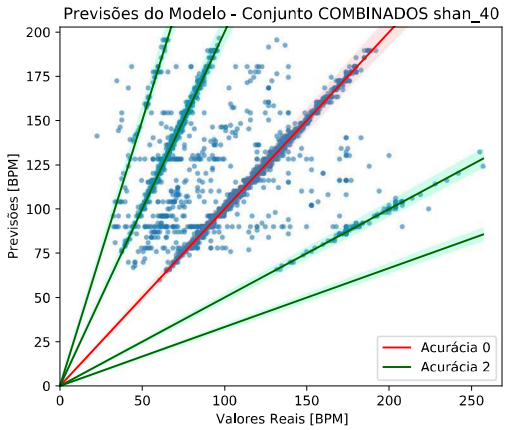
(m) Morl_40 - SMC.



(n) Shan_40 - SMC.



(o) Morl_40 - Combinados.



(p) Shan_40 - Combinados.

Figura 4.23 Valores Reais *versus* Previsões dos modelos propostos Morl_40 e Shan_40 gerados a partir dos bancos de dados de avaliação.

Capítulo 5

CONSIDERAÇÕES FINAIS

5.1 CONCLUSÕES

A estimativa de andamento musical é um problema que permanece em aberto na área de MIR. Baseado nessa problemática, este trabalho desenvolveu um modelo de rede neural convolucional treinado com escalogramas wavelet gerados à partir de sinais de áudio. Após o treinamento, o modelo proposto é capaz de estimar valores de andamento à partir de uma peça musical.

No Capítulo 2 discutiram-se as bases teóricas necessárias para implementação do trabalho. Entre elas, os principais conceitos da MIR e a teoria wavelet necessária para geração dos escalogramas. Também foram estudadas as máquinas de aprendizado, com ênfase nas redes neurais convolucionais que foram as ferramentas utilizadas. Estes tópicos foram essenciais para o desenvolvimento da metodologia do trabalho.

O modelo proposto foi apresentado no Capítulo 3. Foram expostos e referenciados os bancos de dados utilizados para o treinamento e avaliação do modelo. Foram mostrados os métodos de geração dos escalogramas e definidos todos os diferentes parâmetros. A variação destes parâmetros resultou em nove experimentos que foram propostos para avaliar os escalogramas gerados a partir das funções wavelet analisadoras Chapéu Mexicano, Morlet e Shannon. Foi apresentada também a arquitetura proposta da rede neural convolucional e definidos os indicadores que foram avaliados durante e ao final do treinamento.

Ainda no Capítulo 3 foi proposto o método de aumento artificial de dados de forma online. Todo o exemplo musical foi convertido em um escalograma e aleatoriamente expandido ou comprimido e seu valor de andamento reajustado. A partir daí, uma janela aleatória foi escolhida para realizar o treinamento naquela época, e o processo se repetiu à cada época que a rede neural convolucional foi treinada. Desta forma, a exposição da rede a novos exemplos aumentou significativamente. Esse aumento de dados pôde ser variado através do parâmetro DA, diminuindo ou aumentando a sua atuação. Por fim, foi proposto um método de avaliação do modelo definido como previsão por janelas aleatórias, onde 30 janelas foram definidas aleatoriamente ao logo da música e a rede estimou valores de andamento para cada uma destas janelas. O valor de andamento com maior ocorrência foi escolhido como valor de andamento global.

Os resultados foram apresentados e discutidos no Capítulo 4. Os nove experimentos realizados com os diferentes tipos de escalograma se comportaram de maneira similar, apresentando resultados próximos. As curvas de treinamento e validação foram observadas e percebeu-se que os escalogramas gerados a partir de 40 níveis de escala apresentaram comportamento mais estável enquanto que os escalogramas com 6 níveis de escala apresentaram as curvas mais instáveis. Isto também é percebido ao se analisar o resultado final dos conjuntos de validação e teste para os experimentos, e ao compará-los chegou-se a conclusão que os modelos treinados pelos escalogramas Morl_40 e Shan_40 apresentaram os melhores resultados.

Após as definições dos melhores parâmetros de geração dos escalogramas, foi analisada a variação do parâmetro DA, responsável pela inserção do aumento artificial de dados no modelo. Percebeu-se que quanto mais exemplos passaram pelo processo de aumento de dados, melhor era o resultado no conjunto "Combinados" que foi utilizado como validação nesta etapa. Desta forma, concluiu-se que para o problema da estimativa de andamento musical deve-se utilizar variações de todos os exemplos, comprimindo e expandindo os escalogramas, de forma a expor o modelo a maior quantidade de diferentes exemplos possível. Esta afirmação é válida para os experimentos realizados com 50 épocas, pois este processo de aumento de dados tem potencial de gerar uma grande quantidade de dados redundantes.

Para a avaliação final do modelo e comparação com o estado da arte foi aplicada a previsão por janelas aleatórias. Percebeu-se que ao aplicar esta técnica, os resultados para acurácia 2 aumentaram aproximadamente 2% em ambos os modelos Morl_40 e Shan_40. Para a acurácia 0, os resultados dos modelos propostos foram inferiores ao estado da arte. Isto era previsível, pois houve uma limitação no intervalo de previsão da rede neural convolucional. Para bancos que a maioria dos exemplos estava no intervalo "*Sweet Octave*" os resultados se aproximaram ao estado da arte, como o Ballroom. Para acurácia 1, a análise do resultado é similar visto que nenhum múltiplo ou submúltiplo é considerado como acerto.

Os resultados mais significativos foram para a acurácia 2. O modelo foi treinado e definido visando atingir bons resultados para este tipo de acurácia. O modelo proposto Shan_40 atingiu uma acurácia 2 de 90,1% para o banco de dados "Combinados", se aproximando do estado da arte que é 93,6%. Destaca-se que ambos os modelos propostos superaram o estado da arte para o banco de dados "GiantSteps" , e o modelo Morl_40 atingiu o melhor resultado para acurácia 2, 92,6%. Porém, para o conjunto de dados "SMC Mirum" os resultados foram muito abaixo ao estado da arte. Isto ocorreu devido à pouca representabilidade de músicas clássicas, sem instrumentos percussivos, nos bancos de dados de treinamento.

Por fim, pode-se concluir que os modelos propostos foram capazes de estimar o andamento musical com resultados similares ao estado da arte. A utilização dos escalogramas wavelet como representação dos sinais de áudio se mostrou vantajosa visto que é possível realizar ajustes nos parâmetros de geração de forma a investigar uma representação que se comporte melhor para um determinado tipo de problema. Porém, para o problema da estimativa de andamento musical, os escalogramas não apresentaram melhorias significativas quando comparado aos espectrogramas, que são a forma de representação mais

utilizada. A técnica de aumento artificial de dados proposta se mostrou eficaz, comprovado pelos resultados da variação da quantidade de exemplos que eram submetidos a esta alteração. Da mesma forma, a técnica de janelas aleatórias para estimativa de andamento global se mostrou importante para melhoria dos resultados finais do modelo proposto, para a acurácia 2.

5.2 SUGESTÕES PARA TRABALHOS FUTUROS

Após a conclusão deste trabalho, algumas ideias para melhoria dos resultados para o problema de estimativa de andamento musical foram visualizadas. A principal delas é o desenvolvimento de um "decisor de oitavas" que poderia ser uma etapa final do modelo proposto ou incorporado à arquitetura da rede. Sua função seria decidir se a previsão do modelo foi feita na oitava certa ou não. Desta forma, seria possível observar melhorias dos resultados de acurácia 0 e acurácia 1. Classificadores híbridos poderiam ser utilizados, incorporando à rede neural convolucional outros classificadores como SVM (*Support Vector Machine*), *Random Forest*, HMM (*Hidden Markov Model*), etc.

Outra possibilidade de melhoria é investigar mais escalogramas gerados à partir de outros parâmetros. Percebeu-se que os escalogramas com mais níveis de escala apresentaram melhores resultados, por isso seria uma alternativa aumentar a quantidade de escalas dos escalogramas que atingiram os melhores desempenhos. Para isto, seria necessário alterar a camada de entrada da rede de forma a se adequar ao novo escalograma gerado. A arquitetura da rede também pode ser modificada e estruturas com mais camadas e mais conexões devem ser estudadas.

Com relação aos bancos de dados, é necessário a criação de novos bancos para treinamento da rede e também uma verificação dos bancos de dados de avaliação. Principalmente com relação ao banco de dados "SMC Mirum" que não apresenta bons resultados em nenhum dos principais trabalhos da área. Percebe-se que para este banco a maior parte dos valores de andamento se concentra em valores menores do que 80 BPM, o que é incomum em termos práticos.

Como a maioria dos bancos de dados de treinamento possuem músicas que possuem instrumentos percussivos na sua composição, uma alternativa eficiente de aumento artificial de dados seria realizar uma separação entre instrumentos percussivos e instrumentos harmônicos, criando diferentes exemplos para o treinamento da rede. Esta separação poderia ser feita utilizando ferramentas já existentes na área de MIR, ou uma rede poderia ser treinada com este objetivo.

Os pontos expostos podem indicar caminhos a serem seguidos para prosseguimento da pesquisa sobre o problema da estimativa de andamento musical, que é um problema que permanece em aberto com necessidade de melhorias nos resultados nos diferentes tipos de acurácia. Em 2020, Schreiber *et al.* [12] alertou que durante muitos anos os pesquisadores tem trabalhado na estimativa de andamento musical com pouca conexão com a indústria musical. Algumas métricas foram utilizadas no passado e não foram adotadas pela indústria. Por isso é importante que os métodos de avaliação também sejam estudados e, eventualmente, novas métricas de avaliação possam ser desenvolvidas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. Lerch, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. 07 2012.

[2] A. C. L. Fernandes Júnior, *Contribuições ao Problema de Extração de Tempo Musical*. Campinas, São Paulo, Brasil: Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação - Tese (Doutorado), 2015.

[3] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. USA: Cambridge University Press, 1st ed., 2019.

[4] F. Chollet, *Deep Learning with Python*. Shelter Island, NY, USA: Manning, Nov. 2017.

[5] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017.

[6] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[7] H. Schreiber and M. Muller, "*A Single-Step Approach to Musical Tempo Estimation Using a Convolutional Neural Network*," 2018.

[8] C. ISMIR, "International Society for Music Information Retrieval (ISMIR)." Disponível em: `ismir.net`. Acesso em: 20 dezembro, 2022.

[9] S. Böck, F. Krebs, and G. Widmer, "*Accurate Tempo Estimation based on Recurrent Neural Networks and Resonating Comb Filter*," 2015.

[10] M. Y. Goto, Masataka, "*A Beat Tracking System for Acoustic Signals of Music*," *In Proceedings of the Second ACM International Conference on Multimedia*, vol. 0, p. 365–372, 1994.

[11] E. D. Scheirer, "*Tempo and Beat Analysis of Acoustic Musical Signals*," *The Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.

[12] H. Schreiber, J. Urbano, and M. Müller, "*Music Tempo Estimation: Are We Done Yet?*," *Transactions of the International Society for Music Information Retrieval*, vol. 3, p. 111, 08 2020.

[13] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "*An Experimental Comparison of Audio Tempo Induction Algorithms*," no. 14(5), p. 1832– 1844, 2006.

[14] J. R. Zapata and E. Gómez, "*Comparative Evaluation and Combination of Audio Tempo Estimation Approaches*," 2011.

[15] S. Bock and M. Schedl, "*Enhanced Beat Tracking With Context-Aware Neural Networks*," 2011.

[16] A. Gkiokas and V. Katsouros, "*Convolutional Neural Networks for Real-Time Beat Tracking: A Dance Robot Application*," 2011.

[17] K. Choi, G. Fazekas, and M. Sandler, "*Automatic Tagging Using Deep Convolutional Neural Networks*," 06 2016.

[18] Z. Nasrullah and Y. Zhao, "*Music Artist Classification With Convolutional Recurrent Neural Networks*," 2019.

[19] K. Palanisamy, D. Singhania, and A. Yao, "*Rethinking CNN Models for Audio Classification*," 2020.

[20] H. Chen, P. Zhang, H. Bai, Q. Yuan, X. Bao, and Y. Yan, "*Deep Convolutional Neural Network With Scalogram for Audio Scene Modeling*," 2018.

[21] M. Souza, P. Moura, and J.-P. Briot, "*Tempo Estimation Via Neural Networks - A Comparative Analysis*," pp. 17–24, 2021.

[22] X. Sun, Q. He, Y. Gao, and W. Li, "*Musical Tempo Estimation Using a Multi-scale Network*," 2021.

[23] E. Quinton, "*Equivariant Self-Supervision for Musical Tempo Estimation*," 2022.

[24] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*. Boston: McGraw Hill, fourth ed., 2002.

[25] B. P. Lathi and Z. Ding, *Sistemas de Comunicações Analógicos e Digitais Modernos*. LTC, 4 ed., 2012.

[26] B. P. Lathi, *Linear Systems and Signals*. USA: Oxford University Press, Inc., 2nd ed., 2009.

[27] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk, "*Beyond Nyquist: Efficient Sampling of Sparse Bandlimited Signals*," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 520–544, 2010.

[28] A. W. A.V. Oppenheim, *Signal Systems*. Upper Saddle River, NJ, EUA: Prentice-Hall, 2.ed ed., 1999.

[29] M. Domingues, O. Mendes, M. Kaibara, V. Menconi, and E. Bernardes, "*Explorando a Transformada Wavelet Contínua*," *Revista Brasileira de Ensino de Física*, vol. 38, 09 2016.

[30] J. M. P. Lima e Silva, *Aplicação da Transformada Wavelet Packet e Redes Neurais Artificiais para Monitoramento de Condição de Motores de Indução*. Universidade Federal de Pernambuco - Tese(doutorado), 2017.

[31] E. Large and C. Palmer, "*Perceiving Temporal Regularity in Music*," *Cognitive Science*, vol. 26, pp. 1–37, 01 2002.

[32] M. Wright, *The Shape of an Instant: Measuring and Modeling Perceptual Attack Time with Probability Density Functions*. PhD thesis, 01 2008.

[33] G. W. Cooper and L. B. Meyer, *The Rhythmic Structure of Music*. Chicago, IL; Toronto, ON: The University of Chicago Press, 1st ed., 1960.

[34] L. A. Magrini, "*Funções Wavelet e Transformada Wavelet Contínua: Representação Simultânea nos Domínios do Tempo e da Frequência*," *C.Q.D. - Revista Eletrônica Paulista de Matemática*, vol. 19, dez. 2020.

[35] G. Lee, R. Gommers, F. Waselewski, K. Wohlfahrt, and Aaron, "*PyWavelets: A Python Package for Wavelet Analysis*," *Journal of Open Source Software*, vol. 4, p. 1237, 04 2019.

[36] T. Gao and V. Jojic, "*Degrees of Freedom in Deep Neural Networks*," *CoRR*, vol. abs/1603.09260, 2016.

[37] T. Götz, S. Göb, S. Sawant, F. Erick, T. Wittenberg, C. Schmidkonz, A. Tomé, E. Lang, and A. Ramming, "*Number of Necessary Training Examples for Neural Networks with Different Number of Trainable Parameters*," *Journal of Pathology Informatics*, vol. 13, 07 2022.

[38] S. Haykin, "*Neural Networks: Principles and Practice*," *Bookman*, 2001.

[39] R. Venkatesan and B. Li, *Convolutional Neural Networks in Visual Computing: A Concise Guide*. CRC Press Taylor & Francis Group, 2018.

[40] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "*Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*," 2016.

[41] D. Kingma and J. Ba, "*Adam: A Method for Stochastic Optimization*," *International Conference on Learning Representations*, 12 2014.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "*Dropout: A Simple Way to Prevent Neural Networks from Overfitting*," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 06 2014.

[43] UNC Vision Lab, "*Large Scale Visual Recognition Challenge 2016 (ILSVRC2016).*" Disponível em: `http://image-net.org/challenges/LSVRC/2016/index`. Acesso em: 07 março, 2021.

[44] H. Schreiber and M. Müller, "*A Post-Processing Procedure for Improving Music Tempo Estimates Using Supervised Learning*," 10 2017.

[45] C. Raffel, "*Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*," 2016.

[46] Á. Faraldo, S. Jordà, and P. Herrera, "*A Multi-Profile Method for Key Estimation in EDM*," 2017.

[47] G. P. Ugo Marchand, "*The Extended Ballroom Dataset*," 2016.

[48] G. Peeters and J. Flocon-Cholet, "*Perceptual Tempo Estimation using GMM Regression*," pp. 45–50, 11 2012.

[49] P. Knees, Á. Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. L. Goff, "*Two Data Sets for Tempo Estimation and Key Detection in Electronic Dance Music Annotated from User Corrections*," pp. 364–370, 27/10/2015 2015.

[50] G. Tzanetakis and P. Cook, "*Musical Genre Classification of Audio Signals*," *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 293 – 302, 08 2002.

[51] S. Hainsworth, "*Techniques for the Automated Analysis of Musical Audio*," 06 2004.

[52] A. Holzapfel, M. Davies, J. Zapata, J. Oliveira, and F. Gouyon, "*Selective Sampling for Beat Tracking Evaluation*," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, pp. 2539–2548, 11 2012.

[53] S. Dieleman and B. Schrauwen, "*End-to-End Learning for Music Audio*," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 6964–6968, 05 2014.

[54] J. Lee, J. Park, K. Kim, and J. Nam, "*Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms*," 07 2017.

[55] F.-H. Wu and S.-H. Lai, "*Audio Tempo Estimation Method Improved by Rhythm Pattern and Data Augmentation*," pp. 779–784, 04 2019.

[56] A. Copiaco, C. Ritz, S. Fasciani, and N. Abdulaziz, "*Scalogram Neural Network Activations with Machine Learning for Domestic Multi-channel Audio Classification*," 12 2019.

[57] Z. Ren, K. Qian, Z. Zhang, V. Pandit, A. Baird, and B. Schuller, "*Deep Scalogram Representations for Acoustic Scene Classification*," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, 05 2018.

[58] H. Abbasi, L. Bennet, A. Gunn, and C. Unsworth, "*2D Wavelet Scalogram Training of Deep Convolutional Neural Network for Automatic Identification of Micro-Scale Sharp Wave Biomarkers in the Hypoxic-Ischemic EEG of Preterm Sheep*," *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2019, pp. 1825–1828, 07 2019.

[59] N. Kumar and R. Kumar, "*Wavelet Transform-Based Multipitch Estimation in Polyphonic Music*," *Heliyon*, vol. 6, p. e03243, 01 2020.

[60] A. Azizi, K. Faez, A. Rezaeian, and S. Quchani, "*Automatic Music Transcription Based on Wavelet Transform*," pp. 158–165, 09 2009.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "*Deep Residual Learning for Image Recognition*," pp. 770–778, 06 2016.

[62] K. Simonyan and A. Zisserman, "*Very Deep Convolutional Networks for Large-Scale Image Recognition*," *arXiv 1409.1556*, 09 2014.

[63] F. Chollet, "*Xception: Deep Learning with Depthwise Separable Convolutions*," 2017.

[64] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "*Rethinking the Inception Architecture for Computer Vision*," 06 2016.

[65] M. Ferreira, D. Corrêa, L. Nonato, and R. Mello, "*Designing Architectures of Convolutional Neural Networks to Solve Practical Problems*," *Expert Systems with Applications*, vol. 94, 10 2017.

[66] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "*Going Deeper With Convolutions*," *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 06 2015.