# Aplicação de uma Rede Neural Convolucional para a Identificação de Acordes Musicais

**Jessé Santana Veloso**[1], **Enrique Augusto da Roza**[1], **Murilo Falleiros Lemos Schmitt**[1]

[1]Departamento de Ciência da Computação (DECOMP)
Universidade Estadual do Centro-Oeste (UNICENTRO)
Caixa Postal 730 – 85.040-167 – Guarapuava – PR – Brasil

jessesantanaveloso23@gmail.com,{enriqueroza, mschmitt}@unicentro.br

***Abstract.*** *This study proposes the use of Convolutional Neural Networks (CNNs) for automatic musical chord recognition from audio files, utilizing the Jazznet dataset containing piano samples. The CNN architecture processes spectral representations (spectrograms) through two specialized models: one for root note identification and another for chord type classification. Results demonstrate 78% accuracy in the main task, while revealing challenges in distinguishing between similar harmonic categories. The research highlights the potential synergy between music theory and artificial intelligence, providing foundations for developing educational and technological tools in the musical domain.*

***Resumo.*** *Este estudo propõe o uso de Redes Neurais Convolucionais (CNNs) para identificação automática de acordes musicais a partir de arquivos de áudio, utilizando o conjunto de dados Jazznet com amostras de piano. A arquitetura CNN processa representações espectrais (espectrogramas) através de dois modelos especializados: um para identificação da nota tônica e outro para classificação do tipo de acorde. Os resultados demonstram 78% de acurácia na tarefa principal, embora revelem desafios na distinção entre categorias harmônicas similares. A pesquisa evidencia a potencial sinergia entre teoria musical e inteligência artificial, oferecendo bases para o desenvolvimento de ferramentas educacionais e tecnológicas no domínio musical.*

**Palavras-chave**: Reconhecimento de Acordes, Redes Neurais Convolucionais, Processamento de Áudio, Inteligência Artificial, Análise Musical.

## 1. Introdução

A música tem sido, ao longo da história, uma das formas mais universais de expressão humana, transcendendo barreiras culturais e linguísticas [Meyer, 1956]. Nesse contexto, os acordes surgem como elementos fundamentais da linguagem musical, consistindo em conjuntos de notas tocadas simultaneamente que, quando organizados em sequências temporais - as chamadas progressões harmônicas -, formam a espinha dorsal da estrutura harmônica de uma obra [Schoenberg, 1978]. A identificação e classificação precisa desses acordes constituem uma etapa fundamental no entendimento e no aprendizado necessário para a análise musical, porém demandam expertise técnica especializada que, frequentemente, representa um desafio mesmo para músicos qualificados [Parncutt, 2007].

Diante dessa complexidade, o desenvolvimento de sistemas automatizados para reconhecimento de acordes mostra-se como uma solução promissora para democratizar o aprendizado e o acesso a essa análise. Tal abordagem contribui para tornar mais acessível e eficiente o processo de análise harmônica, beneficiando músicos, educadores e desenvolvedores de tecnologias musicais.

O *Music Information Retrieval* (MIR) é uma área interdisciplinar que estuda métodos computacionais para analisar, recuperar e organizar informações musicais a partir de sinais de áudio, partituras ou metadados [Mauch et al., 2009]. No âmbito do MIR, o reconhecimento automático de acordes tem ganhado destaque como área de pesquisa relevante [Fujishima, 1999], oferecendo representações

intermediárias valiosas para aplicações que vão desde a segmentação de estruturas musicais até a comparação automática entre composições [Mauch et al., 2009]. O presente estudo alinha-se a essa linha de investigação, propondo a aplicação de técnicas avançadas de Inteligência Artificial (IA), com ênfase em *Convolutional Neural Networks* (CNNs), para a classificação automática de acordes a partir de arquivos de áudio.

As CNNs, inicialmente desenvolvidas para processamento de imagens [LeCun et al., 1998], demonstraram adaptabilidade ao domínio do processamento de sinais de áudio [Dieleman and Schrauwen, 2014]. Para a abordagem deste trabalho, foi implementada uma arquitetura CNN dividida em dois tipos de modelo, um para a identificação de acordes (maiores, menores, etc.) e outro para suas constituintes notas fundamentais, utilizando, para isso, um conjunto de dados que possui amostras de piano devidamente anotadas. O sistema opera transformando sinais de áudio em representações espectrais que destacam características harmônicas, permitindo ao modelo aprender padrões musicais de forma indutiva [Humphrey et al., 2012]. O sistema foi treinado em amostras de piano devidamente anotadas, demonstrando:

- Acurácia de 78% na classificação de acordes isolados;
- Robustez na identificação de notas fundamentais;
- Dificuldades com acordes estruturalmente similares (e.g., sus2 vs sus4);
- Sensibilidade a variações timbrísticas e dinâmicas.

A relevância desta pesquisa situa-se na interface entre pedagogia musical e tecnologia, buscando desenvolver ferramentas computacionais que possam auxiliar tanto estudantes iniciantes quanto músicos experientes em seu processo de aprendizagem e criação [Chordia and Rae, 2007]. Ao integrar conhecimentos da teoria musical com técnicas avançadas de aprendizado de máquina, pretende-se contribuir para o desenvolvimento de soluções tecnológicas que tornem o conhecimento musical mais acessível, promovendo uma sinergia entre arte e inovação computacional.

Este trabalho está organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica, abordando conceitos de teoria musical e aprendizado profundo. Na Seção 3, são discutidos os trabalhos correlatos que embasaram esta pesquisa. A Seção 4 detalha o processo de desenvolvimento. A Seção 5 inclui a metodologia e métricas adotadas. Por fim, a Seção 6 apresenta os resultados.

## 2. Fundamentação Teórica

Nesta seção, apresentam-se os conceitos fundamentais que embasam este estudo. A Seção 2.1 aborda os principais aspectos da teoria musical e dos acordes musicais. Na Seção 2.2, discute-se o Aprendizado Profundo, enquanto a Seção 2.3 apresenta o conceito de CNNs.

### 2.1. Teoria Musical

Historicamente, a música foi transmitida por meio da tradição oral, até que se tornou necessária a criação de um sistema de notação capaz de representar graficamente os sons musicais [Med, 1996]. Embora a diversidade sonora seja ampla, apenas sete notas são suficientes para descrever a base da música ocidental: dó, ré, mi, fá, sol, lá e si. Esse sistema silábico foi introduzido por Guido d'Arezzo, por volta do século XI, como uma ferramenta didática para o ensino do canto [Med, 1996]. Posteriormente, a notação foi simplificada com a atribuição de letras às notas, originando o sistema alfabético atualmente utilizado: C (dó), D (ré), E (mi), F (fá), G(sol), A (lá) e B (si) [Med, 1996].

Além das sete notas naturais, a música utiliza acidentes para representar variações de altura. Os acidentes são símbolos que modificam a entonação das notas, criando alterações ascendentes ou descendentes [Levine, 2011]. Se a nota for elevada em um semitom, é utilizada a nomenclatura sustenido (♯) após a nota; se for abaixada em um semitom, é utilizado o bemol (♭).

A compreensão das notas musicais e acidentes é essencial para o estudo da teoria musical, servindo como base para a construção de escalas, harmonias e, principalmente, de acordes musicais.

Na próxima seção, será explorado como essas formalizações são necessárias para a formação dessas estruturas.

### 2.1.1. Acordes Musicais

De acordo com [Med, 1996], uma nota é um monossílabo que representa um som regular, enquanto os acordes são definidos como a sonoridade resultante da emissão simultânea de três ou mais sons. Quando dois sons são tocados simultaneamente, são denominados díades ou intervalos; três sons formam tríades, e quatro sons simultâneos compõem as chamadas tétrades [Levine, 2011].

Uma escala musical é um conjunto de notas ordenadas de acordo com sua frequência [Adegbija, 2023]. Um acorde é constituído por, no mínimo, três notas fundamentais, dentre as quais uma é responsável por definir sua nomenclatura: a tônica [Med, 1996]. A nota tônica representa o ponto de partida de uma escala musical específica, sendo ela essencial para determinar a sequência de notas que formam o acorde. A Tabela 1 apresenta um exemplo da escala maior, também conhecida como escala jônica.

**Tabela 1. Escalas Maiores com Tons (T) e Semitons (ST). [Med, 1996]**

| **Escala** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|
| **Intervalos** | T | T | ST | T | T | T | ST |
| C | D | E | F | G | A | B | C |
| C♯ | D♯ | E♯ | F♯ | G♯ | A♯ | B♯ | C♯ |
| D | E | F♯ | G | A | B | C♯ | D |
| E♭ | F | G | A♭ | B♭ | C | D | E♭ |
| E | F♯ | G♯ | A | B | C♯ | D♯ | E |
| F | G | A | B♭ | C | D | E | F |
| F♯ | G♯ | A♯ | B | C♯ | D♯ | E♯ | F♯ |
| G | A | B | C | D | E | F♯ | G |
| A♭ | B♭ | C | D♭ | E♭ | F | G | A♭ |
| A | B | C♯ | D | E | F♯ | G♯ | A |
| B♭ | C | D | E♭ | F | G | A | B♭ |
| B | C♯ | D♯ | E | F♯ | G♯ | A♯ | B |

Cada acorde deriva diretamente de sua escala correspondente, sendo construído através da sobreposição de intervalos específicos a partir da nota fundamental. A Tabela 2 detalha a classificação dos intervalos musicais fundamentais utilizados neste estudo. Complementarmente, a Tabela 3 apresenta uma análise dos principais tipos de acordes, exemplificando sua construção intervalar e nomenclatura padrão no contexto da escala de Dó maior [Levine, 2011].

## 2.2. Aprendizado profundo

O aprendizado profundo é um subcampo do aprendizado de máquina que se baseia no uso de Redes Neurais Artificiais (RNAs) com múltiplas camadas de processamento [LeCun et al., 2015]. Essa abordagem tem se mostrado útil quando utilizada em contextos musicais e de processamento de áudio [Hershey et al., 2017]. Cada camada é responsável por extrair representações progressivamente mais abstratas dos dados, permitindo que o modelo capture relações complexas e não lineares. Essa abordagem é inspirada na estrutura do córtex visual humano, onde os neurônios são organizados em camadas hierárquicas que processam informações sensoriais de maneira gradativa [Zhang et al., 2023].

Existem diversos tipos de RNAs profundas, cada uma projetada para lidar com tipos específicos de problemas. Por exemplo, os Transformers são especialmente eficazes no tratamento

**Tabela 2. Intervalos na escala de C. [Levine, 2011]**

| Intervalo | Fórmula | Exemplo |
|---|---|---|
| 2ª menor (min2) | 1 2♭ | C D♭ |
| 2ª maior (maj2) | 1 2 | C D |
| 3ª menor (min3) | 1 3♭ | C E♭ |
| 3ª maior (maj3) | 1 3 | C E |
| 4ª justa (perf4) | 1 4 | C F |
| Trítono (tritone) | 1 4♯ | C F♯ |
| 5ª justa (perf5) | 1 5 | C G |
| 6ª menor (min6) | 1 6♭ | C A♭ |
| 6ª maior (maj6) | 1 6 | C A |
| 6ª aumentada (aug6) | 1 6♯ | C A♯ |
| 7ª maior (maj7) | 1 7 | C B |
| Oitava (octave) | 1 8 | C C’ |

**Tabela 3. Acordes na escala de C [Levine, 2011]**

| Tipo | Fórmula | Exemplo | Nome |
|---|---|---|---|
| Maior (maj) | 1 3 5 | C E G | C |
| Menor (min) | 1 3♭ 5 | C E♭ G | Cm |
| Aumentado (aug) | 1 3 5♯ | C E G♯ | Caug |
| Diminuto (dim) | 1 3♭ b5 | C E♭ G♭ | Cdim |
| Sus2 (sus2) | 1 2 5 | C D G | Csus2 |
| Sus4 (sus4) | 1 4 5 | C F G | Csus4 |
| Dim7 (dim7) | 1 3♭ 5♭ 5 | C E♭ G♭ A | Cdim7 |
| Maior7 (maj7) | 1 3 5 7 | C E G B | Cmaj7 |
| Menor7 (min7) | 1 3♭ 5 7♭ | C E♭ G B♭ | Cm7 |
| Min7b5 | 1 3♭ 5♭ 7♭ | C E♭ G♭ B♭ | Cm7b5 |
| 7ª (7th) | 1 3 5 7♭ | C E G B♭ | C7 |
| 6ª (6th) | 1 3 5 6 | C E G A | C6 |

de dados sequenciais, como séries temporais e processamento de linguagem natural, devido à sua capacidade de manter informações contextuais ao longo do tempo [Vaswani et al., 2017]. Já as CNNs são particularmente adequadas para o processamento de dados com estrutura espacial, como imagens e áudios — foco principal deste trabalho [Goodfellow et al., 2016].

## 2.3. Rede Neural Convolucional

Conforme destacado por [Goodfellow et al., 2016], as CNNs foram originalmente desenvolvidas para processamento de imagens, apresentando uma arquitetura especializada que permite o aprendizado hierárquico de características visuais. No entanto, sua aplicação tem se mostrado igualmente eficaz no domínio do áudio, especialmente quando o sinal sonoro é representado em formatos espectrais como espectrogramas, *Mel-Frequency Cepstral Coefficients* ou representações tempo-frequenciais [Hershey et al., 2017].

As CNNs aplicadas ao áudio mantêm sua arquitetura característica, como exemplificado na Figura 1 composta por camadas de convolução, *pooling* (camada de agrupamento) e totalmente conectada [LeCun et al., 2015] descritas a seguir.

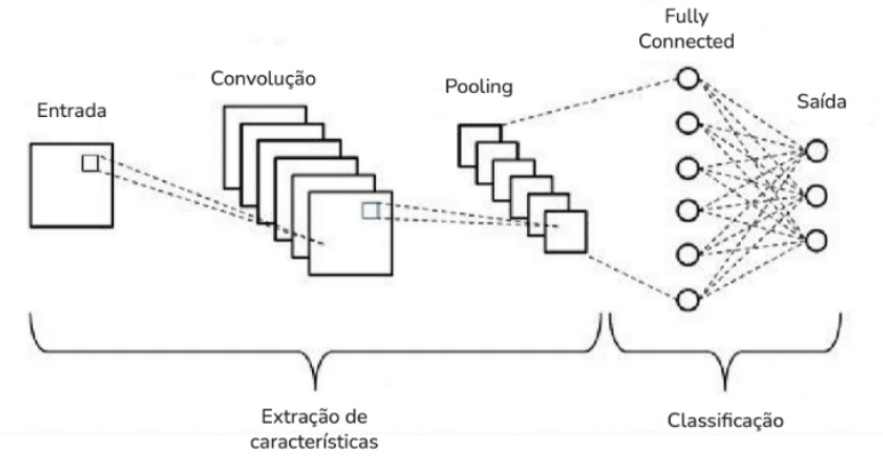


**Figura 1. Arquitetura da CNN. Adaptado de [Sakurai, 2024].**

### 2.3.1. Camada Convolucional

A camada convolucional constitui o núcleo das CNNs, empregando operações matemáticas de convolução. Nesse processo, filtros (ou *kernels*) são deslizados sobre toda a extensão da imagem, calculando-se o produto escalar entre os pesos do filtro e os valores dos pixels da região sobreposta na entrada [Zeiler and Fergus, 2014], como exemplificado na Figura 2. Tecnicamente, cada valor das características de saída da CNN (*feature map*) é obtido pela soma ponderada (produto de convolução) entre o *kernel* e a região local da entrada, seguida pela aplicação de uma função de ativação, como ReLU (*Rectified Linear Unit*). Esse processo permite a extração de características de baixo nível (como bordas, gradientes de intensidade e texturas) nas camadas iniciais, enquanto filtros em camadas profundas capturam atributos semânticos complexos [Zeiler and Fergus, 2014]. Além disso, também permite reduzir o tamanho das características de saída da CNN (*feature map*) juntamente com a camada de *pooling*.

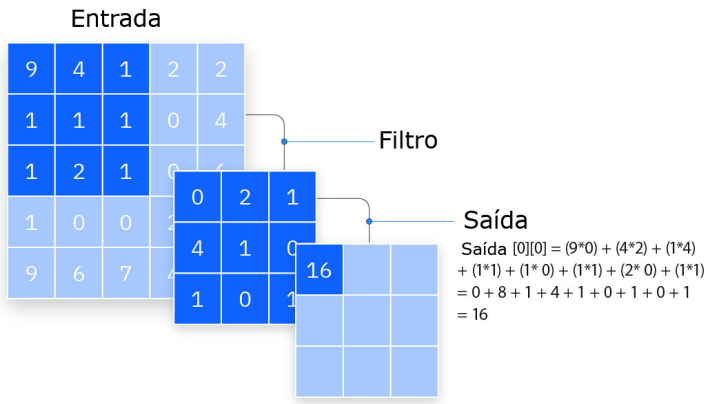


**Figura 2. Processo de convolução através da soma ponderada. Adaptado de [IBM, 2024]**

### 2.3.2. Camada de *Pooling*

A camada de *pooling* tem como objetivo diminuir progressivamente a dimensão espacial das representações internas, e subsequentemente reduz o tamanho de entrada para as próximas camadas. Essa redução é alcançada por meio da consolidação de informações através das camadas, ampliando o campo receptivo de cada neurônio oculto [Zhang et al., 2023]. Esse mecanismo é necessário para aplicações que exigem uma compreensão abrangente da imagem, como a identificação de objetos em cenários complexos e detecção de picos de frequência em arquivos de áudio, já que possibilita a construção de representações hierárquicas sem perder as propriedades vantajosas das camadas convolucionais [Zhang et al., 2023].

Existem diferentes tipos de operações de *pooling*, sendo as mais utilizadas o *Max Pooling* e o *Average Pooling*. Neste trabalho, ambas as técnicas foram empregadas em etapas distintas da rede, conforme as necessidades de cada camada.

O *Max Pooling*, que seleciona o maior valor dentro de uma janela definida pelo *stride* (Figura 3), foi utilizado em camadas onde o objetivo era destacar características dominantes, como picos de intensidade ou padrões locais relevantes, como utilizado no estudo de [Nadar et al., 2019]. Essa abordagem aumenta a robustez da rede a variações espaciais e preserva *features* mais discriminativas, sendo especialmente útil em tarefas de detecção de padrões complexos [Yani et al., 2019].

Já o *Average Pooling*, que calcula a média dos valores na janela, foi aplicado em estágios onde a suavização das *features* era desejável, como na redução de ruídos ou na transição entre camadas densas, como utilizado por [Huang et al., 2017]. Essa técnica ajuda a manter informações contextuais

sem amplificar *outliers*, sendo vantajosa em cenários que exigem generalização espacial [Yani et al., 2019].

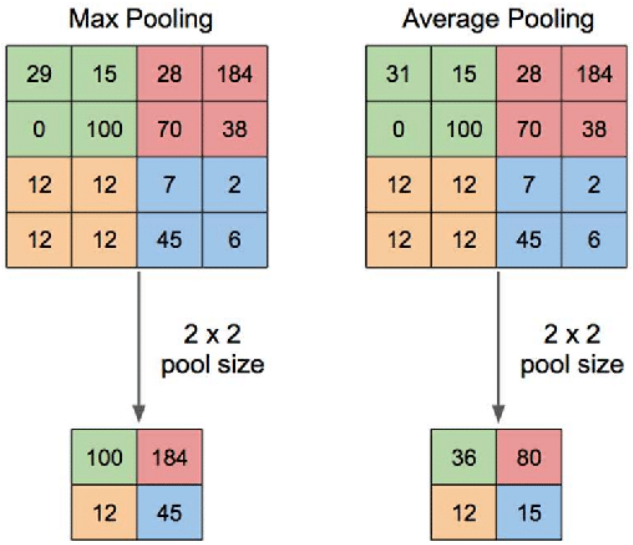


**Figura 3. Operação de *Max Pooling* e *Average Pooling* [Almeida and Barros, 2020].**

### 2.3.3. Camada Totalmente Conectada

Nessa camada, todos os neurônios de uma camada estabelecem conexões diretas com cada unidade da camada subsequente [Goodfellow et al., 2016]. Sua principal função é processar as características extraídas nas etapas anteriores para realizar a classificação. Um aspecto relevante é que a quantidade de neurônios na última camada densa corresponde exatamente ao número de categorias a serem preditas [Nadar et al., 2019]. Adicionalmente, em cenários de classificação multiclasse, é comum empregar a função de ativação *softmax* na camada final. Essa função transforma um vetor de saída em uma distribuição probabilística, onde a soma de todos os valores resulta em 1, conforme definido pela Equação 1 [Zhang et al., 2023]:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{para } i = 1, \ldots, K \text{ e } z = (z_1, \ldots, z_K) \in \mathbb{R}^K \tag{1}$$

onde:

- $K$: O número total de classes ou categorias possíveis. Representa o número de elementos no vetor $z$, ou seja, o número de classes de saída no problema de classificação;
- $\sigma(z)_i$: Representa a saída da função *softmax* para a $i$-ésima classe. Ou seja, $\sigma(z)_i$ é a probabilidade associada à classe $i$, dada a entrada $z$;
- $z$: Um vetor $z = (z_1, z_2, \ldots, z_K)$ contendo os valores de entrada para a função. Esses valores geralmente vêm da última camada de uma rede neural (*logits*). O vetor $z$ pode ter qualquer dimensão $K$;
- $z_i$: O valor específico no vetor $z$ para a $i$-ésima classe. Cada $z_i$ é um *logit* ou a pontuação não normalizada associada à classe $i$;
- $e^{z_i}$: A exponenciação do valor $z_i$, que garante que as saídas da função *softmax* sejam positivas. A função exponencial é usada para amplificar as diferenças entre os valores $z_i$, destacando classes com maior pontuação;
- $\sum_{j=1}^{K} e^{z_j}$: O denominador da equação, representando a soma das exponenciais de todos os valores $z_j$ para $j = 1, \ldots, K$; Esse termo normaliza as probabilidades para que a soma de todas as probabilidades seja igual a 1.

## 3. Trabalhos Correlatos

Esta seção apresenta trabalhos relacionados a aplicações de IA na análise musical, com ênfase especial na identificação e classificação de acordes. A seleção dos trabalhos foi realizada através de buscas em plataformas como IEEE Xplore e *Google Scholar*, utilizando os seguintes termos de busca: "*chord recognition*","*music classification*","*CNN for music analysis*"e "*deep learning in music theory*". Os critérios de seleção priorizaram artigos com relevância no campo e ao escopo deste estudo.

Além dos estudos acadêmicos, destacam-se aplicações práticas como o Moises AI, uma plataforma desenvolvida por pesquisadores brasileiros que utiliza aprendizado de máquina para separação de fontes de áudio [AI, 2025]. A ferramenta emprega arquiteturas como U-Net e Conv-TasNet para isolar instrumentos e vocais, e também oferece funcionalidades avançadas, como a identificação automática de acordes.

O estudo de [Nadar et al., 2019] propõe uma arquitetura de CNN para o reconhecimento automático de acordes. Os experimentos compararam estratégias de classificação conjunta e separada do tipo de acorde e da nota fundamental (tônica), utilizando um ou dois modelos distintos. Além disso, o estudo introduziu um novo conjunto de dados sinteticamente gerados, composto por gravações isoladas de diversos instrumentos, facilitando futuras pesquisas em ACR com vocabulários estendidos.

Por fim, [Mauch et al., 2009] propõe o uso de CNNs para reconhecimento de acordes, superando as limitações do vetor PCP (*Pitch Class Profile*). Embora útil na análise harmônica, o PCP não representa adequadamente estruturas complexas. Para contornar isso, os autores transformam o PCP em uma matriz bidimensional (2D-PCP), permitindo que a CNN aprenda automaticamente as características e a classificação dos acordes.

Os trabalhos correlatos forneceram a base teórica e metodológica para este estudo, permitindo o uso de IA na extração de acordes com um vocabulário ampliado, superando limitações de representações tradicionais como o vetor PCP. Inspirado em [AI, 2025], este estudo focou na identificação de acordes, ao contrário da separação de fontes abordada na ferramenta original. Adotou-se uma abordagem semelhante à de [Nadar et al., 2019], com dois modelos distintos, mas aplicados a acordes isolados de piano, em vez de músicas completas. As contribuições teóricas de [Mauch et al., 2009] também foram fundamentais para justificar o abandono do vetor PCP em favor de modelos baseados em aprendizado profundo.

## 4. Desenvolvimento

Esta seção descreve o processo de desenvolvimento do trabalho, abordando o conjunto de dados, propriedades do som, a metodologia adotada e as tecnologias utilizadas ao longo do estudo.

### 4.1. Conjunto de dados

A base de dados utilizada neste trabalho foi a *Jazznet*[1] [Adegbija, 2023], disponível na plataforma *GitHub*[2]. Esse conjunto contém acordes, arpejos, escalas e progressões harmônicas baseadas em um piano de 88 teclas. A escolha dessa base deve-se à sua diversidade musical, pois permite extrair diferentes estruturas harmônicas com ampla variação tonal. Ao todo, a *Jazznet* reúne 5.525 acordes distintos, armazenados em arquivos de áudio no formato .wav, com duração de 3 segundos cada, totalizando aproximadamente 259 horas de áudio [Adegbija, 2023].

Os acordes foram gerados automaticamente por meio do algoritmo *Distance-Based Pattern Structures* (DBPS), que determina a estrutura de um acorde com base nas distâncias sucessivas entre as notas, segundo padrões predefinidos [Adegbija, 2023].

[1] https://github.com/tosiron/jazznet
[2] GitHub.com

Os dados contêm as seguintes informações:

- **Tônica**: Nota fundamental que dá nome ao acorde (ex.: A, B, C);
- **Oitava**: Faixa de altura em que o acorde está localizado no piano (ex.: 1, 2);
- **Tipo de acorde**: Classificação harmônica do acorde, indicando sua qualidade (ex.: maior, menor, diminuto);
- **Inversão**: Indica a disposição das notas do acorde, especificando qual nota está na posição mais grave (ex.: fundamental, 1ª inversão, 2ª inversão).

### 4.2. Divisão de Dados

O conjunto de dados *Jazznet* é dividido em três subconjuntos: treinamento (80%), validação (10%) e teste (10%), seguindo as melhores práticas em aprendizado de máquina [Zhang et al., 2023]. Essa divisão estratificada preserva a distribuição original das classes em cada subconjunto.

### 4.3. Propriedades do Som

O som é uma onda mecânica que se propaga pelo meio físico, caracterizada por variações periódicas na pressão do ar ou em outro meio elástico. Essa onda possui duas propriedades físicas fundamentais: frequência e amplitude. A frequência, medida em Hertz (Hz), determina a altura do som percebida pelo ouvido humano, enquanto a amplitude, medida em decibéis (dB), está relacionada à intensidade ou volume do som [Humphrey et al., 2012]. Na análise musical computacional, compreender essas propriedades é essencial para a extração de características relevantes dos sinais de áudio, pois elas influenciam diretamente a percepção e a representação dos eventos sonoros. Uma representação gráfica pode ser visualizada na Figura 4.

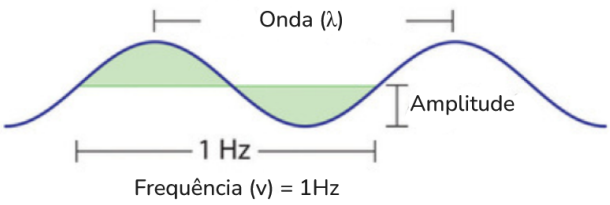


**Figura 4. Propriedades importantes da onda. Adaptado de [University of Wisconsin-Madison, 2023].**

### 4.4. Espectrograma

O espectrograma é uma representação visual e matemática do conteúdo espectral de um sinal de áudio ao longo do tempo. Ele é obtido por meio da aplicação da Transformada de Fourier de Curto Prazo (*Short-Time Fourier Transform* – STFT), que segmenta o sinal em janelas temporais e calcula a distribuição das frequências presentes em cada segmento [Smith, 2021]. Essa representação bidimensional exibe a intensidade das frequências (eixo vertical) em função do tempo (eixo horizontal), possibilitando a análise detalhada das características temporais e espectrais do áudio [Smith, 2021].

No presente estudo, foram utilizadas as matrizes da STFT como entrada para a CNN, geradas pela biblioteca TorchAudio [3], uma ferramenta integrada ao *framework* PyTorch [4] que oferece funções eficientes para o processamento de sinais acústicos. Isso facilita o processo de formação de *inputs* para a CNN, a Figura 5 exibe um exemplo gráfico de espectrograma gerado pela biblioteca.

[3]https://docs.pytorch.org/audio/stable/index.html
[4]https://pytorch.org/

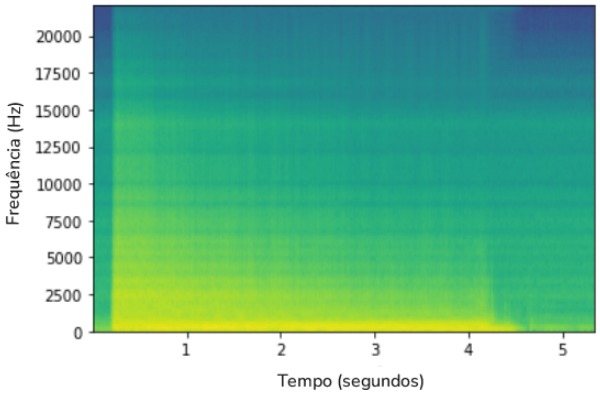


**Figura 5. Espectrograma do acorde de Dó maior.**

## 5. Metodologia

A metodologia deste estudo envolveu o treinamento e teste de modelos desenvolvidos, cuja arquitetura é exibida através da Figura 6. A arquitetura é composta por blocos convolucionais empilhados, uma abordagem amplamente utilizada com o propósito de facilitar a reutilização de componentes e a formação de arquiteturas mais profundas [Nadar et al., 2019]. Cada bloco da rede segue uma estrutura padronizada e encapsulada no bloco CNNblock [Nadar et al., 2019]. Esse bloco é uma unidade modular composta por quatro operações sequenciais:

- **Conv2D**: Aplica filtros convolucionais bidimensionais com kernel de tamanho 3×3 e *padding* de 1 — ou seja, adiciona bordas de zeros ao redor da entrada para preservar suas dimensões espaciais após a convolução, de forma a preservar as dimensões espaciais da entrada;
- **MaxPool2D**: Reduz a resolução espacial das características extraídas pela convolução por meio de uma operação de máxima agregação com janela 2×2 e passo 2 — ou seja, a janela se desloca 2 pixels por vez, diminuindo pela metade as dimensões espaciais da entrada;
- **BatchNorm2D**: Normaliza a saída do MaxPool2D ao longo do *batch* — isto é, o conjunto de amostras processadas simultaneamente durante uma iteração do treinamento — e dos canais, estabilizando o processo de treinamento e acelerando a convergência;
- **ReLU**: Introduz não linearidade ao modelo, essencial para que a rede aprenda funções complexas. A ReLU é escolhida por sua simplicidade computacional e eficiência na propagação do gradiente.

A arquitetura organiza uma sequência de blocos CNNblock, seguidos por operações de agregação e classificação:

- **CNNBlock(32 → 32)**: Recebe entrada com 32 canais e aplica 32 filtros, resultando em saída de dimensão 32;
- **CNNBlock(32 → 64)**: Aumenta a profundidade para 64 canais, com saída de dimensão 64;
- **CNNBlock(64 → 128)**: Amplia a capacidade representacional para 128 canais;
- **GlobalAvgPool**: Após a extração hierárquica de características, aplica-se um Global Average Pooling que reduz cada mapa de ativação — representações espaciais que indicam a resposta dos filtros convolucionais a diferentes padrões na entrada — para um único valor, resultando em um vetor de características com dimensão igual ao número de canais de saída do último bloco convolucional;
- **Flatten**: Após a etapa de *Global Average Pooling*, a saída da rede possui formato tridimensional por amostra. Para que essa saída seja compatível com a camada totalmente conectada (Linear), aplica-se uma operação de *flatten*, que transforma o *tensor* — uma estrutura de dados multidimensional que generaliza vetores e matrizes — em uma matriz bidimensional;

- **Dropout**: Introduz regularização durante o treinamento ao descartar aleatoriamente neurônios da camada anterior com uma probabilidade definida;
- **Totalmente Conectada (Linear)**: Responsável pela decisão final de classificação. Recebe o vetor condensado de características e produz uma saída de dimensão igual ao número de classes, correspondente à probabilidade de cada classe de acorde.

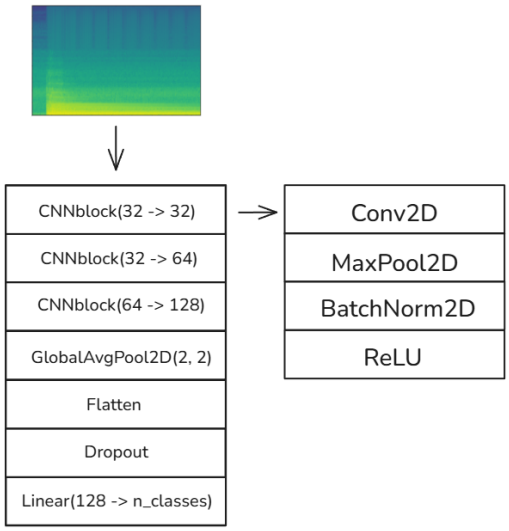


**Figura 6. Arquitetura aplicada da CNN.**

Neste estudo foram desenvolvidos três tipos de modelos com a mesma arquitetura, o Modelo 1 realiza a classificação considerando acordes e intervalos. O Modelo 2, considerando apenas acordes. E o Modelo 3 que realiza a classificação das notas tônicas. O Modelo 2 foi utilizado na modelagem final por conta da sua melhor acurácia obtida nos resultados.

A Figura 7 ilustra a estratégia adotada para a classificação de acordes. Utilizando o Modelo 3 para classificação das notas tônicas dos acordes e o Modelo 2 para classificação de acordes desconsiderando intervalos. Essa abordagem modular resulta em dois sistemas especializados, o que permite uma maior flexibilidade ao expandir o projeto, sem a necessidade de um número excessivo de classes de saída. Esse modelo modular também abre a possibilidade, em estudos futuros, de incluir a detecção da oitava da tônica e a identificação de inversões no acorde.

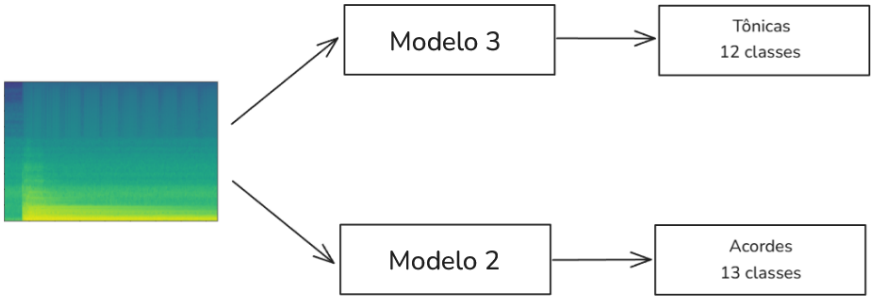


**Figura 7. Estratégia de modelagem utilizando os Modelos 2 e 3.**

O treinamento dos modelos utilizou o Adam (*Adaptive Moment Estimation*) [Mauch et al., 2009], um otimizador adaptativo que ajusta individualmente as taxas de aprendizagem para cada parâmetro com base nos momentos dos gradientes. Sua escolha deve-se à eficiência em problemas de aprendizado profundo, acelerando a convergência e reduzindo a sensibilidade para hiperparâmetros iniciais [Kingma and Ba, 2015].

As atualizações do Adam são calculadas a cada passo $t$ conforme:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \nabla_\theta J(\theta_t) \tag{2}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot (\nabla_\theta J(\theta_t))^2 \tag{3}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4}$$

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{5}$$

onde:

- $\alpha$: Taxa de aprendizagem (*learning rate*);
- $\beta_1, \beta_2$: Fatores de decaimento (tipicamente 0.9 e 0.999);
- $\epsilon$: Termo de estabilização numérica ($\approx 10^{-8}$);
- $\nabla_\theta J(\theta_t)$: Gradiente da função de perda $J$ em relação aos parâmetros $\theta$.

A arquitetura proposta baseia-se em uma rede neural convolucional composta por três blocos convolucionais sequenciais (*CNNBlocks*), seguidos por uma camada de *pooling* global, uma camada de regularização por *dropout* e uma camada totalmente conectada responsável pela saída final.

Nos modelos treinados, houve a variação de seus hiperparâmetros como: épocas, *batch size*, *learning rate* e *dropout*, para análise de desempenho do modelo. Ao longo de cada época de treinamento, o modelo processou os dados de entrada e produziu uma saída, que foi comparada com os rótulos reais por meio de uma função de perda *Cross-Entropy Loss* (função de entropia cruzada), a qual permite a classificação multiclasse. A partir dessa comparação, o erro foi calculado e utilizado para atualizar os pesos do modelo, empregando o algoritmo de retropropagação em conjunto com o otimizador Adam.

### 5.1. Tecnologias e Ferramentas

Para o desenvolvimento deste trabalho, foram utilizadas as ferramentas:

#### 5.1.1. Librosa

Librosa[5] é uma biblioteca de processamento de áudio em Python, amplamente utilizada para análise e manipulação de sinais musicais. Ela fornece funções para extração de características musicais, como espectrogramas, transformadas de Fourier e extração de características timbrísticas.

#### 5.1.2. PyTorch

PyTorch[6] é um framework de aprendizado profundo de código aberto, desenvolvido pela Meta AI. Ele permite o desenvolvimento de redes neurais complexas, com ênfase em velocidade e facilidade de uso. O PyTorch é utilizado para treinar modelos de aprendizado de máquina, e sua integração com a CUDA permite otimizar o treinamento utilizando unidades de processamento gráfico (GPUs).

[5]https://librosa.org/doc/latest/index.html
[6]https://pytorch.org/

### 5.1.3. NumPy

NumPy[7] é uma biblioteca fundamental para computação científica em Python. Ela oferece suporte para arrays multidimensionais e funções matemáticas eficientes. Em projetos de aprendizado de máquina e processamento de áudio, o NumPy é utilizado para manipular e realizar cálculos em grandes matrizes e vetores.

### 5.1.4. Scikit-learn

Scikit-learn[8] é uma biblioteca de aprendizado de máquina em Python que fornece ferramentas simples e eficientes para análise de dados. Ela oferece uma ampla gama de algoritmos para classificação e regressão, além de ferramentas para validação de modelos e pré-processamento de dados [Pedregosa et al., 2012].

### 5.1.5. Matplotlib

Matplotlib[9] é uma biblioteca de visualização de dados em Python, capaz de gerar gráficos de alta qualidade. Ela é amplamente usada para visualização de dados científicos, como espectrogramas, curvas de aprendizado e distribuições de características.

### 5.1.6. Visual Studio Code

Visual Studio Code[10] é um editor de código-fonte leve e altamente extensível, desenvolvido pela Microsoft. Ele oferece suporte a uma ampla gama de linguagens de programação e integrações com ferramentas de desenvolvimento.

### 5.1.7. CUDA

CUDA[11] (Compute Unified Device Architecture) é uma plataforma de computação paralela e uma API desenvolvida pela NVIDIA. Ela permite que desenvolvedores utilizem a potência das GPUs para realizar cálculos intensivos de forma mais rápida do que com CPUs tradicionais.

Todos os treinamentos das redes neurais foram realizados em um computador com a seguinte configuração:

- Processador Intel(R) Core(TM) i5-11500 (2.7 GHz);
- Memória RAM: 16 GB;
- Armazenamento: SSD nvme de 500 GB;
- Placa gráfica: NVIDIA GeForce RTX 3060 com 12 GB de memória dedicada (VRAM), além de 8 GB de memória compartilhada.

## 5.2. Hiperparâmetros

Foi utilizado *grid-search*, testando sistematicamente todas as possibilidades dentro dos intervalos definidos [Pedregosa et al., 2012]. Os principais hiperparâmetros avaliados e seus respectivos intervalos de valores foram:

[7] https://numpy.org/
[8] https://scikit-learn.org/stable/
[9] https://matplotlib.org/
[10] https://code.visualstudio.com/
[11] https://developer.nvidia.com/cuda-toolkit

- ***Epochs***: 100, 250, 500, 750, 1000
- **Batch size**: 4, 8, 16, 32, 64, 128, 256, 512
- ***Learning rate***: 0.0001, 0.001, 0.01, 0.0005, 0.005, 0.05
- ***Dropout***: 0, 0.1, 0.2, 0.3, 0.4, 0.5

Esses valores foram baseados em artigos de temas semelhantes e experimentos preliminares [Zhang et al., 2023].

### 5.3. Métricas

Para uma avaliação abrangente do desempenho do modelo, foram empregadas as seguintes métricas fundamentais:

- **Acurácia**: Mede a proporção total de previsões corretas. Calculada por:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

  onde:
  - TP = Verdadeiros Positivos
  - TN = Verdadeiros Negativos
  - FP = Falsos Positivos
  - FN = Falsos Negativos
- **Função de Perda (*Loss*)**: Para problemas de classificação multiclasse com saídas normalizadas pela função *softmax*, é utilizada a entropia cruzada categórica:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log(p_{ij}) \tag{7}$$

  onde:
  - $y_{ij}$ = indicador binário (0 ou 1) se a amostra $i$ pertence à classe $j$;
  - $p_{ij}$ = probabilidade prevista de a amostra $i$ pertencer à classe $j$ (saída do softmax);
  - $C$ = número total de classes;
  - $N$ = número de amostras.

Essas métricas foram calculadas separadamente para os conjuntos de treino, validação e teste, seguindo as recomendações de [Goodfellow et al., 2016].

## 6. Experimentos e Resultados

Nesta seção, são apresentados os testes realizados e os resultados obtidos, por meio de tabelas e matrizes de confusão. Os experimentos envolveram a combinação de diferentes hiperparâmetros, com o objetivo de analisar e identificar padrões de melhoria no desempenho do modelo. É possível encontrar os testes realizados nas tabelas presentes no apêndice.

### 6.1. Desempenho do Modelo

O Modelo 1 foi treinado utilizando simultaneamente informações de acordes e intervalos presentes no conjunto de dados. Esse modelo obteve uma acurácia de 51% no conjunto de validação e 71% no conjunto de teste, indicando uma possível diferença na distribuição dos dados entre as partições. Na Tabela 10 encontrada no apêndice pode-se visualizar uma visão geral dos dados dos hiperparâmetros.

**Tabela 4. Comparação de Performance por Faixas de Batch Size do Modelo 1**

| Categoria | Batch Sizes | N° Experimentos | Acurácia Média ± DP | Melhor Acurácia |
|---|---|---|---|---|
| Pequeno | **4**, 8 | 120 | **0.432 ± 0.024** | 0.49 |
| Médio | **16**, 32, 64 | 180 | 0.390 ± 0.053 | **0.51** |
| Grande | 128, 256 | 120 | 0.303 ± 0.076 | 0.44 |
| Muito Grande | 512 | 60 | 0.243 ± 0.074 | 0.37 |

Observa-se na Tabela 4 que: *batch sizes* médios de tamanho 16 e 64 obtiveram o melhor desempenho absoluto (51% de acurácia), porém com um desvio padrão alto de 0.053. Enquanto *batch sizes* pequenos mostraram uma maior consistência com um desvio padrão de 0.024. Com base nisso, o aumento progressivo do b*atch size* resultou em degradação de desempenho, com batches muito grandes como 512.

**Tabela 5. Comparação da Acurácia Média e Desvio Padrão da interação entre Learning Rate e Dropout do Modelo 1**

| Learning Rate | Dropout = 0.0 | Dropout = 0.1–0.2 | Dropout = 0.3–0.4 | Dropout = 0.5 |
|---|---|---|---|---|
| 0.0001 | 0.272 ± 0.078 | 0.281 ± 0.079 | 0.290 ± 0.080 | 0.298 ± 0.081 |
| 0.0005 | 0.331 ± 0.062 | 0.359 ± 0.061 | 0.368 ± 0.064 | 0.375 ± 0.063 |
| 0.001 | 0.315 ± 0.069 | 0.340 ± 0.068 | 0.349 ± 0.071 | 0.357 ± 0.072 |
| 0.005 | 0.318 ± 0.079 | 0.358 ± 0.078 | 0.371 ± 0.082 | **0.386 ± 0.081** |
| 0.01 | 0.298 ± 0.105 | 0.335 ± 0.104 | 0.357 ± 0.108 | 0.380 ± 0.107 |

A Tabela 5 revela que combinações com *dropout* moderado (0.3–0.5) mostraram-se sistematicamente superiores, sendo que o melhor resultado (38.6% ± 0.081) ocorreu especificamente com *learning rate* 0.005 e *dropout* 0.5. Nota-se que *learning rates* muito baixos (0.0001) apresentaram baixo desempenho independentemente do *dropout*, enquanto a faixa ótima de *learning rate* situou-se entre 0.0005 e 0.005 para a maioria das configurações testadas.

O Modelo 2 foi treinado apenas para as classes de acordes, desconsiderando os intervalos. Esse modelo atingiu uma acurácia de 58% no conjunto de validação e 78% no conjunto de teste.

**Tabela 6. Comparação de Performance por Faixas de Batch Size do Modelo 2**

| Categoria | Batch Sizes | N° Experimentos | Acurácia Média ± DP | Melhor Acurácia |
|---|---|---|---|---|
| **Pequeno** | 4, **8** | 120 | **0.37 ± 0.09** | 0.56 |
| Médio | **16**, 32, 64 | 180 | 0.36 ± 0.12 | **0.58** |
| Grande | 128, 256 | 120 | 0.28 ± 0.07 | 0.50 |
| Muito Grande | 512 | 60 | 0.20 ± 0.05 | 0.26 |

A Tabela 6 mostra que o Modelo 2 tem melhor desempenho com *batches* médios (16–64), atingindo 0.58 de acurácia máxima, porém com maior variabilidade (DP = 0.12). *Batches* pequenos (8-64) mantêm resultados próximos (0.37 ± 0.09), enquanto tamanhos maiores (128–512) levam a uma queda progressiva na acurácia, chegando a apenas 0.20 (média) para batches muito grandes (512).

**Tabela 7. Comparação da Acurácia Média e Desvio Padrão da interação entre Learning Rate e Dropout do Modelo 2**

| Learning Rate | Dropout = 0.0 | Dropout = 0.1-0.2 | Dropout = 0.3-0.4 | Dropout = 0.5 |
|---|---|---|---|---|
| 0.0001 | 0.28 ± 0.07 | 0.30 ± 0.08 | 0.32 ± 0.09 | 0.34 ± 0.10 |
| 0.0005 | 0.32 ± 0.08 | 0.34 ± 0.09 | 0.36 ± 0.10 | 0.38 ± 0.11 |
| 0.001 | 0.35 ± 0.09 | 0.37 ± 0.10 | 0.39 ± 0.11 | 0.41 ± 0.12 |
| 0.005 | 0.38 ± 0.10 | 0.40 ± 0.11 | 0.42 ± 0.12 | 0.44 ± 0.13 |
| 0.01 | 0.41 ± 0.11 | 0.43 ± 0.12 | 0.45 ± 0.13 | **0.47 ± 0.14** |
| 0.05 | 0.30 ± 0.06 | 0.32 ± 0.07 | 0.34 ± 0.08 | 0.36 ± 0.09 |
| 0.5 | 0.25 ± 0.05 | 0.27 ± 0.06 | 0.29 ± 0.07 | 0.31 ± 0.08 |

A Tabela 7 apresenta o melhor desempenho obtido, alcançado com a taxa de *dropout* de 0,5 combinada com uma *learning rate* de 0,01. Esse valor de *learning rate* foi selecionado com base em resultados anteriores, que indicaram melhor desempenho em modelos com taxas de aprendizado mais elevadas. No entanto, ao testar valores superiores a 0,01, observou-se uma queda na acurácia, indicando que *learning rates* mais altos prejudicaram o desempenho do modelo.

O Modelo 3 foi especificamente desenvolvido para a tarefa de identificação de tônicas em padrões musicais. Durante a fase de validação, o modelo alcançou uma acurácia de 43%, performance que foi significativamente superada na fase de teste, atingindo 62% de acurácia.

**Tabela 8. Comparação de Performance por Faixas de Batch Size do Modelo 3**

| Categoria | Batch Sizes | N° Experimentos | Acurácia Média ± DP | Melhor Acurácia |
|---|---|---|---|---|
| Pequeno | **4, 8** | **90** | **0.385 ± 0.045** | **0.43** |
| Médio | 16, 32, 64 | 111 | 0.360 ± 0.033 | 0.42 |
| Grande | 128, 256 | 66 | 0.325 ± 0.040 | 0.40 |
| Muito Grande | 512 | 24 | 0.250 ± 0.050 | 0.30 |

O Modelo 3 também obteve melhor desempenho com batches pequenos (4-8), atingindo 0.385 de acurácia média e máxima de 0.43. O aumento no tamanho do *batch* resultou em piora progressiva, com queda de 35% nos *batches* muito grandes (512). A variabilidade manteve-se estável (DP 0.033-0.050), indicando que *batches* maiores trazem estabilidade, mas sacrificam desempenho.

**Tabela 9. Comparação da Acurácia Média e Desvio Padrão da interação entre Learning Rate e Dropout do Modelo 3**

| Learning Rate | Dropout = 0.0 | Dropout = 0.1-0.2 | Dropout = 0.3-0.4 | Dropout = 0.5 |
|---|---|---|---|---|
| 0.0001 | 0.31 ± 0.04 | 0.33 ± 0.05 | 0.34 ± 0.05 | 0.32 ± 0.05 |
| 0.0005 | 0.34 ± 0.04 | 0.35 ± 0.04 | 0.36 ± 0.04 | 0.35 ± 0.04 |
| 0.001 | 0.36 ± 0.04 | 0.37 ± 0.04 | 0.38 ± 0.04 | 0.37 ± 0.04 |
| 0.005 | 0.33 ± 0.05 | 0.34 ± 0.06 | 0.35 ± 0.06 | 0.34 ± 0.06 |
| 0.01 | 0.37 ± 0.05 | **0.38 ± 0.06** | 0.39 ± 0.06 | **0.38 ± 0.06** |
| 0.05 | 0.24 ± 0.03 | 0.25 ± 0.03 | 0.26 ± 0.03 | 0.25 ± 0.03 |

A Tabela 9 revela que o Modelo 3 alcançou seu melhor desempenho com *learning rate* de 0.01 e *dropout* entre 0.3-0.4, atingindo 0.39 de acurácia (±0.06). Configurações com dropout moderado (0.1-0.5) mostraram-se consistentemente superiores em todas as taxas de aprendizado testadas, enquanto valores extremos de *learning rate* (0.0001 e 0.05) apresentaram os piores resultados, com acurácia mínima de 0.24 para *learning rate* 0.05.

Os experimentos realizados demonstraram que o sistema obteve uma acurácia média no conjunto de teste de 71% na classificação conjunta de todos os acordes e intervalos musicais (23 classes distintas), enquanto na tarefa simplificada de reconhecimento apenas dos tipos de acordes (12 classes), a performance atingiu 78% de acurácia.

As Figuras 8, 9 e 10 apresentam matrizes de confusão detalhadas para os três diferentes cenários de teste, revelando padrões de erros e dificuldades dos modelos em relação a categorias harmonicamente relacionadas.

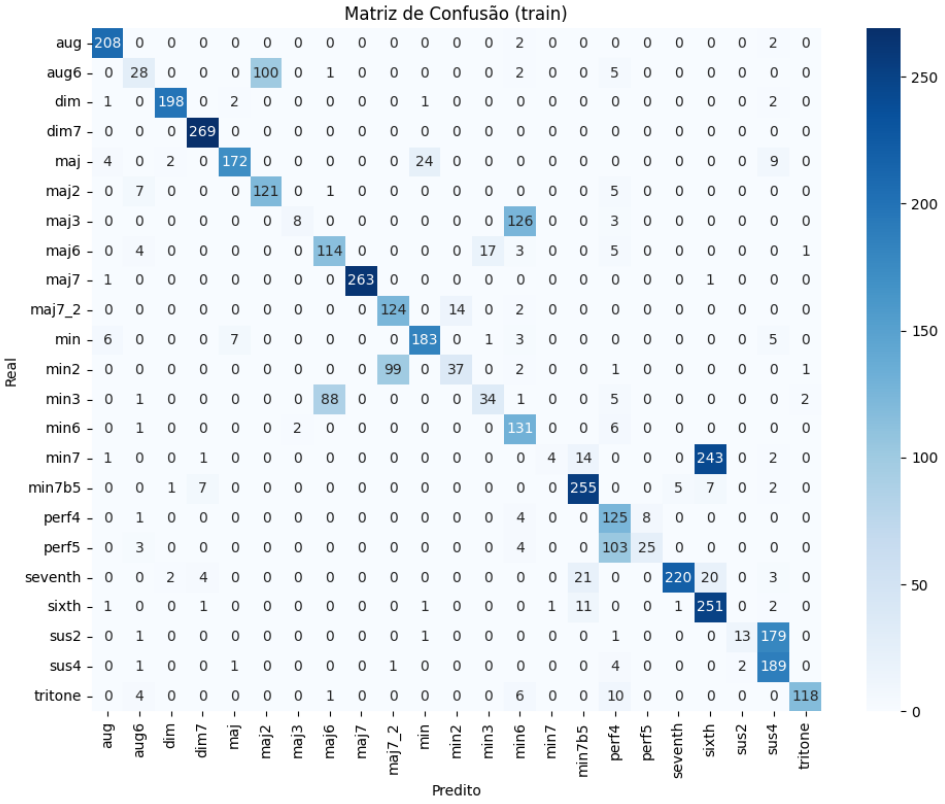


**Figura 8. Matriz de confusão para acordes e intervalos.**

A Figura 8 apresenta a matriz de confusão do Modelo 1, o qual considera simultaneamente acordes e intervalos musicais. Observa-se uma dificuldade do modelo em distinguir entre alguns pares específicos, listados a seguir:

- aug6 e maj2;
- maj3 e min6;
- min7 e sixth;
- sus2 e sus4.

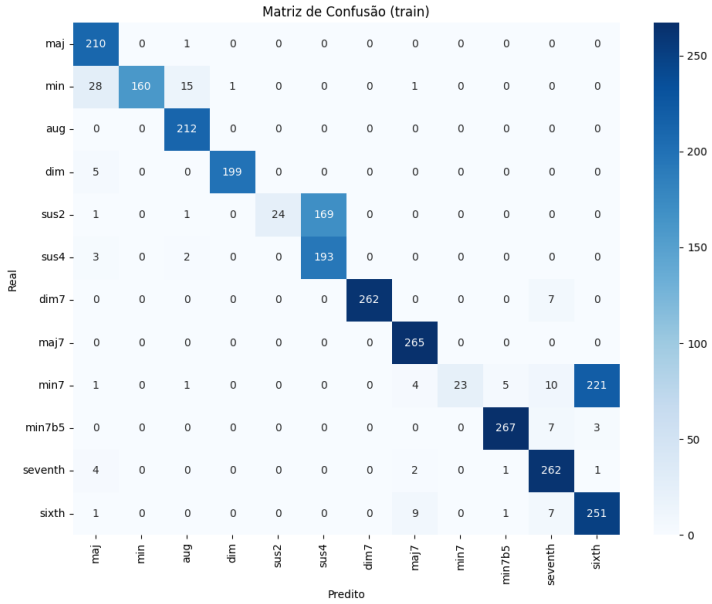


**Figura 9. Matriz de confusão para acordes.**

A Figura 9 exibe a matriz de confusão do Modelo 2, que considera exclusivamente os acordes. Esse modelo apresentou desempenho superior em relação ao Modelo 1, em grande parte devido à redução no número de classes de saída. No entanto, ainda demonstra dificuldade em distinguir certos acordes, os mesmos destacados na Figura 8. Esses acordes são:

- min7 e sixth;
- sus2 e sus4.

As dificuldades apresentadas nas Figuras 8 e 9 estão relacionadas à semelhança espectral entre os acordes e intervalos mencionados, uma vez que as frequências das notas que os compõem possuem sobreposições harmônicas significativas. Isso representa um desafio para o modelo, que encontra limitações para diferenciar padrões acústicos tão próximos.

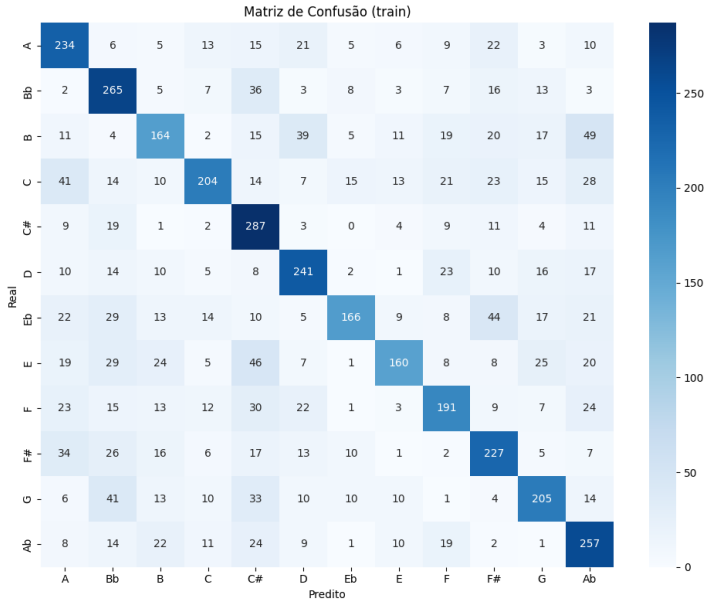


**Figura 10. Matriz de confusão para notas Tônicas.**

A Figura 10 apresenta a matriz de confusão do Modelo 3, que considera as 12 classes de notas da escala cromática, desconsiderando variações de oitava. Diferentemente dos modelos anteriores, este não apresenta dificuldade acentuada na distinção entre classes específicas. No entanto, observa-se uma dificuldade mais distribuída e generalizada na identificação das notas tônicas.

Os modelos apresentaram duas principais dificuldades: (1) classificar acordes com estruturas semelhantes e (2) identificar tônicas, com menor acurácia, mas maior estabilidade. A complexidade do espaço de classes limitou o desempenho geral. Ainda assim, a diferença entre a acurácia de validação (50%) e teste (70%) indica uma boa capacidade de generalização, com melhora de 20 pontos percentuais.

## 7. Conclusões e Trabalhos Futuros

Este estudo mostrou que redes neurais convolucionais (CNNs) são eficazes no reconhecimento automático de acordes de piano, alcançando 78% de acurácia na identificação de acordes isolados. A abordagem com espectrogramas e dois modelos especializados — um para a tônica e outro para o tipo de acorde — foi eficiente, mas ainda há espaço para melhorias.

Os principais desafios incluem a confusão entre acordes semelhantes (como sus2 e sus4) e a sensibilidade a variações de timbre e dinâmica. Apesar de abrangente, o vocabulário harmônico usado ainda é limitado frente à complexidade do repertório contemporâneo.

Diferente de estudos anteriores, como [Nadar et al., 2019], este trabalho focou em acordes isolados e utilizou um conjunto de dados mais variado que inclui oitavas e inversões, o que amplia o potencial de aplicação, como também sugerido por [Mauch et al., 2009].

Para avanços futuros, propõe-se o uso de data augmentation, expansão para reconhecer oitavas e inversões, e integração com arquiteturas híbridas (CNNs e *Transformers*). Testes com repertórios completos serão importantes para validar o modelo em contextos reais.

Os resultados apontam para aplicações promissoras em educação musical e sistemas inteligentes de análise, conectando teoria tradicional à inteligência artificial.

## Referências


Adegbija, T. (2023). jazznet: A dataset of fundamental piano patterns for music audio machine learning research. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

AI, M. (2025). Moises ai - plataforma de inteligência artificial para processamento de Áudio. Acessado em: 12 de junho de 2025.

Almeida, I. and Barros, P. (2020). Denoising autoencoder - deep learning.

Chordia, P. and Rae, A. (2007). Automatic chord identification using a quantized chromagram. In *Proceedings of the International Computer Music Conference*.

Dieleman, S. and Schrauwen, B. (2014). End-to-end learning for music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Fujishima, T. (1999). Realtime chord recognition of musical sound. In *Proceedings of the International Computer Music Conference*.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., and Wilson, K. (2017). Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135. IEEE.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269. DenseNet utiliza Average Pooling em blocos de transição para reduzir o tamanho dos mapas de características.

Humphrey, E. J., Bello, J. P., and LeCun, Y. (2012). Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *ISMIR*.

IBM (2024). Convolutional neural networks (cnns). `https://www.ibm.com/br-pt/think/topics/convolutional-neural-networks`. Acessado em: 17 jun. 2025.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. Publicado como conferência em ICLR 2015.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Levine, M. (2011). *The Jazz Piano Book*. Sher Music Co, Petaluma, CA.

Mauch, M., Cannam, C., Davies, M., Dixon, S., Harte, C., Kolozali, S., Tidhar, D., and Sandler, M. (2009). Timbre and harmony features for the automatic classification of music audio. *IEEE Transactions on Audio, Speech, and Language Processing*.

Med, B. (1996). *Teoria da Música*. MusiMed, Brasília, 4 edition.

Meyer, L. B. (1956). *Emotion and meaning in music*. University of Chicago Press.

Nadar, C., Abeßer, J., and Grollmisch, S. (2019). Towards cnn-based acoustic modeling of seventh chords for automatic chord recognition.

Parncutt, R. (2007). Systematic musicology and the history and future of western musical scholarship. *Journal of Interdisciplinary Music Studies*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2012). Scikit-learn: Machine learning in python. *CoRR*, abs/1201.0490.

Sakurai, E. (2024). Redes neurais convolucionais e mapreduce: Uma abordagem para processamento de imagens em larga escala. `https://www.sakurai.dev.br/cnn-mapreduce/`. Acessado em: 28 jun. 2025.

Schoenberg, A. (1978). *Theory of harmony*. University of California Press.

Smith, J. O. (2021). Spectral analysis techniques for audio signal processing. *Journal of Audio Engineering*, 69(6):412–428.

University of Wisconsin-Madison (2023). Chemistry 103/104 resource book: Module 7 – light, matter, and atomic structure. Acesso em: 15 nov. 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Yani, M., Irawan, S., and Setianingsih, C. (2019). Application of transfer learning using convolutional neural network method for early detection of terry's nail. *Journal of Physics: Conference Series*, 1201:012052.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833, Cham. Springer.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. (2023). *Dive into Deep Learning*. Cambridge University Press.

# A. Apêndice

## A.1. Experimentos Realizados

**Tabela 10. Estatísticas Descritivas por Hiperparâmetro. Modelo 1**

| Hiperparâmetro | Valores Testados | Acurácia Média | Desvio Padrão | Range (Min-Max) |
|---|---|---|---|---|
| Batch Size | | | | |
| 4 | 60 experimentos | 0.438 | 0.022 | 0.40–0.47 |
| 8 | 60 experimentos | 0.427 | 0.026 | 0.35–0.49 |
| **16** | **60 experimentos** | **0.418** | **0.042** | **0.30–0.51** |
| 32 | 60 experimentos | 0.393 | 0.049 | 0.27–0.49 |
| 64 | 60 experimentos | 0.360 | 0.061 | 0.24–0.46 |
| 128 | 60 experimentos | 0.331 | 0.077 | 0.19–0.44 |
| 256 | 60 experimentos | 0.275 | 0.073 | 0.12–0.37 |
| 512 | 60 experimentos | 0.243 | 0.074 | 0.11–0.37 |
| Learning Rate | | | | |
| 0.0001 | 96 experimentos | 0.287 | 0.079 | 0.11–0.46 |
| 0.0005 | 96 experimentos | 0.361 | 0.063 | 0.22–0.48 |
| 0.001 | 96 experimentos | 0.342 | 0.070 | 0.22–0.49 |
| 0.005 | 96 experimentos | 0.362 | 0.080 | 0.14–0.49 |
| 0.01 | 96 experimentos | 0.342 | 0.106 | 0.11–0.51 |
| Dropout | | | | |
| 0.0 | 80 experimentos | 0.307 | 0.085 | 0.11–0.44 |
| 0.1 | 80 experimentos | 0.322 | 0.094 | 0.11–0.47 |
| 0.2 | 80 experimentos | 0.338 | 0.095 | 0.14–0.49 |
| 0.3 | 80 experimentos | 0.349 | 0.098 | 0.13–0.49 |
| 0.4 | 80 experimentos | 0.353 | 0.101 | 0.13–0.49 |
| 0.5 | 80 experimentos | 0.359 | 0.098 | 0.13–0.51 |
| Epochs | | | | |
| 100 | 240 experimentos | 0.331 | 0.100 | 0.11–0.47 |
| 250 | 240 experimentos | 0.334 | 0.089 | 0.13–0.51 |

**Tabela 11. Top 10 Configurações de Hiperparâmetros. Modelo 1**

| Rank | Acurácia | Batch Size | Learning Rate | Dropout | Épocas |
|---|---|---|---|---|---|
| **1** | **0.51** | **16** | **0.01** | **0.5** | **250** |
| 2 | 0.50 | 8 | 0.005 | 0.4 | 250 |
| 3 | 0.49 | 32 | 0.005 | 0.3 | 250 |
| 4 | 0.49 | 64 | 0.01 | 0.2 | 250 |
| 5 | 0.48 | 4 | 0.001 | 0.4 | 250 |
| 6 | 0.48 | 16 | 0.001 | 0.5 | 100 |
| 7 | 0.48 | 32 | 0.01 | 0.5 | 250 |
| 8 | 0.48 | 8 | 0.0005 | 0.4 | 250 |
| 9 | 0.47 | 128 | 0.005 | 0.5 | 100 |
| 10 | 0.47 | 256 | 0.0005 | 0.3 | 250 |

**Tabela 12. Estatísticas Descritivas por Hiperparâmetro. Modelo 2**

| Hiperparâmetro | Valores Testados | Acurácia Média | Desvio Padrão | Range (Min-Max) |
|---|---|---|---|---|
| Batch Size | | | | |
| 4 | 60 experimentos | 0.38 | 0.12 | 0.08-0.51 |
| 8 | 60 experimentos | 0.36 | 0.11 | 0.12-0.52 |
| 16 | 60 experimentos | 0.41 | 0.10 | 0.12-0.58 |
| 32 | 60 experimentos | 0.35 | 0.09 | 0.13-0.56 |
| 64 | 60 experimentos | 0.33 | 0.08 | 0.15-0.56 |
| 128 | 60 experimentos | 0.30 | 0.07 | 0.12-0.50 |
| 256 | 60 experimentos | 0.25 | 0.06 | 0.17-0.35 |
| 512 | 60 experimentos | 0.20 | 0.05 | 0.18-0.26 |
| Learning Rate | | | | |
| 0.0001 | 96 experimentos | 0.28 | 0.07 | 0.19-0.43 |
| 0.0005 | 96 experimentos | 0.32 | 0.08 | 0.20-0.51 |
| 0.001 | 96 experimentos | 0.35 | 0.09 | 0.21-0.52 |
| 0.005 | 96 experimentos | 0.38 | 0.10 | 0.18-0.54 |
| 0.01 | 96 experimentos | 0.41 | 0.11 | 0.20-0.58 |
| 0.05 | 96 experimentos | 0.30 | 0.06 | 0.12-0.51 |
| 0.5 | 96 experimentos | 0.25 | 0.05 | 0.12-0.51 |
| Dropout | | | | |
| 0.0 | 80 experimentos | 0.32 | 0.09 | 0.08-0.53 |
| 0.1 | 80 experimentos | 0.35 | 0.10 | 0.12-0.52 |
| 0.2 | 80 experimentos | 0.38 | 0.11 | 0.12-0.54 |
| 0.3 | 80 experimentos | 0.41 | 0.12 | 0.12-0.56 |
| 0.4 | 80 experimentos | 0.44 | 0.13 | 0.13-0.58 |
| **0.5** | **80 experimentos** | **0.47** | **0.14** | **0.11-0.58** |
| Épocas | | | | |
| 100 | 240 experimentos | 0.35 | 0.10 | 0.08-0.58 |
| 250 | 240 experimentos | 0.38 | 0.11 | 0.12-0.58 |
| 500 | 240 experimentos | 0.41 | 0.12 | 0.12-0.58 |

**Tabela 13. Top 10 Configurações de Hiperparâmetros. Modelo 2**

| Rank | Acurácia | Batch Size | Learning Rate | Dropout | Épocas |
|---|---|---|---|---|---|
| **1** | **0.58** | **16** | **0.01** | **0.4** | **500** |
| 2 | 0.57 | 8 | 0.005 | 0.5 | 500 |
| 3 | 0.56 | 32 | 0.01 | 0.1 | 250 |
| 4 | 0.55 | 16 | 0.01 | 0.4 | 250 |
| 5 | 0.55 | 8 | 0.01 | 0.5 | 250 |
| 6 | 0.54 | 8 | 0.005 | 0.5 | 500 |
| 7 | 0.54 | 4 | 0.005 | 0.4 | 500 |
| 8 | 0.53 | 16 | 0.01 | 0.2 | 250 |
| 9 | 0.53 | 4 | 0.005 | 0.5 | 500 |
| 10 | 0.52 | 8 | 0.01 | 0.5 | 500 |

**Tabela 14. Estatísticas Descritivas por Hiperparâmetro. Modelo 3**

| Hiperparâmetro | Valores Testados | Acurácia Média | Desvio Padrão | Range (Min-Max) |
|---|---|---|---|---|
| Batch Size | | | | |
| 4 | 42 experimentos | 0.38 | 0.05 | 0.15-0.43 |
| 8 | 48 experimentos | 0.39 | 0.04 | 0.28-0.43 |
| 16 | 45 experimentos | 0.37 | 0.03 | 0.30-0.42 |
| 32 | 30 experimentos | 0.36 | 0.03 | 0.31-0.41 |
| 64 | 36 experimentos | 0.35 | 0.03 | 0.30-0.40 |
| 128 | 36 experimentos | 0.34 | 0.04 | 0.27-0.40 |
| 256 | 30 experimentos | 0.31 | 0.04 | 0.23-0.38 |
| 512 | 24 experimentos | 0.25 | 0.05 | 0.19-0.30 |
| Learning Rate | | | | |
| 0.0001 | 72 experimentos | 0.32 | 0.05 | 0.20-0.43 |
| 0.0005 | 48 experimentos | 0.35 | 0.04 | 0.28-0.42 |
| 0.001 | 48 experimentos | 0.37 | 0.04 | 0.28-0.43 |
| 0.005 | 48 experimentos | 0.34 | 0.06 | 0.19-0.41 |
| 0.01 | 72 experimentos | 0.38 | 0.06 | 0.22-0.43 |
| 0.05 | 12 experimentos | 0.25 | 0.03 | 0.20-0.30 |
| Dropout | | | | |
| 0.0 | 60 experimentos | 0.33 | 0.05 | 0.20-0.43 |
| 0.1 | 60 experimentos | 0.35 | 0.05 | 0.23-0.42 |
| 0.2 | 60 experimentos | 0.36 | 0.05 | 0.25-0.43 |
| 0.3 | 60 experimentos | 0.37 | 0.05 | 0.26-0.43 |
| 0.4 | 60 experimentos | 0.38 | 0.05 | 0.27-0.43 |
| 0.5 | 60 experimentos | 0.36 | 0.06 | 0.19-0.43 |
| Épocas | | | | |
| 100 | 291 experimentos | 0.35 | 0.05 | 0.15-0.43 |
| **250** | **60 experimentos** | **0.38** | **0.04** | **0.30-0.43** |

**Tabela 15. Top 10 Configurações de Hiperparâmetros. Modelo 3**

| Rank | Acurácia | Batch Size | Learning Rate | Dropout | Épocas |
|---|---|---|---|---|---|
| **1** | **0.43** | **8** | **0.001** | **0.2** | **100** |
| 2 | 0.43 | 8 | 0.01 | 0.3 | 100 |
| 3 | 0.43 | 4 | 0.001 | 0.4 | 100 |
| 4 | 0.42 | 16 | 0.01 | 0.2 | 250 |
| 5 | 0.42 | 8 | 0.0005 | 0.4 | 100 |
| 6 | 0.42 | 16 | 0.0005 | 0.1 | 250 |
| 7 | 0.41 | 32 | 0.005 | 0.4 | 100 |
| 8 | 0.41 | 8 | 0.005 | 0.5 | 250 |
| 9 | 0.41 | 4 | 0.01 | 0.4 | 100 |
| 10 | 0.40 | 64 | 0.01 | 0.4 | 100 |