Identificação de Instrumentos Musicais em Trilhas de Áudio Utilizando Deep Learning

# 1. Introdução ao Problema

Este projeto teve como objetivo desenvolver um modelo de *deep learning* capaz de identificar automaticamente o instrumento musical presente em trilhas de áudio curtas. A tarefa pertence ao campo da classificação de áudio, uma área desafiadora da inteligência artificial devido à natureza complexa dos sinais sonoros.

Diferente de imagens, que já são matrizes de números que redes neurais conseguem processar diretamente, os arquivos de áudio precisam ser convertidos para representações numéricas adequadas. Áudios digitais são essencialmente sequências de amplitudes variando ao longo do tempo, geralmente salvos em formatos como .wav ou .mp3, que não são diretamente interpretáveis por redes neurais. Por isso, uma etapa fundamental do projeto foi a conversão desses sinais para formas que representem melhor suas características sonoras, como espectrogramas ou coeficientes cepstrais (MFCCs).

Além disso, trabalhar com áudio envolve diversos desafios adicionais:

* Variação na duração das trilhas, exigindo cortes ou preenchimentos (padding);
* Diferenças de volume, ruído e qualidade das gravações;
* Desequilíbrio entre classes no dataset, o que afeta o desempenho do modelo;
* Transformações de frequência e tempo que precisam preservar características relevantes para a tarefa de classificação.

Utilizamos para este estudo o dataset “Musical Instruments Sound Dataset”, disponível no Kaggle, que contém quatro classes:

* **Guitar** – 700 amostras
* **Drum** – 700 amostras
* **Violin** – 700 amostras
* **Piano** – 528 amostras

O conjunto de teste inclui 80 trilhas (20 de cada classe), permitindo validar a generalização do modelo em sons nã o vistos previamente.

# 2. Preparação dos Dados

A preparação dos dados foi uma das etapas mais importantes deste projeto, pois o sucesso de modelos de deep learning com áudio depende fortemente da forma como os dados são representados e enriquecidos. Trabalhar com áudio cru requer uma série de transformações para torná-lo compatível com o formato de entrada das redes neurais.

**Download e Organização**

Utilizamos a biblioteca kagglehub para baixar diretamente o dataset do Kaggle e realizamos os seguintes ajustes iniciais:

* Padronização dos nomes dos arquivos e das classes (por exemplo, corrigindo erros como guiatr para guitar);
* Conversão de todos os arquivos para o formato .wav, quando necessário;
* Normalização das trilhas para manter um volume consistente entre os arquivos.

**Data Augmentation**

Para aumentar a robustez do modelo e reduzir o risco de overfitting, aplicamos técnicas de *data augmentation* no domínio do áudio. Essas transformações simulam variações naturais que podem ocorrer em gravações reais. Abaixo, descrevemos cada técnica utilizada:

* **Pitch Shift:** altera a afinação do áudio. Exemplo: pitch\_up\_0\_5 (meio tom acima), pitch\_down\_1 (um tom abaixo).
* **Time-Stretch:** modifica a velocidade do áudio sem alterar sua afinação. Exemplo: stretch\_1.02 (mais rápido), stretch\_0.95 (mais lento).
* **Gain:** aumenta ou diminui o volume do áudio. Exemplo: gain\_0.8 (volume reduzido), gain\_1.2 (volume aumentado).
* **Noise Injection:** adiciona ruído branco leve ao áudio, simulando ambientes com interferência. Exemplo: noise\_0\_003, noise\_0\_01.
* **Combinações de transformações:** criamos variações complexas combinando duas ou mais transformações para gerar dados ainda mais diversos. Exemplos:
  + stretch\_pitch\_up: combina estiramento temporal com aumento de pitch;
  + pitch\_noise\_gain: combina alteração de pitch, adição de ruído e aumento de volume;
  + stretch\_down\_pitch\_up: estica o áudio e eleva sua afinação;
  + pitch\_down\_1\_noise: abaixa o pitch e adiciona ruído.

Essas técnicas aumentaram significativamente a diversidade do dataset, ajudando o modelo a generalizar melhor para diferentes condições sonoras.

**Slicing dos Áudios (para uso com Espectrogramas)**

Na primeira tentativa de modelagem, utilizando espectrogramas de Mel, foi necessário padronizar a duração das trilhas de áudio. Para isso, realizamos o corte (slicing) das gravações em janelas fixas de 3 segundos.

Adotamos também o uso de **overlap** entre as janelas (por exemplo, 50%) para evitar perda de informação nos pontos de corte e garantir melhor cobertura temporal. Essa estratégia resultou em múltiplos segmentos por arquivo original, aumentando o volume de dados e mantendo a continuidade das características sonoras.

**Espectrograma de Mel**

Após o slicing, convertemos os segmentos de áudio em espectrogramas de Mel. O espectrograma é uma representação visual da energia das frequências ao longo do tempo. A escala de Mel, por sua vez, busca aproximar essa representação da percepção auditiva humana, oferecendo maior resolução em frequências mais baixas.

O processo consistiu nas seguintes etapas:

1. Conversão do sinal de tempo para o domínio da frequência utilizando FFT (Fast Fourier Transform);
2. Aplicação de filtros na escala de Mel para extrair bandas perceptualmente relevantes;
3. Aplicação de uma transformação logarítmica para destacar variações mais sutis de energia.

O resultado é uma matriz 2D (tempo × frequência), semelhante a uma imagem, o que possibilita a utilização de redes convolucionais. Contudo, esse pipeline se mostrou pesado computacionalmente, tanto na preparação quanto no treinamento, e não trouxe ganhos significativos de acurácia. Por isso, posteriormente, migramos para uma abordagem mais leve baseada em MFCCs.

A close-up of a sound wave

AI-generated content may be incorrect.

A screen shot of a graph

AI-generated content may be incorrect.

A conversão de sinais de áudio para espectrogramas de Mel é computacionalmente intensiva, pois envolve operações como a transformada rápida de Fourier (FFT) em janelas deslizantes. Esse processo exige bastante uso de CPU, especialmente ao lidar com grandes volumes de dados e múltiplas janelas com sobreposição.

# 3. Arquitetura e Modelagem

## 3.1 Mel Spectogram e Redes Convolucionais

Na primeira abordagem de modelagem, optamos por transformar os segmentos de áudio em espectrogramas de Mel e tratá-los como imagens bidimensionais. Essa escolha foi motivada pela natureza visual do espectrograma, que contém padrões temporais e espectrais bem definidos — similares a texturas visuais — e que podem ser capturados eficientemente por redes convolucionais (CNNs).

Utilizamos uma arquitetura sequencial composta por camadas Conv2D, que extraem padrões espaciais locais das representações tempo-frequência. Cada bloco de convolução foi seguido por normalização em lote (BatchNormalization) para estabilizar o treinamento e pooling (MaxPooling2D) para reduzir a dimensionalidade espacial. Após a extração dos mapas de ativação, aplicamos GlobalAveragePooling2D para resumir a informação espacial, seguido por camadas densas para aprendizado de padrões mais abstratos e classificação final com softmax.

A screenshot of a computer

AI-generated content may be incorrect.

A motivação por trás dessa arquitetura foi:

* **Camadas convolucionais:** detectar padrões locais recorrentes nos espectrogramas (como harmônicos ou transientes);
* **Pooling:** reduzir o custo computacional e capturar características invariantes a pequenas mudanças;
* **Camadas densas finais:** realizar a tomada de decisão com base nas representações extraídas pelas convoluções.

Além disso, identificamos um desequilíbrio moderado no número de amostras por classe — especialmente no caso de piano, com menos exemplos. Para mitigar esse efeito, calculamos pesos por classe utilizando a função compute\_class\_weight da biblioteca sklearn, atribuindo maior peso às classes menos representadas durante o treinamento. Esses pesos foram aplicados diretamente na função de perda da rede.

Apesar da estrutura bem fundamentada e dos cuidados com o balanceamento, o modelo não teve desempenho satisfatório. No próximo tópico explicamos os resultados obtidos e os fatores que levaram à limitação dessa abordagem.

Além da complexidade computacional envolvida na conversão para espectrogramas, essa abordagem exigiu que todos os áudios fossem cortados em janelas fixas de 3 segundos com sobreposição. Essa estratégia era necessária para padronizar a entrada do modelo, uma vez que redes convolucionais esperam tensores de forma fixa.

No entanto, esse processo introduziu um efeito colateral significativo: a perda de continuidade rítmica e estrutural dos áudios. Muitos instrumentos, como violino e bateria, dependem fortemente de variações temporais e dinâmicas ao longo da execução, que são prejudicadas quando o áudio é segmentado fora de contexto. Isso pode fazer com que trechos isolados percam características essenciais do instrumento, dificultando a tarefa da rede em aprender padrões discriminativos consistentes.

Essa limitação, somada ao alto custo de processamento para geração dos espectrogramas e ao desempenho apenas razoável do modelo, motivou a busca por alternativas mais leves e eficientes. A seguir, detalharemos os resultados obtidos com essa arquitetura e os motivos pelos quais ela não foi bem-sucedida.

**Resultados e Análise**

Apesar do esforço em estruturar uma rede convolucional apropriada e aplicar técnicas de regularização e balanceamento de classes, os resultados obtidos com essa abordagem foram extremamente insatisfatórios.

O modelo atingiu uma **acurácia de apenas 7%**, com **precisão, recall e F1-score praticamente nulos** para três das quatro classes (drum, guitar e piano). O único desempenho aceitável ocorreu para a classe **violin**, com **recall de 1.00**, mas com uma **precisão extremamente baixa (0.07)**, o que indica que o modelo classificou quase todas as amostras incorretamente como violino.

Esses resultados sugerem que a rede aprendeu a **classificar tudo como uma única classe** (neste caso, violin), comportamento típico de um modelo que sofre de *colapso de predição*. Esse problema pode estar relacionado a diversos fatores combinados:

* **Representações ineficazes:** embora o espectrograma de Mel seja visualmente rico, ele pode ter codificado informações irrelevantes ou ruidosas nos cortes de 3 segundos;
* **Perda de contexto temporal:** a fragmentação das trilhas em janelas desconexas eliminou estruturas rítmicas e harmônicas que são cruciais para distinguir instrumentos;
* **Desbalanceamento efetivo:** mesmo com o uso de pesos por classe, o volume de dados por classe após o slicing e data augmentation pode ter distorcido a distribuição original;
* **Overfitting prematuro:** o modelo pode ter memorizado padrões espúrios em uma única classe devido à baixa variabilidade nas representações ou configuração inadequada de treinamento.

Além disso, o custo computacional da extração dos espectrogramas e o tempo de treinamento elevado não foram compensados por ganhos de desempenho. Isso motivou a reavaliação da abordagem, levando à adoção de estratégias mais leves e eficazes baseadas em MFCCs.

## 3.2 Extração de MFCC com Rede Densa

Após os resultados insatisfatórios com espectrogramas de Mel e redes convolucionais, optamos por uma abordagem mais leve e direta: utilizar extração de características no domínio do tempo-frequência com MFCCs (Mel Frequency Cepstral Coefficients), combinada com uma rede densa sem convoluções.

**Extração de MFCC**

Os MFCCs são amplamente utilizados em tarefas de reconhecimento de fala e classificação de áudio por representarem de forma compacta e perceptivamente relevante o conteúdo espectral de um sinal sonoro. Em resumo, os MFCCs descrevem como a energia do sinal é distribuída em bandas de frequência ajustadas à percepção humana (escala de Mel), e em seguida aplicam uma transformada discreta do cosseno para capturar apenas os coeficientes mais informativos.

Utilizamos a função librosa.feature.mfcc para extrair os MFCCs diretamente dos áudios do dataset já aumentado (sem cortes), evitando o custo computacional do slicing e preservando melhor a estrutura temporal dos instrumentos.

**Estrutura do Modelo**

A arquitetura utilizada foi uma rede **totalmente conectada (dense)**, projetada para receber vetores de características MFCC e derivadas (como delta, chroma e spectral contrast). O modelo foi construído em camadas sucessivas com regularização, normalização e ativação swish, visando maximizar a capacidade expressiva com baixo custo de processamento.

A rede apresenta a seguinte organização:

* **Camada inicial:** normaliza os valores de entrada com BatchNormalization, essencial para estabilidade numérica e aceleração do treinamento.
* **Bloco 1:** camada densa com 512 neurônios e ativação swish, seguida de normalização e dropout de 40% para prevenir overfitting.
* **Bloco 2 (Residual 1):** sequência de duas camadas densas de 256 neurônios com dropout, conectadas com um somatório residual entre x1 e x2. A adição residual permite preservar informações da etapa anterior e facilita a propagação do gradiente, semelhante a arquiteturas ResNet.
* **Bloco 3 (Residual 2):** repete o mesmo padrão com duas camadas de 128 neurônios e soma residual entre x3 e x4.
* **Bloco final:** camadas densas menores (64 e 32 neurônios) com ativação ReLU, preparando o vetor de representação para a camada de saída.
* **Saída:** camada Dense com ativação softmax, produzindo as probabilidades para cada uma das quatro classes.

A screen shot of a computer

AI-generated content may be incorrect.

**Focal Loss**

Como a classe Violin continuava difícil de ser identificada, mesmo após o balanceamento dos dados, testamos uma função de perda customizada baseada na **Focal Loss**, com os parâmetros gamma e alpha ajustáveis.

A Focal Loss penaliza menos os exemplos fáceis e foca o aprendizado nos exemplos difíceis, o que se mostrou especialmente útil para lidar com as classes mais desafiadoras e desbalanceadas. Isso ajudou a reduzir o viés de predição dominante que havíamos observado na primeira abordagem.

**Vantagens da Abordagem**

* Evitamos o slicing, mantendo o áudio original completo;
* O processamento com MFCC foi muito mais rápido do que a geração de espectrogramas;
* A rede densa foi mais leve e simples de treinar;
* O uso de blocos residuais ajudou a preservar informações em profundidade sem degradar o gradiente;
* A Focal Loss contribuiu para melhorar o recall em classes mais difíceis.

Essa combinação nos levou a resultados significativamente melhores que a abordagem anterior, com excelente desempenho em três das quatro classes, como veremos a seguir.

**Resultados e Análise**

A segunda abordagem apresentou avanços significativos em relação à tentativa anterior. Com a extração de MFCCs e uso de uma rede densa com arquitetura mais estável, obtivemos **acurácia geral de 73%**, com excelente desempenho nas classes **guitar** e **piano**, e um bom avanço na detecção de **drum**.

A diagram of different types of musical instruments

AI-generated content may be incorrect.

Os principais resultados por classe foram:

* **Guitar e Piano:** precisão e recall próximos de 99%, indicando que o modelo aprendeu com sucesso os padrões sonoros desses instrumentos. Provavelmente, essas classes apresentam características melódicas e harmônicas consistentes e mais facilmente separáveis pelas features MFCC.
* **Drum:** recall muito alto (0.98), com precisão moderada (0.50), sugerindo que a rede tende a identificar corretamente instâncias de bateria, mas também incorretamente classifica outras como drum (falsos positivos).
* **Violin:** desempenho extremamente fraco, com recall de apenas 0.02 e F1-score de 0.04. Isso indica que, embora a rede consiga classificar quase tudo com boa precisão nas demais classes, ela praticamente **falha em reconhecer violino**.

A matriz de confusão evidencia esse comportamento: quase todas as instâncias de violin foram classificadas como drum, mostrando **grande sobreposição nas representações dessas duas classes**. É possível que os MFCCs não estejam capturando nuances específicas do timbre do violino, ou que o dataset possua gravações ruidosas ou inconsistentes dessa classe.

Além disso, há indícios de que o modelo priorizou performance nas classes com padrões mais estáveis e bem representados, e não conseguiu generalizar adequadamente para aquelas com maior variabilidade, como o violino.

Esses resultados mostram que, embora a abordagem tenha melhorado bastante, ainda há **espaço para refino**, especialmente com relação ao violin. A seguir, detalharemos nossa tentativa de contornar essa limitação através de um modelo binário especializado.

## 3.3 Modelo Binário Especializado: Drum vs Violin

Diante da dificuldade recorrente do modelo multiclasse em distinguir entre as classes *drum* e *violin*, decidimos criar uma abordagem mais especializada: um modelo binário treinado exclusivamente para classificar essas duas categorias. A premissa foi que, ao reduzir o espaço de decisão para apenas duas classes, o modelo poderia aprender melhor as sutis diferenças entre elas.

**Arquitetura da Rede**

A arquitetura utilizada para essa tarefa foi uma versão simplificada da rede densa anterior, com o mesmo tipo de entrada baseada em MFCCs, mas com uma **saída binária (sigmoid)** e um pipeline totalmente supervisionado com função de perda binary\_crossentropy.

A estrutura do modelo foi organizada da seguinte forma:

A screenshot of a computer program

AI-generated content may be incorrect.

* **Entrada normalizada:** camada BatchNormalization aplicada diretamente ao vetor de entrada para estabilizar o treinamento e acelerar a convergência;
* **Camada 1:** 512 neurônios com ativação *swish* e regularização L2. Essa camada inicial visa capturar padrões amplos e abstratos da distribuição sonora entre os dois instrumentos;
* **Camadas 2 e 3:** densidades progressivamente menores (256 e 128 neurônios), também com *swish*, normalização e dropout, para promover o aprendizado hierárquico de características;
* **Camada 4:** camada com 64 neurônios e ativação *ReLU*, responsável por uma compressão final das informações antes da tomada de decisão;
* **Saída:** um único neurônio com ativação sigmoid, retornando a probabilidade da amostra ser *drum* (ou *violin*, dependendo da codificação de labels).

Essa arquitetura foi projetada para balancear capacidade de representação e simplicidade, evitando overfitting em uma tarefa binária, especialmente considerando que drum e violin são classes com padrões temporais e espectrais complexos que podem se sobrepor nos MFCCs.

A expectativa era que, com foco exclusivo nesses dois instrumentos, o modelo fosse capaz de captar nuances não aprendidas na tarefa multiclasse.

**Resultados e Análise**

Apesar da motivação estratégica por trás da criação de um modelo binário dedicado à distinção entre *drum* e *violin*, os resultados não alcançaram as expectativas. O modelo obteve uma **acurácia global de apenas 46%**, com desempenho equilibrado, mas **fraco em ambas as classes**.

* **Drum:** recall de 0.60 indica que o modelo consegue identificar a maioria das amostras reais de drum, mas a precisão (0.47) revela que muitas amostras classificadas como drum eram, na verdade, violin.
* **Violin:** o modelo teve ainda mais dificuldade. O recall de 0.31 mostra que a maioria dos violinos foi incorretamente classificada como drum, o que é consistente com os problemas enfrentados na abordagem multiclasse.

A matriz de confusão reforça esse diagnóstico:

* 333 amostras de drum foram classificadas como violin;
* 582 amostras de violin foram classificadas como drum.

A blue squares with white text

AI-generated content may be incorrect.

Esses números confirmam uma **forte sobreposição** nas representações das duas classes, mesmo quando isoladas em um cenário mais simples. As possíveis causas incluem:

* **Limitações dos MFCCs**: essa feature capta bem o conteúdo espectral estático, mas pode ser insuficiente para distinguir instrumentos com padrões temporais e articulações distintas, como a percussividade do drum versus o timbre contínuo e vibrato do violin.
* **Gravações ruidosas ou inconsistentes no dataset**: se houver variações muito grandes nas condições de gravação (ambiente, intensidade, técnica), o modelo pode ter aprendido mais o “ruído” do que o “instrumento”.
* **Similaridade no domínio de representação**: embora os instrumentos sejam diferentes ao ouvido humano, as características acústicas captadas por MFCC em curto prazo podem não ser tão discriminativas entre essas duas categorias.

Esses resultados mostram que simplesmente separar drum e violin em um classificador binário **não foi suficiente** para superar a dificuldade de distinção. Estratégias mais sofisticadas serão necessárias para lidar com essa limitação.

## 3.4 Predição Final com Refinamento por Modelo Especializado

Mesmo com o desempenho limitado do modelo binário para distinguir *drum* e *violin*, decidimos incorporá-lo a uma etapa final de **refinamento da predição**, com o objetivo de testar se a especialização poderia contribuir para decisões mais precisas nesses casos específicos.

A lógica da predição foi estruturada da seguinte forma:

1. **Predição principal:** o modelo multiclasse faz a predição inicial sobre todas as classes (guitar, piano, drum e violin), gerando uma distribuição de probabilidades softmax para cada amostra.
2. **Identificação de casos ambíguos:** para cada amostra cujo rótulo predito inicialmente foi *drum* ou *violin*, o sistema aciona uma segunda etapa.
3. **Refinamento com modelo binário:** essas amostras são processadas novamente, agora pelo modelo binário especializado, que estima a probabilidade da amostra pertencer à classe *violin* (e implicitamente, à *drum*, via 1 - probabilidade).
4. **Substituição seletiva das probabilidades:** os valores correspondentes às classes *drum* e *violin* na predição original são substituídos pelas novas probabilidades fornecidas pelo modelo binário. As demais classes permanecem inalteradas.
5. **Renormalização:** a distribuição de saída final é renormalizada para manter a coerência com o formato softmax.

Essa abordagem híbrida, ainda que não tenha resolvido completamente os desafios de diferenciação entre as duas classes, permitiu observar se havia ganhos marginais em termos de precisão ao combinar especialização e generalização.

**Resultados Finais com Predição Refinada**

A aplicação da etapa de refinamento com o modelo binário resultou em uma **leve melhoria nos resultados globais**, especialmente na **classificação da classe violin**, que foi consistentemente o maior desafio do projeto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classe | Precision | Recall | F1-score | Support |
| drum | 0.53 | 0.3 | 0.38 | 840 |
| guitar | 0.99 | 0.99 | 0.99 | 840 |
| piano | 0.99 | 0.99 | 0.99 | 635 |
| violin | 0.51 | 0.73 | 0.6 | 840 |
|  |  |  |  |  |
| macro avg | 0.76 | 0.75 | 0.74 | 3155 |
| weighted avg | 0.74 | 0.74 | 0.73 | 3155 |

Os principais destaques foram:

* **Acurácia geral:** aumentou ligeiramente para 74%, mantendo o desempenho elevado nas classes **guitar** e **piano** (ambas com F1-score de 0.99).
* **Violin:** houve **melhora substancial** no desempenho, com recall subindo de 0.02 (modelo multiclasse) para 0.73, e F1-score de 0.60. Isso indica que o modelo binário conseguiu recuperar muitas instâncias de violin que anteriormente estavam sendo classificadas incorretamente como drum.
* **Drum:** apresentou **queda no recall** (de 0.98 para 0.30) e desempenho mais modesto (F1-score de 0.38). Isso ocorreu possivelmente porque o modelo binário começou a classificar incorretamente amostras legítimas de drum como violin, ao tentar compensar a subdetecção anterior.

A matriz de confusão mostra esse efeito de forma clara: agora o modelo consegue capturar corretamente a maioria dos violinos (616 de 840), mas ao custo de reduzir a acurácia em *drum* (apenas 251 acertos de 840).

A diagram of a guitar instrument

AI-generated content may be incorrect.

Essa troca evidencia o comportamento típico de modelos compensatórios: a introdução de uma correção para um problema específico pode acabar prejudicando a performance em outra parte do modelo. Mesmo assim, o ganho significativo em violin justifica o experimento, principalmente em contextos onde a detecção de violino for crítica.

Em resumo, **a predição refinada representou um avanço significativo na classificação do violin**, e pode servir como base para estratégias mais robustas no futuro, como comitês de modelos, votações ponderadas ou uso de metamodelos para roteamento de decisões.

# 4. Conclusão e Próximos Passos

Este projeto proporcionou uma jornada prática e técnica por diferentes abordagens de classificação de áudio utilizando redes neurais. A tarefa de identificar instrumentos musicais com base em trilhas curtas revelou-se desafiadora, mas também rica em aprendizados.

Inicialmente, utilizamos espectrogramas de Mel combinados com redes convolucionais (CNNs). Embora essa combinação seja comum em tarefas de áudio, ela apresentou limitações no nosso caso: alto custo computacional, necessidade de cortar os áudios (prejudicando a continuidade rítmica), e baixa performance, com colapso de predição em apenas uma classe. Esse resultado demonstrou a importância de considerar o impacto do pré-processamento na representação dos dados e na capacidade de generalização do modelo.

A segunda abordagem, baseada em MFCCs e redes densas, mostrou-se mais eficiente e produtiva. A representação MFCC capturou com leveza e precisão boa parte das características relevantes dos instrumentos, especialmente para as classes *guitar*, *piano* e *drum*. A adoção de técnicas como blocos residuais, normalização, dropout e função de perda Focal Loss permitiu melhorar a estabilidade e a capacidade de aprendizado da rede, resultando em um modelo robusto e com desempenho competitivo.

Contudo, a classe *violin* se manteve desafiadora ao longo de todas as abordagens. Mesmo com a tentativa de especialização via modelo binário (drum vs. violin), os ganhos foram parciais e vieram acompanhados de perdas em *drum*. A aplicação de um mecanismo de predição refinada — que combinou o modelo principal com o binário — ajudou a equilibrar os resultados, aumentando significativamente o recall em *violin*, ainda que com sacrifício de precisão em *drum*. Essa estratégia híbrida ilustra o potencial de combinar modelos para alcançar resultados mais robustos em cenários desbalanceados.

**Lições Aprendidas**

* A escolha da representação do áudio tem grande impacto no desempenho e na viabilidade da arquitetura;
* Nem sempre soluções mais complexas (como CNNs com espectrogramas) resultam em melhores desempenhos;
* Estratégias simples, bem ajustadas e com preprocessing adequado (MFCC + dense + focal loss) podem superar modelos mais pesados;
* Técnicas como *early stopping*, *redução de learning rate* e *dropout* foram fundamentais para estabilizar o treinamento e evitar overfitting;
* A abordagem modular — combinando modelos para cenários específicos — pode ser promissora em tarefas com classes com características sobrepostas.

**Próximos Passos e Recomendações**

Para evolução futura deste trabalho, sugerimos explorar as seguintes direções:

* **Modelos especializados para áudio sequencial**, como **CRNNs** (convolucional + recorrente) ou **Transformers para áudio**, que capturam melhor padrões temporais de longa duração;
* **Pré-processamento avançado**, incluindo técnicas de *noise reduction*, *equalização* e análise de silêncios para eliminar partes irrelevantes dos áudios;
* **Curadoria aprofundada do dataset**, com verificação de ruídos, consistência entre classes e separação de múltiplos instrumentos por faixa;
* **Uso de embeddings pré-treinados**, como **YAMNet** ou **VGGish**, que oferecem representações de alto nível treinadas em grandes coleções de áudio e podem reduzir o custo de extração de features;
* **Experimentação com funções de perda mais sofisticadas**, como *adaptive focal loss*, *label smoothing* ou *class-balanced loss*, para lidar melhor com desbalanceamentos e ambiguidade entre classes;
* **Testes com data augmentation mais inteligentes**, baseados em modelos generativos, para criar amostras sintéticas de classes minoritárias.

Este projeto reforça a importância de alinhar a complexidade do modelo ao problema real, considerando tanto os aspectos computacionais quanto a natureza dos dados. A capacidade de iterar, analisar os erros e adaptar as abordagens foi o diferencial para evoluir da primeira solução fracassada até um modelo funcional e promissor.