UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI

Projeto Final - TINYML

FrogSoundGuard – Identificação de sons de sapos para proteção de animais domésticos

Afonso Henriques Massunari - 2024002176
Henrique Beraldo Vilela Marques - 2021011627
Thiago Batista Araújo - 2021032968

ITAJUBÁ

2025

Sumário

RESUMO

Este projeto, desenvolvido na disciplina de Aprendizado de Máquina Aplicado a Dispositivos IoT Embarcados (TinyML) da Universidade Federal de Itajubá (UNIFEI), tem como objetivo classificar sons entre duas classes: vocalizações de sapos (*frog*) e sons de ambiente (*ambient*). A motivação está na presença de espécies de sapos venenosos, cuja detecção automática pode contribuir para a prevenção de acidentes. Foram utilizados os conjuntos de dados Noise Audio Dataset e Anuran Sound Dataset. Após o pré-processamento dos áudios e extração de características com o método MFE (Mel Frequency Energy), foi treinado um modelo leve de classificação. O modelo final foi otimizado para execução em dispositivos embarcados, demonstrando viabilidade para detecção em tempo real em ambientes com recursos limitados.

1 Introdução

Neste projeto, exploramos a implementação de um sistema de classificação de sons utilizando técnicas de TinyML em uma plataforma embarcada. O objetivo foi desenvolver uma rede neural leve e eficiente, capaz de distinguir entre vocalizações de sapos (frog) e sons de ambiente (ambient). O sistema foi desenvolvido utilizando a placa **Arduino Nano 33 BLE Sense**, um microcontrolador de baixo consumo com sensores integrados e suporte a aplicações de aprendizado de máquina.

Para o treinamento do modelo, foram utilizados dois conjuntos de dados: o **Anuran Sound Dataset**, que contém gravações de vocalizações de sapos, e o **Noise Audio Dataset**, composto por diversos sons ambientais. Os dados foram tratados para uniformizar a taxa de amostragem em **16 kHz**, garantindo compatibilidade com o pipeline de processamento de áudio embarcado. As características dos sinais foram extraídas por meio do método **Mel Frequency Energy (MFE)**, adequado para representar informações relevantes em tarefas de reconhecimento acústico.

O treinamento e a implantação do modelo foram realizados na plataforma Edge Impulse, permitindo a implementação em tempo real em dispositivos com recursos computacionais limitados. A motivação do projeto está relacionada à presença de espécies de sapos venenosos em certas regiões, cujas vocalizações podem representar um risco para animais domésticos, especialmente cães, que correm o risco de envenenamento ao entrarem em contato com esses sapos.

Este projeto visa mitigar esse risco ao identificar sons de anuros em tempo real, utilizando um modelo de **Machine Learning** embarcado em microcontroladores de baixo consumo. Baseando-se no dataset disponível publicamente **Anuran Sound Dataset** (Kaggle), o objetivo é treinar um modelo de classificação de sons capaz de identificar vocalizações características de sapos, com foco nas espécies encontradas no Brasil, ou que apresentem sonorização semelhante àquelas encontradas no país. Como o dataset base contém apenas três espécies encontradas no Brasil, a equipe considera expandir o escopo, incluindo sons de outros animais comuns em zonas rurais, como galinhas, porcos, grilos, entre outros. Essa expansão visa não apenas melhorar a precisão do modelo, mas também enriquecer sua capacidade de generalização, permitindo diferenciar sons não relacionados a sapos e, assim, reduzir falsos positivos e fornecer alertas mais assertivos.

2 Diagrama De Blocos do Sistema

Este diagrama de blocos, **Figura 1**, ilustra de forma clara e linear o fluxo de processamento de áudio em um sistema de detecção de sons, otimizado para TinyML.

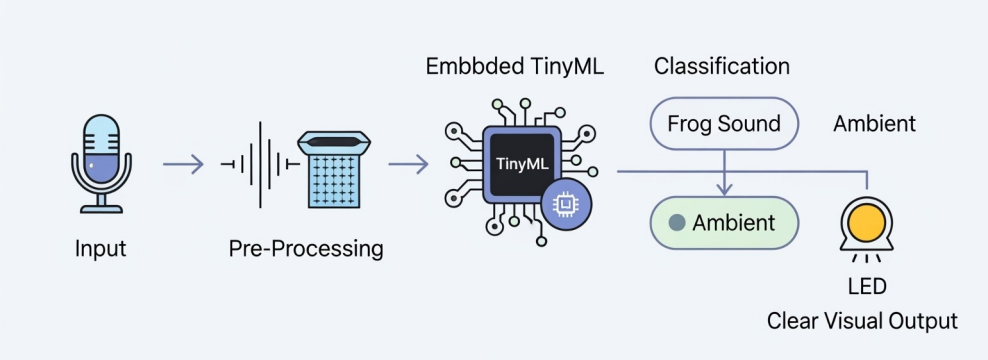

Figura 1: Diagrama de blocos do sistema.

1. **Microfone (Entrada)**: O processo inicia-se com o microfone, responsável por captar os sons do ambiente. Este componente atua como ponto de entrada do sinal de áudio no sistema.
2. **Pré-processamento de Áudio**: O sinal captado é encaminhado para a etapa de pré-processamento, onde o áudio bruto é tratado e características relevantes são extraídas. Nessa fase, emprega-se o método *Mel Frequency Energy* (MFE), que prepara os dados de forma eficiente para a análise subsequente.
3. **Modelo TinyML Embarcado**: Em seguida, os dados pré-processados são enviados para o modelo TinyML embarcado — o "cérebro" do sistema. Trata-se de uma rede neural leve, otimizada para operar em dispositivos com recursos computacionais restritos. Essa etapa realiza a classificação do som em tempo real.
4. **Classificação (Som de Sapo / Ambiente)**: O bloco de classificação exibe o resultado gerado pelo modelo, indicando se o som detectado corresponde a vocalizações de sapos (*Frog Sound*) ou a ruídos ambientais (*Ambient*). Essa distinção é feita internamente pelo modelo e disponibilizada de forma explícita ao usuário.
5. **Alerta via LED (Saída Visual)**: Por fim, o resultado da classificação aciona o alerta visual por LED. Este componente fornece uma indicação imediata e clara da detecção, sinalizando, por exemplo, a presença de sapos no ambiente por meio de um padrão específico de iluminação.

3 Hardware Utilizado

A **Figura 2** apresenta os principais componentes de hardware empregados na implementação do sistema de detecção de sons otimizado para TinyML:

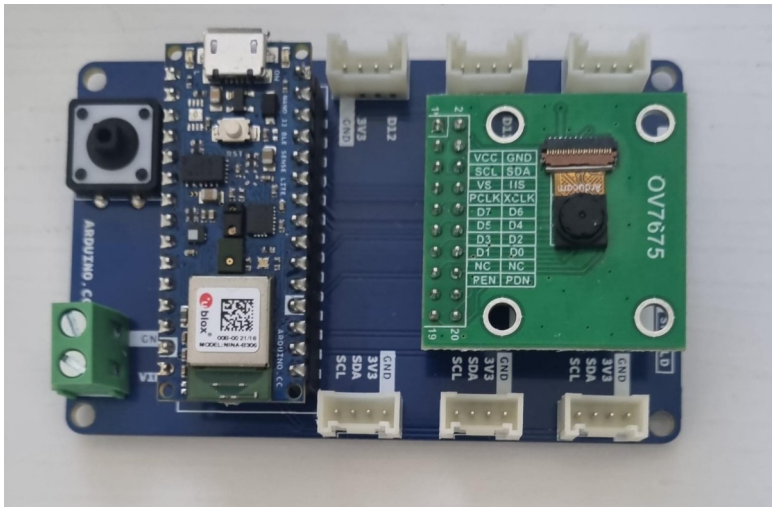

Figura 2: Hardware Utilizado.

- **Arduino Nano 33 BLE Sense**: Placa de desenvolvimento compacta, equipada com microcontrolador ARM Cortex-M4 e conectividade Bluetooth Low Energy (BLE). Possui sensores integrados, incluindo um microfone digital, além de oferecer suporte nativo para aplicações de TinyML.
- **Tiny Machine Learning Shield**: Placa de expansão projetada para facilitar o desenvolvimento de aplicações TinyML. Disponibiliza interfaces adicionais e otimiza o processamento de tarefas de aprendizado de máquina embarcado.
- **Microfone MP34DT05**: Sensor de áudio digital MEMS de alta sensibilidade, responsável por captar sons do ambiente com precisão. Sua interface digital I²S permite uma transmissão de sinal com baixo ruído.
- **LED RGB**: Elemento de saída visual que fornece uma indicação clara do resultado da classificação. Por meio da variação de cores, o LED RGB sinaliza diferentes estados do sistema — por exemplo, uma cor específica para indicar a detecção de sons de sapo e outra para sons ambientes.

4 Metodologia

Este projeto foi realizado com base nas aulas ministradas pelo professor da disciplina. Todos os materiais utilizados, incluindo vídeos, slides e recursos técnicos, foram disponibilizados pela universidade para fins educacionais.

4.1 Aquisição e Tratamento de dados

A **Figura 3** apresenta o fluxo geral de aquisição, pré-processamento e preparação dos dados utilizados no treinamento do classificador TinyML. Todo o processo foi cuidadosamente estruturado para garantir a qualidade, padronização e balanceamento do conjunto de dados.

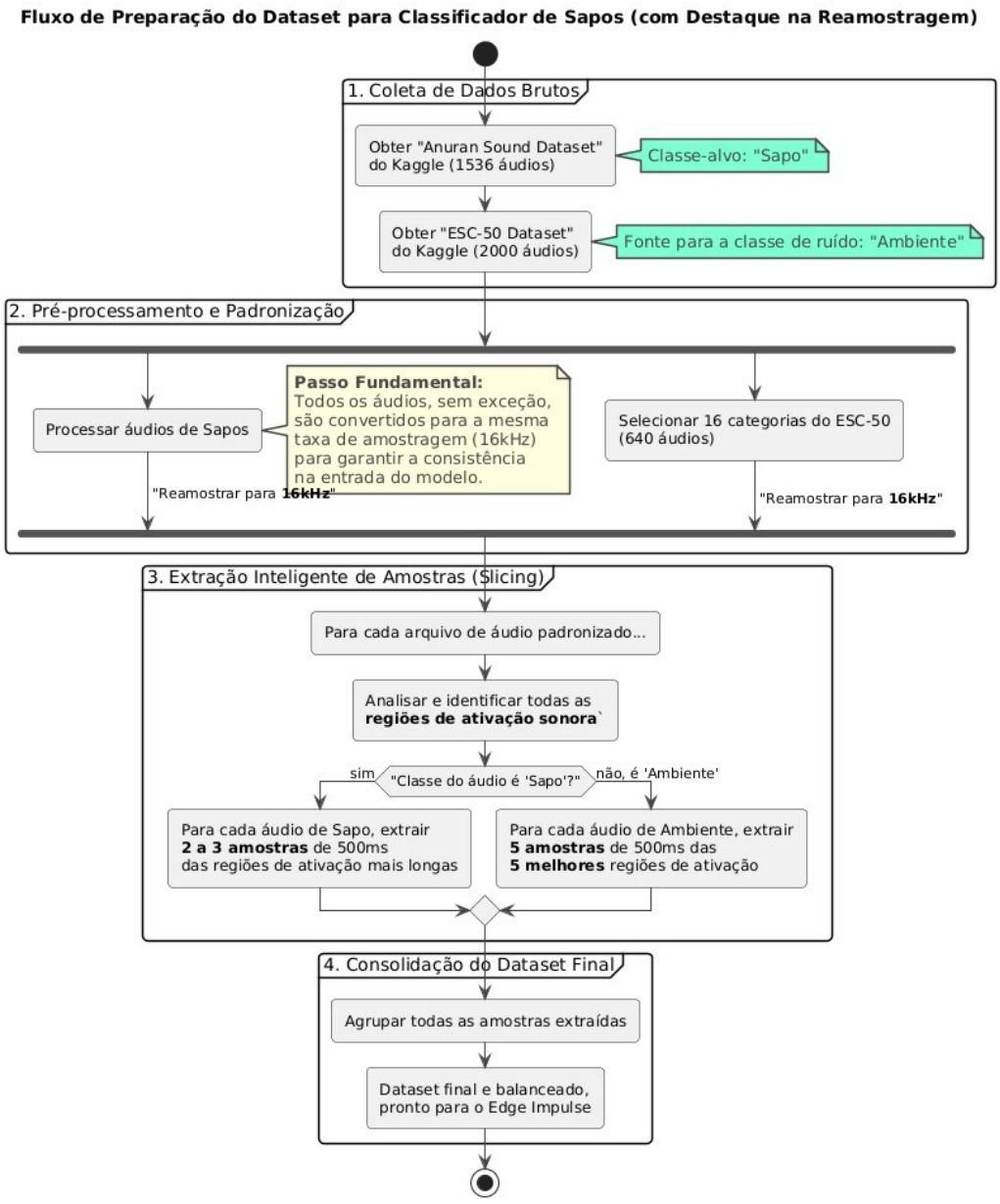

Figura 3: Fluxo de Preparação do Dataset Para o Classificador de Sapos.

4.1.1 Obtenção e Padronização dos Datasets de Origem

Foram utilizados dois conjuntos de dados públicos como fonte primária, cuidadosamente selecionados para garantir diversidade, qualidade e relevância dos sons para o problema proposto:

- **Anuran Sound (Frogs or Toads)**: 1.536 gravações de 26 espécies de anuros, formando a base da classe-alvo *"Sapo"*.
- **Noise Audio Data (ESC-50)**: 2.000 gravações de sons ambientais, das quais foram selecionadas categorias para compor a classe *"Ambiente"*.

Todos os arquivos passaram por uma padronização obrigatória:

- Conversão para o formato **.wav**.
- Reamostragem para **16.000 Hz**, taxa adequada para capturar frequências relevantes, mantendo o equilíbrio entre qualidade e tamanho dos arquivos.

4.1.2 Estruturação das Classes para Classificação

O problema foi modelado como uma **classificação binária** com duas classes bem definidas:

- **Classe "Sapo"**: 1.536 gravações do *Anuran Sound*.
- **Classe "Ambiente"**: Seleção de 16 categorias do *ESC-50*, totalizando 640 arquivos que representam sons comuns na natureza e possíveis interferências sonoras.

4.1.3 Extração de Amostras (Slicing)

A etapa mais crítica do pré-processamento consistiu na divisão inteligente dos áudios para criar amostras curtas e relevantes:

- **Identificação de Regiões de Ativação Sonora**:

  O processo para gerar os datasets de áudio desejados foi executado em etapas bem definidas, utilizando um método de extração inteligente para isolar os sons mais relevantes. Inicialmente, o ambiente de trabalho foi preparado com a criação dos diretórios de saída, FROG_SAMPLES_DIR e AMBIENT_SAMPLES_DIR, através da função os.makedirs.
  Para cada arquivo de áudio original, o primeiro passo foi carregá-lo e padronizá-lo para uma taxa de amostragem de 16kHz usando a função librosa.load. O núcleo da extração consistiu em identificar os momentos de maior energia sonora, em vez de fatiar o áudio aleatoriamente. Para isso, a função scipy.signal.find_peaks foi aplicada sobre o valor absoluto do sinal de áudio (obtido com np.abs) para localizar os picos de volume. Essa busca foi refinada para garantir que os picos fossem proeminentes e tivessem uma distância mínima entre si, evitando a captura de ruídos irrelevantes ou do mesmo evento sonoro várias vezes. Na **Figura 4**, é apresentado um exemplo do processo de identificação de picos e seleção de amostras.

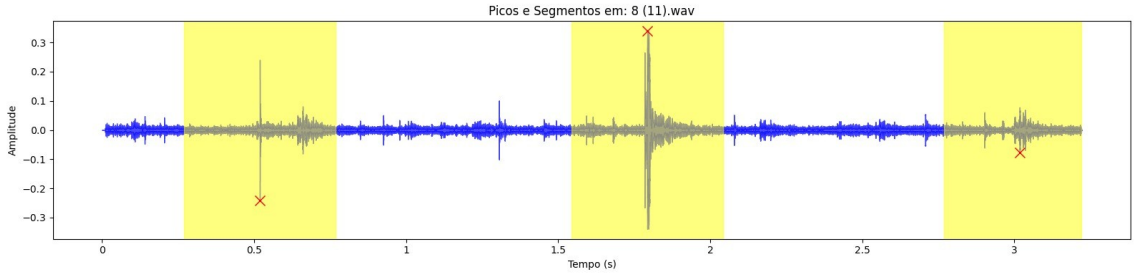

Figura 4: Exemplo de identificação de picos e seleção de amostras.

- **Amostragem por Classe**:
  - **Ambiente**: Para cada um dos 640 áudios, foram extraídas **5 amostras de 500ms**, selecionadas a partir dos segmentos mais proeminentes.
  - **Sapo**: De cada gravação, foram extraídas de **2 a 3 amostras de 500ms**, priorizando os trechos de vocalizações mais longos.

Essa estratégia permitiu:

- **Balancear o Dataset**: Garantiu números de amostras próximos entre as classes, evitando viés do modelo.
- **Focar no Sinal Relevante**: Treinar o modelo com trechos que realmente contêm os sons de interesse.
- **Padronizar as Entradas**: Gerar amostras curtas, uniformes, otimizando a arquitetura do modelo no Edge Impulse.

4.1.4 Consolidação e Divisão do Dataset Final

Após todo o processamento e refinamento, o conjunto final ficou definido da seguinte forma, com dados balanceados e prontos para o treinamento do modelo:

- **3.162 amostras de 500ms da classe "Sapo"**
- **3.083 amostras de 500ms da classe "Ambiente"**

Para o treinamento e validação do modelo na plataforma Edge Impulse, o dataset foi dividido em **80% para treinamento** e **20% para teste**, assegurando uma avaliação justa do desempenho do classificador. Essa divisão está ilustrada na **Figura 5**.

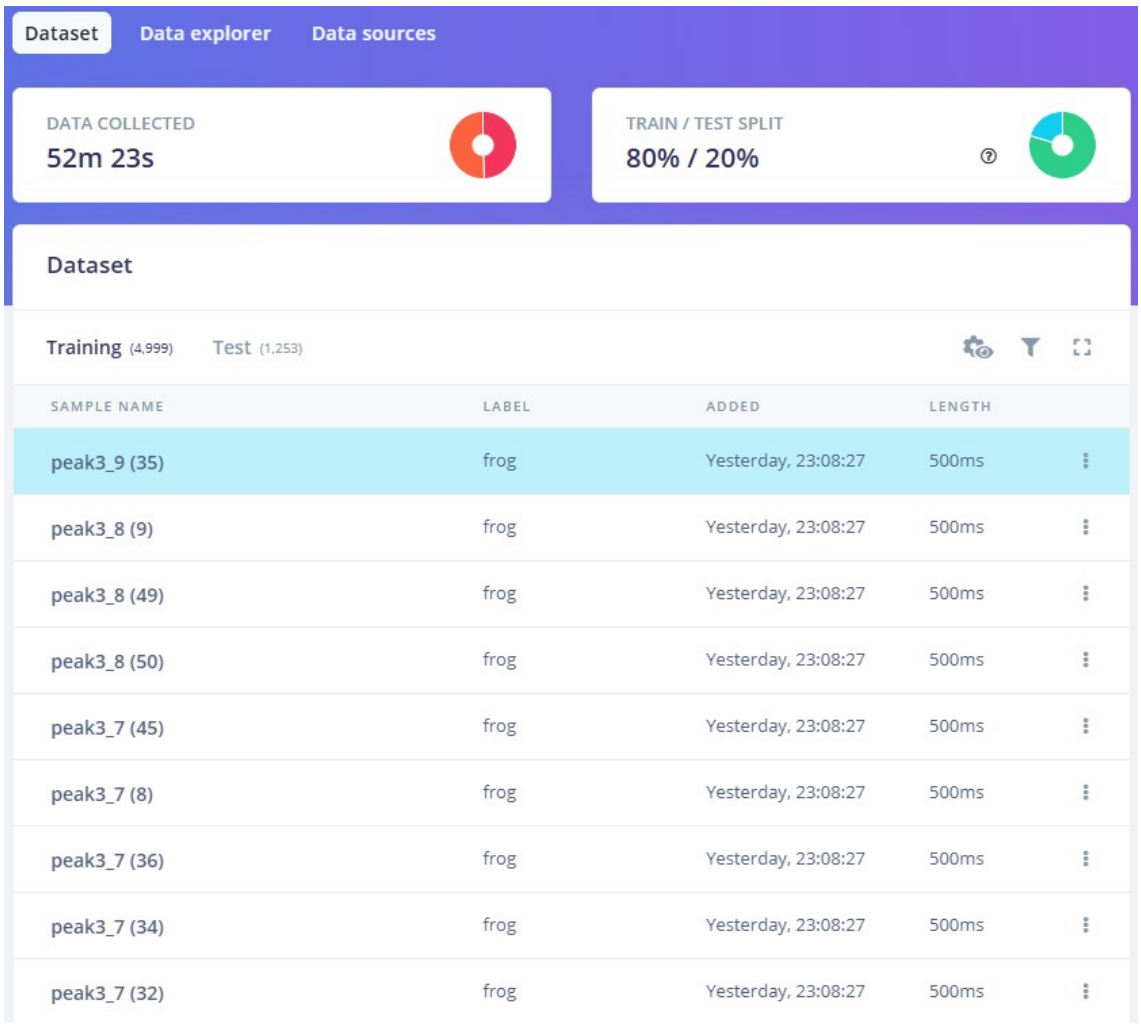

Figura 5: Divisão do dataset final balanceado em conjuntos de treinamento e teste para o classificador TinyML.

4.2 Configuração do Impulso

O impulso foi configurado no Edge Impulse para classificar dados de áudio em séries temporais amostrados a 16.000 Hz. Foi utilizada uma janela de 500 ms com incremento (stride) também de 500 ms, e o preenchimento com zeros (zero-padding) foi habilitado para garantir comprimento uniforme das entradas. O bloco de processamento **Mel Frequency Energy (MFE)** foi aplicado para extrair características relevantes do sinal de áudio. Essas características foram então alimentadas em um bloco de aprendizado para classificação, projetado para distinguir entre duas classes de saída: **ambient** e **frog**.

4.3 Pré-Processamento

A extração das características de áudio foi realizada utilizando o método Mel Frequency Energy (MFE), que emprega um banco de filtros Mel para capturar as propriedades espectrais relevantes do sinal. Os principais parâmetros configurados para o processamento foram:

- **Frame length:** 0,025 segundos

- **Frame stride:** 0,01 segundos
- **Número de filtros (Filter number):** 41
- **Comprimento da FFT (FFT length):** 512 pontos
- **Frequência mínima (Low frequency):** 80 Hz
- **Frequência máxima (High frequency):** não definida
- **Normalização:** aplicada com nível de ruído (Noise floor) ajustado para -91 dB

Essas configurações permitem uma representação eficiente do áudio, focando nas faixas de frequência mais relevantes para a classificação entre sons de sapos e ambiente.

A **Figura 6** mostra um exemplo das energias do banco de filtros Mel (Mel Energies) extraídas de um áudio. Essa representação destaca as variações espectrais usadas como entrada para o modelo de classificação entre sons de sapos e ambiente.

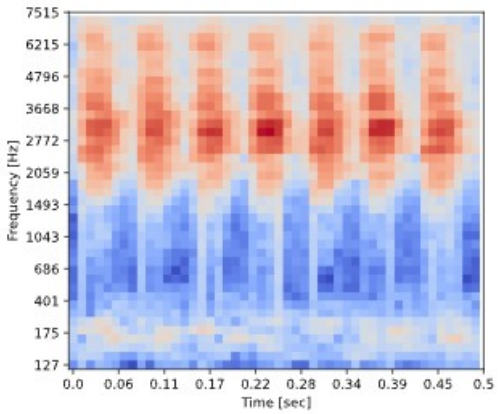

Figura 6: Exemplo do resultado do processamento DSP mostrando as energias Mel extraídas de um áudio.

Os resultados desse processo foram usados para gerar um mapa de características que visualiza a distribuição das amostras de áudio, conforme mostrado na **Figura 7**.

Figura 7: Visualização das características extraídas (Mel Frequencies) utilizada para análise e treinamento do modelo.

4.4 Arquitetura do modelo

O modelo foi treinado por 100 épocas utilizando uma taxa de aprendizado (learning rate) de 0,005 em um processador CPU. Os dados de treinamento foram divididos em 80% para treino e 20% para validação, baseando-se em chaves de metadados.

A arquitetura da rede neural, ilustrada na **Figura 8**, consistiu em uma camada de entrada com 1.968 características, seguida por uma camada de reshape que organizou as características em 41 colunas. Em seguida, foram aplicadas duas camadas convolucionais 1D com pooling, utilizando 8 e 16 filtros respectivamente, ambos com tamanho de kernel 3. Cada camada convolucional foi seguida por uma camada de dropout com taxa de 0,25 para evitar overfitting. Após a camada de flatten, a camada de saída classificou as entradas em duas classes: **ambient** e **frog**.

Figura 8: Arquitetura da Rede neural.

O modelo mais recente alcançou uma acurácia de **93,5%** no conjunto de validação, com um valor de perda (**loss**) de **0,15**. A **matriz de confusão**, apresentada na **Figura 9**, mostra um desempenho consistente na classificação das duas classes. Para a classe **ambient**, 97,8% das amostras foram corretamente classificadas, enquanto 2,2% foram incorretamente identificadas como **frog**. Já para a classe **frog**, 89,4% das amostras foram corretamente reconhecidas, com 10,6% classificadas erroneamente como **ambient**.

Os **F1-scores** foram **0,94** para a classe *ambient* e **0,93** para a classe *frog*, indicando um bom equilíbrio entre precisão e recall. As métricas médias ponderadas no conjunto de validação também foram satisfatórias, com **Área sob a Curva ROC (AUC)** de **0,94**, **Precisão média ponderada** de **0,94**, **Revocação média ponderada** de **0,94** e **F1-score médio ponderado** de **0,93** — demonstrando um desempenho robusto do modelo em ambas as classes.

	AMBIENT	FROG
AMBIENT	97.8%	2.2%
FROG	10.6%	89.4%
F1 SCORE	0.94	0.93

Figura 9: Matriz de Confusão do conjunto de dados de validação

A **Figura 10** apresenta o **Explorador de Dados** com as amostras do conjunto de treinamento, destacando visualmente os acertos e erros do modelo.

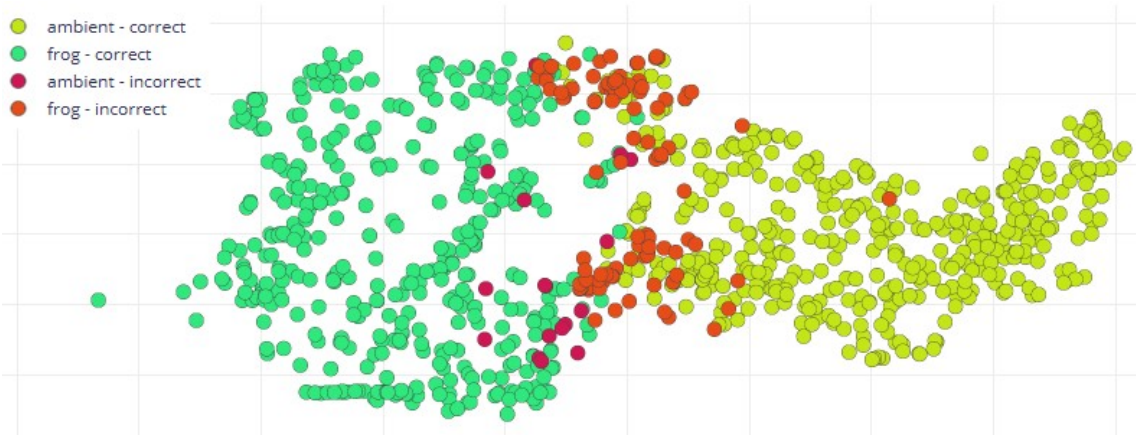

Figura 10: Explorador de dados do conjunto completo de treinamento.

4.5 Teste do Modelo

Após o treinamento, o modelo foi avaliado com o conjunto de teste separado, atingindo uma acurácia geral de **90,76%**, o que demonstra sua capacidade de diferenciar vocalizações de sapos de ruídos de fundo de forma confiável.

As métricas detalhadas foram:

- **Área sob a Curva ROC (AUC)**: 0,94
- **Precisão (média ponderada)**: 0,95
- **Revocação (média ponderada)**: 0,94
- **F1-Score (média ponderada)**: 0,94

A **Figura 11** apresenta a matriz de confusão, destacando o desempenho por classe:

- *"Ambiente"*: 91,5% de acertos, 1,5% de falsos positivos para *"Sapo"* e 7,0% como incertos.
- *"Sapo"*: 90,0% de acertos, 3,2% de falsos positivos para *"Ambiente"* e 6,8% como incertos.
- F1-Score por classe: **0,94** para ambas.

	AMBIENT	FROG	UNCERTAIN
AMBIENT	91.5%	1.5%	7.0%
FROG	3.2%	90.0%	6.8%
F1 SCORE	0.94	0.94	

Figura 11: Matriz de confusão do modelo no conjunto de teste.

Além disso, a **Figura 12** mostra o **Explorador de Características**, que permite visualizar como as amostras das duas classes se distribuem no espaço de características extraídas, ajudando a validar a separabilidade dos dados no modelo final.

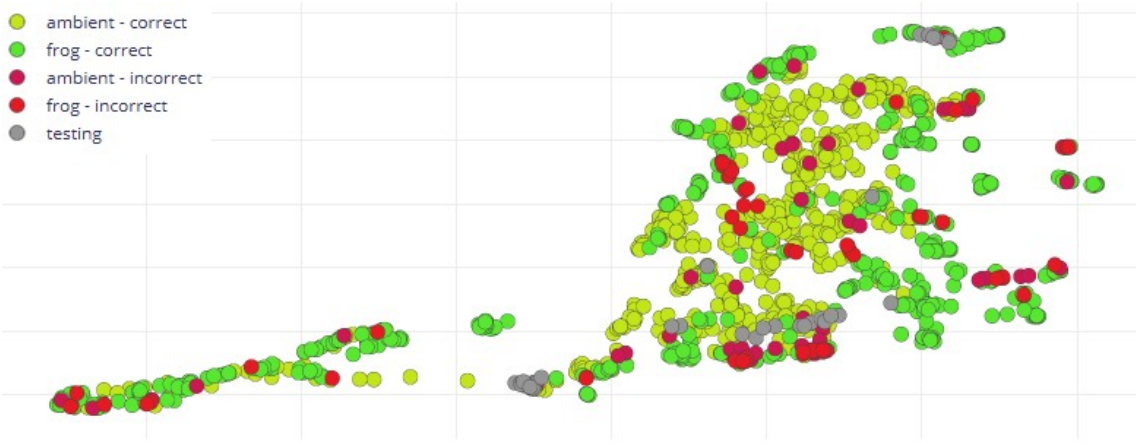

Figura 12: Distribuição das amostras no explorador de características.

4.6 Implantação do modelo (Deployment)

Para a implantação, o modelo treinado foi exportado como uma biblioteca Arduino compatível com placas ARM, incluindo o Arduino Nano 33 BLE. Foram aplicadas otimizações com o compilador EON™ da Edge Impulse, mantendo a acurácia e reduzindo o uso de RAM em 29% e de memória flash em 48%. O modelo foi quantizado para int8, diminuindo o uso de memória e possibilitando inferência em tempo real em sistemas embarcados com recursos limitados.

No Arduino Nano 33 BLE, o pipeline completo — extração de características MFE e classificação — apresentou latência total de 130 ms (125 ms MFE + 5 ms classificação). O uso de RAM ficou em cerca de 11,8 KB, e o classificador ocupou 30,9 KB de flash. O modelo manteve alta acurácia, demonstrando desempenho eficiente e confiável no dispositivo embarcado.

5 Resultados

Nos testes realizados diretamente na placa Nano 33 BLE, os resultados foram satisfatórios e estiveram alinhados com as expectativas.

O modelo classificou os sons de sapos (*frogs*) com uma porcentagem de predição variando entre **70% e 95%**, enquanto a classe ambiente (*ambient*) foi identificada com níveis de predição entre **85% e 99%**. Esses resultados, ilustrados na **Figura 13**, demonstram a capacidade do modelo em distinguir as duas classes com alta confiança.

```
Recording done
Predictions (DSP: 1695 ms., Classification: 10 ms., Anomaly: 0 ms.):
    ambient: 0.14453
    frog: 0.85547
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 1697 ms., Classification: 10 ms., Anomaly: 0 ms.):
    ambient: 0.99219
    frog: 0.00781
```

Figura 13: Porcentagens de predição do modelo para as classes "Sapo" (frog) e "Ambiente" (ambient) durante os testes reais na placa Nano 33 BLE.

6 Conclusão

Neste projeto, foi desenvolvido e validado um sistema de classificação de sons baseado em TinyML, capaz de distinguir vocalizações de sapos e sons ambientais em tempo real, utilizando hardware embarcado de baixo consumo. A combinação da extração de características por meio do método Mel Frequency Energy (MFE) e o treinamento na plataforma Edge Impulse mostrou-se adequada para viabilizar uma solução compacta e eficiente de monitoramento acústico.

Embora o dataset empregado inclua gravações de diversas espécies, inclusive algumas comuns no Brasil, ainda há limitações quanto à quantidade de amostras específicas de certas espécies locais. Além disso, novos cenários de captação podem apresentar variações sonoras não contempladas nas gravações originais. Por isso, a coleta e inclusão de novas amostras de campo são recomendadas para aumentar a robustez do modelo, reduzir falsos positivos e melhorar a generalização em ambientes reais.

Apesar desses pontos de atenção, os resultados obtidos demonstraram que o modelo foi eficaz em cumprir o propósito para o qual foi projetado, apresentando bom desempenho na identificação de vocalizações de sapos com alta confiança, mesmo em um dispositivo embarcado com recursos computacionais limitados. Assim, o sistema desenvolvido representa uma base sólida para aplicações de monitoramento ambiental, com potencial para expansão e aprimoramento contínuo para atender demandas específicas do contexto brasileiro.

7 Referências

BAYİN, Mehmet. *Anuran Sound (Frogs or Toads) Dataset*. Kaggle. Disponível em: https://www.kaggle.com/datasets/mehmetbayin/anuran-sound-frogs-or-toads-dataset/data. Acesso em: 29 jun. 2025.

PAVITHRA, D.; BALACHANDAR, R. A real-time frog sound classification using improved deep convolutional neural networks. *Expert Systems with Applications*, v. 222, 119557, 2023. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417423005912. Acesso em: 29 jun. 2025.

REDDI, Vijay Janapa. *Machine Learning Systems: Principles and Practices of Engineering Artificially Intelligent Systems*. Harvard University, 2025.

ROVAI, Marcelo. UNIFEI - TinyML - IESTI01 2022.1 - Class 24 - KWS and Edge Impulse Studio. [Vídeo]. YouTube, 2021. Disponível em: https://www.youtube.com/watch?v=-4BqS_KMJtc&list=PLwmX8w17Zuqb4RX9L-xcjNYgZYYj_84To&index=28. Acesso em: 29 jun. 2025.

TOSHQORGONOV, Javohir. *Noise Audio Data (ESC-50)*. Kaggle. Disponível em: https://www.kaggle.com/datasets/javohirtoshqorgonov/noise-audio-data. Acesso em: 29 jun. 2025.

APÊNDICE

O projeto completo pode ser acessado pelo Edge Impulse Studio no seguinte link: https://studio.edgeimpulse.com/public/732300/live

O processo de aquisição e tratamento de dados pode ser acessado por meio dos seguintes links:
https://colab.research.google.com/drive/1xhx4RFAM15ddUKf0M4hm1jB-SXeeMCul?usp=sharing
ou
https://github.com/thibaraujo/df_frog_ambient

Para este projeto, foram utilizados dois conjuntos de dados públicos disponíveis na plataforma Kaggle:

- **Anuran Sound (Frogs or Toads)**:

  https://www.kaggle.com/datasets/mehmetbayin/anuran-sound-frogs-or-toads-dataset/data

- **Noise Audio Data (ESC-50)**:

  https://www.kaggle.com/datasets/javohirtoshqorgonov/noise-audio-data

Um vídeo com a explicação do projeto e a apresentação de seus resultados pode ser acessado por meio do seguinte link:
https://www.youtube.com/watch?v=8DZ4IFw-Abk