Sistema AIoT para Detecção de Sirenes com Aprendizado de Máquina Embarcado em Dispositivos de Borda

Lucas Barreto, Levy Galvão, Raimundo Barreto, Horácio Oliveira
Instituto de Computação - Universidade Federal do Amazonas
Instituto de Pesquisas Eldorado
{lucas.barreto, levy}@eldorado.org.br
{rbarreto, horacio}@icomp.ufam.edu.br

***Resumo*—Este trabalho apresenta uma solução de AIoT para detecção de sirenes com inferência embarcada em microcontroladores, baseada em redes neurais convolucionais (CNNs) quantizadas. A solução proposta utiliza um microcontrolador com microfone digital integrado para captar sons do ambiente e realizar inferência local com redes neurais convolucionais otimizadas. Essa abordagem elimina a necessidade de conexão com a nuvem, reduzindo latência e aumentando a confiabilidade e a privacidade dos dados. O modelo foi treinado em espectrogramas gerados a partir de áudios de tráfego e sirenes e implantado utilizando TensorFlow Lite para microcontroladores. Os resultados demonstram acurácia de 96,83%, associada a uma precisão de 94,49%, validando a viabilidade de integrar IA embarcada em soluções IoT urbanas, como veículos autônomos e semáforos inteligentes.**

***Index Terms*—AIoT, Aprendizado de Máquina Embarcado, Detecção de Sirenes, Computação de Borda, IoT Urbana, Segurança no Trânsito.**

I. Introdução

Em áreas urbanas, veículos de emergência utilizam sirenes para alertar motoristas e pedestres em situações críticas como acidentes, incêndios e resgates. Entretanto, a eficácia dessas sirenes tem sido comprometida por fatores como isolamento acústico dos veículos modernos, buzinas, música alta e conversas, resultando em um ambiente sonoro altamente desafiador para sistemas automáticos de detecção [1].

Para enfrentar tais desafios, apresentamos uma abordagem baseada em **AIoT (Artificial Intelligence of Things)** que realiza a detecção de sirenes diretamente em microcontroladores de baixo consumo, dispensando comunicação com a nuvem e reduzindo a latência, aspecto fundamental em aplicações de segurança pública [2]. A proposta combina três eixos estratégicos da IoT moderna: *(i)* sensoriamento acústico local via microfone MEMS; *(ii)* processamento inteligente de sinais com redes neurais convolucionais leves; e *(iii)* computação de borda para respostas imediatas e eficiência energética.

O sistema foi implementado em um *Arduino Nano 33 BLE Sense Rev2*, capaz de executar modelos otimizados com *TensorFlow Lite* (TF Lite), tecnologia que viabiliza o movimento TinyML em dispositivos com recursos restritos [3], [4]. Ensaios experimentais demonstram que, mesmo sob tais restrições, atingimos acurácia superior a 96% na detecção de sirenes em tempo real.

O restante do artigo está organizado da seguinte forma: a Seção II aborda os fundamentos teóricos de computação de borda, sensoriamento acústico e TinyML; a Seção III analisa os trabalhos correlatos; a Seção IV descreve a metodologia, a coleta de dados e a arquitetura proposta; a Seção V apresenta os experimentos; a Seção VI apresenta a discussão dos resultados; e a Seção VII discute as conclusões e perspectivas futuras.

II. Fundamentação Teórica

A. Computação de Borda

A computação de borda (edge computing) desloca o processamento para próximo da fonte de dados, reduzindo latência, tráfego na rede e riscos de privacidade [5]. Para cenários de segurança urbana, onde cada milissegundo conta na liberação de uma rota para ambulâncias, a execução local é vital; estudos mostram que arquiteturas névoa-borda diminuem o tempo de resposta em sistemas de classificação sonora distribuída [2]. Neste trabalho, todo o pipeline de inferência ocorre no microcontrolador, atendendo aos requisitos de tempo-real do domínio.

B. Processamento de Áudio

A sirene apresenta padrões harmônicos bem definidos; representar essas variações no domínio tempo-frequência via espectrogramas facilita o aprendizado de redes neurais [1]. Após a amostragem (16 kHz) e quantização (16 bits), os sinais são convertidos em espectrogramas com janelas de 75 ms e sobreposição de 50 %. Essa configuração oferece boa resolução em frequência preservando o contexto temporal necessário para distinguir sirenes de buzinas ou motores.

C. Aprendizado de Máquina Embarcado

Técnicas profundas, como CNNs, fornecem alta acurácia em tarefas acústicas; porém, sua adoção em dispositivos restritos exige otimizações. O *TensorFlow Lite* foi projetado para rodar modelos em poucos kB de RAM, mantendo desempenho competitivo [3]. A comunidade TinyML demonstra que modelos

quantizados ($<$ 20 kB) são viáveis em microcontroladores de baixo consumo [4]. No sistema proposto, uma CNN com 3 camadas convolucionais e apenas 32 k parâmetros realiza inferência em $\approx$320 ms, dentro da janela aceitável para alertas ao motorista.

D. Inteligência Artificial das Coisas (AIoT)

AIoT integra inferência inteligente a dispositivos IoT, permitindo decisões rápidas, descentralizadas e energeticamente eficientes [6].

AIoT é a integração da Inteligência Artificial (IA) com a Internet das Coisas (IoT). Enquanto a IoT conecta dispositivos físicos à internet para coletar e transmitir dados, a IA analisa esses dados e toma decisões de forma inteligente e autônoma.

AIoT funciona da seguinte forma: IoT coleta dados em tempo real por meio de sensores, dispositivos e máquinas conectadas; IA processa esses dados usando algoritmos, modelos preditivos e aprendizado de máquina; e com isso, o sistema pode identificar padrões, prever comportamentos e agir automaticamente, sem intervenção humana constante.

O AIoT representa a evolução da IoT, tornando os dispositivos não apenas conectados, mas inteligentes e autônomos. É uma tecnologia central para áreas como indústria, saúde, transporte, agricultura e cidades inteligentes.

III. Trabalhos Correlatos

Nesta seção, revisamos e comparamos trabalhos que abordam a detecção de sons urbanos, sirenes ou classificação geral de eventos acústicos em dispositivos com recursos limitados. O objetivo é destacar lacunas que o sistema proposto preenche, especialmente no contexto de AIoT e aprendizado de máquina embarcado.

A. Detecção de sirenes em microcontroladores

Um sistema baseado em microcontrolador Atmel foi desenvolvido para detectar sirenes e ativar alertas visuais em navegadores assistivos [7], demonstrando eficácia em ambientes reais, mas sem o uso de redes neurais convolucionais nem otimizações via TinyML.

B. Classificação de som ambiente em microcontroladores

O projeto ESC-CNN implementou classificadores de som urbano no microcontrolador STM32L476, atingindo 70.9% de acurácia no dataset UrbanSound8K, utilizando menos de 50% dos recursos de CPU e RAM [8]. Apesar dos bons resultados, o trabalho não foca especificamente em sons de sirene nem em aplicações AIoT com resposta imediata.

C. Pipeline genérico para som ambiente em edge

Mohaimenuzzaman et al. propuseram a arquitetura ACDNet para compressão e quantização de CNNs aplicadas a classificação de som ambiental (ESC-10 e ESC-50), com acurácia próxima a 96% e inferência viável em microcontroladores [9]. A proposta é genérica e não contempla integração com infraestrutura urbana ou inferência com janela deslizante.

D. Modelos comerciais prontos para sirene

A empresa Imagimob lançou modelos comerciais para detecção de sirenes com execução na borda, voltados a aplicações automotivas e industriais [10]. No entanto, os detalhes técnicos, arquitetura e consumo de recursos não estão disponíveis publicamente.

E. Comparação e lacuna atendida

A Tabela I, resume as principais características técnicas dos trabalhos correlatos analisados e evidencia a contribuição do sistema proposto neste artigo.

Tabela I
Comparativo entre trabalhos correlatos e a proposta.

Trabalho	CNN embarcada	Foco em sirene	TinyML	AIoT
Atmel Siren [7]	–	✓	–	✓
ESC-CNN [8]	✓	–	✓	×
ACDNet [9]	✓	–	✓	–
Imagimob [10]	✓	✓	✓	?
Este trabalho	✓	✓	✓	✓

Este artigo propõe um sistema completo com:

- Execução embarcada de uma CNN otimizada com TF Lite em microcontrolador ARM Cortex-M4;
- Detecção específica de sirenes com validação em áudio urbano real;
- Ações imediatas integradas ao hardware embarcado, caracterizando uma aplicação prática de AIoT.

Esses elementos reforçam a contribuição do trabalho nos eixos de aprendizado embarcado e inteligência distribuída, assim como também mostra que poucos trabalhos conciliam detecção embarcada com resposta física imediata integrada ao hardware. A solução proposta se diferencia por abordar de forma simultânea: (i) a restrição computacional dos dispositivos, (ii) o pré-processamento eficiente de áudio, (iii) o uso de CNNs compactas e otimizadas, e (iv) a ativação imediata de dispositivos sinalizadores. Com isso, contribui diretamente para o avanço de aplicações Edge-AI distribuídas, alinhadas com o paradigma da cidade inteligente.

IV. Método Proposto

Este trabalho propõe a construção de um sistema AIoT (Artificial Intelligence of Things) voltado à detecção de sirenes de veículos de emergência em ambientes urbanos, utilizando aprendizado profundo embarcado. A metodologia foi estruturada para atender os requisitos de desempenho em tempo real, baixo consumo de energia e execução local sem dependência de conectividade externa. O fluxo de desenvolvimento abrange desde a aquisição e preparação dos dados até a inferência embarcada em microcontrolador.

A. Aquisição e Preparação do Conjunto de Dados

O conjunto de dados utilizado foi obtido da plataforma Kaggle [11], contendo gravações de sirenes de ambulância, caminhão de bombeiros e sons urbanos típicos de tráfego,

totalizando 600 arquivos em formato `.wav`. Cada arquivo possui 3 segundos de duração, taxa de amostragem de 44,1 kHz e profundidade de 32 bits.

Para viabilizar a aplicação embarcada, os dados foram pré-processados visando a padronização da duração (9 segundos por arquivo) e a adequação às limitações do hardware. Os arquivos foram regravados com taxa de amostragem de 16 kHz e profundidade de 16 bits. Essa escolha reduz significativamente o volume de dados, diminuindo o custo computacional e mantendo uma qualidade sonora satisfatória para a extração de características relevantes.

O conjunto de dados foi então dividido em 80% para treinamento e 20% para teste, respeitando o balanceamento entre as duas classes: **sirenes** e **sons de tráfego**.

B. Segmentação de Áudio

Depois de padronizar os arquivos de áudio, o próximo passo foi dividi-los em trechos menores, processo conhecido como segmentação. Essa técnica permite que o sistema identifique sons de sirene mesmo que eles apareçam apenas em partes curtas do áudio.

Neste trabalho, cada gravação de 9 segundos foi dividida em janelas de 2 segundos. Essas janelas foram criadas com uma sobreposição de 0,5 segundo entre elas. Ou seja, uma nova janela começa a cada meio segundo, aproveitando parte do trecho anterior. Com isso, diferentes partes da mesma gravação são analisadas em momentos distintos.

É importante destacar que essa técnica **não aumenta a quantidade de dados originais**, mas permite observar diferentes partes do mesmo som, em diferentes momentos. Isso ajuda o modelo a reconhecer sirenes mesmo quando elas ocorrem em posições variadas no tempo.

A escolha da duração de 2 segundos para cada janela foi feita com base em dois critérios principais: (i) garantir que o som da sirene esteja completo dentro do trecho analisado, e (ii) manter o tempo de resposta do sistema curto, algo essencial para aplicações em tempo real.

A sobreposição entre janelas funciona como uma forma de suavizar a transição entre os trechos analisados, melhorando a sensibilidade do sistema a sons rápidos ou de curta duração. Essa técnica também permite que o modelo veja o mesmo som sob ângulos diferentes, o que o torna mais preparado para lidar com situações reais.

Em resumo, a segmentação por janelas sobrepostas contribui para que o sistema seja mais eficiente e confiável na detecção de sirenes, mesmo em ambientes ruidosos. Essa abordagem é especialmente útil para sistemas embarcados [4], [9] com inteligência artificial, onde o tempo de resposta e o uso eficiente dos dados são fatores críticos.

C. Geração de Espectrogramas

Após a segmentação, os trechos de áudio foram convertidos em uma forma visual chamada *espectrograma*. Essa transformação é essencial para que o modelo de aprendizado profundo possa identificar padrões nos sons de sirenes. O espectrograma representa como as frequências do som variam ao longo do tempo e, por isso, é bastante utilizado em tarefas de reconhecimento de sons.

Diferentemente da forma de onda bruta, o espectrograma destaca informações que ajudam o modelo a diferenciar, por exemplo, uma sirene de uma buzina ou de ruído de tráfego. Essa representação é particularmente útil para redes neurais convolucionais (CNN), que são especialistas em identificar padrões visuais.

Para gerar os espectrogramas, foram utilizados os seguintes parâmetros:

- **Frame Length:** 75 milissegundos – determina a duração de cada janela de análise. Com uma taxa de amostragem de 16 kHz, isso equivale a aproximadamente 1200 amostras por janela.
- **Frame Stride:** 37,5 milissegundos – define o intervalo entre o início de uma janela e a seguinte, resultando em sobreposição entre elas. Essa sobreposição ajuda a capturar variações rápidas no sinal de áudio.
- **FFT Length:** 128 pontos – define a resolução das frequências analisadas. Este valor foi escolhido por ser suficiente para distinguir os padrões das sirenes e ainda ser leve para execução no microcontrolador.
- **Noise Floor:** 52 dB – atua como limiar mínimo para descartar componentes espectrais de baixa intensidade, reduzindo a interferência de ruídos de fundo irrelevantes.

Considerando um sinal de 2 segundos de duração, esses parâmetros resultam em um total de 52 janelas temporais, cada uma com 65 componentes espectrais, totalizando 3.380 features por espectrograma.

A escolha desses valores foi feita com base em um equilíbrio entre qualidade da representação e viabilidade de execução em tempo real em um sistema embarcado. Por exemplo, um valor maior de FFT traria melhor resolução de frequência, mas exigiria mais memória e processamento.

Em resumo, com esse conjunto de parâmetros, o espectrograma gerado mantém as características essenciais dos sons de sirene, como variações de tom e ritmo, e ainda é leve o suficiente para ser processado localmente em um microcontrolador com memória e capacidade limitadas. Essa etapa é fundamental para viabilizar a aplicação de IA embarcada (AIoT) em tempo real.

D. Arquitetura e Treinamento da Rede Neural

Para realizar a detecção dos sons de sirene, foi utilizado um modelo de aprendizado profundo baseado em redes neurais convolucionais (CNN). Esse tipo de rede é especialmente eficaz para identificar padrões visuais, como aqueles encontrados nos espectrogramas gerados a partir dos áudios segmentados.

A arquitetura adotada foi projetada para ser simples o suficiente para execução embarcada, mas ainda assim robusta para alcançar alta precisão. A rede é composta pelas seguintes camadas:

- **Três camadas convolucionais (Conv2D)** com 8, 16 e 32 filtros, respectivamente, cada uma com kernel de tamanho 3×3. Essas camadas são responsáveis por extrair características visuais dos espectrogramas, como bordas,

padrões de frequência e variações temporais associadas aos sons de sirene.
- **Camadas de pooling** intercaladas, utilizadas para reduzir a dimensão das representações intermediárias. Isso diminui o custo computacional e ajuda a evitar o sobreajuste (overfitting).
- **Camada de dropout** com taxa de 0,5, aplicada logo antes da camada final. Essa técnica desativa aleatoriamente metade dos neurônios durante o treinamento, forçando o modelo a generalizar melhor e evitar dependência excessiva de padrões específicos.
- **Camada flatten**, que transforma a saída da última camada convolucional em um vetor unidimensional para entrada na próxima etapa.
- **Camada densa (fully connected)** com 2 neurônios na saída, cada um representando uma das duas classes do problema: *"sirene"* e *"ruído comum"*.

Antes de alimentar o modelo, os espectrogramas foram transformados em vetores com 3380 atributos, sendo posteriormente reorganizados em 65 colunas para se ajustar ao formato esperado pelas camadas convolucionais.

O treinamento do modelo foi realizado em ambiente desktop utilizando a biblioteca TensorFlow. Os seguintes parâmetros foram adotados:

- **Taxa de aprendizado:** 0,001 – valor escolhido para permitir uma atualização gradual dos pesos da rede.
- **Número de épocas:** 100 – definido empiricamente para garantir a convergência do modelo sem sinais de sobreajuste.
- **Otimizador:** Foi escolhido o algoritmo Adam (*Adaptive Moment Estimation*) que é amplamente utilizado em tarefas de classificação por sua eficiência e estabilidade na atualização de pesos.

Essa etapa de treinamento foi realizada em um ambiente com capacidade computacional superior (não embarcado), e posteriormente o modelo treinado foi convertido para um formato leve (TF Lite), compatível com o microcontrolador utilizado na aplicação final.

A escolha dessa arquitetura representa um compromisso entre desempenho e simplicidade computacional, atendendo às restrições de memória e processamento típicas de dispositivos embarcados. Os resultados obtidos, apresentados na Seção VI, confirmam a viabilidade de executar esse tipo de rede neural em tempo real em um sistema AIoT com inteligência local.

E. Conversão e Implantação em Microcontrolador (ML Embarcado)

Conforme descrito anteriormente (treinamento e conversão), o modelo de rede neural foi convertido para o formato *TF Lite*, uma versão otimizada para execução em dispositivos com recursos computacionais limitados. Essa etapa é essencial para viabilizar a execução local do modelo, sem dependência de servidores externos ou conexão com a nuvem.

O modelo foi implantado no microcontrolador **Arduino Nano 33 BLE Sense Rev2**, cujas principais características são:

- **Processador:** ARM Cortex-M4, 32 bits, rodando a 64 MHz;
- **Memória:** 256 KB de SRAM e 1 MB de memória Flash;
- **Sensor de áudio:** Microfone digital integrado, baseado em modulação PDM (Pulse Density Modulation).

A escolha deste hardware se deve a três fatores principais: (i) integração direta com sensor de áudio digital; (ii) suporte à biblioteca TF Lite; e (iii) consumo energético reduzido, características importantes em aplicações portáteis e urbanas, onde o sistema precisa funcionar de forma contínua e eficiente.

O pipeline de inferência embarcada foi implementado em linguagem C++ e segue as seguintes etapas:

1) **Captura de áudio em tempo real** pelo microfone MEMS com modulação PDM.
2) **Pré-processamento do sinal** localmente: cálculo do espectrograma com base nas janelas de áudio em tempo real.
3) **Inferência do modelo** de aprendizado profundo embarcado utilizando os dados do espectrograma como entrada.
4) **Resposta do sistema:** caso uma sirene seja detectada, um LED é acionado como sinal de alerta. O LED foi adotado como uma prova de conceito, mas é claro que outras formas de alertas poderiam ser utilizadas.

Para que o modelo pudesse ser executado no dispositivo, foi necessário realizar quantização, reduzindo a representação dos pesos e ativações de 32 bits para 8 bits. Isso reduziu o uso de memória e tornou possível a execução dentro dos limites de RAM e Flash do dispositivo, sem perdas significativas de precisão.

Essa arquitetura oferece as principais vantagens da computação na borda (edge computing): **baixa latência, maior privacidade e independência de conectividade**. Como todo o processamento, da captura do som à inferência da rede neural, ocorre dentro do próprio dispositivo, o sistema é totalmente autônomo e apto a operar em tempo quase real, mesmo em locais sem nenhuma conexão à internet.

Esse tipo de solução ilustra de forma clara um sistema **AIoT (Artificial Intelligence of Things)**, no qual sensores, inteligência artificial e conectividade são combinados de forma eficiente para resolver um problema do mundo real com resposta rápida e inteligente.

F. Resposta Física Imediata

Entende-se "resposta imediata"aqui como resposta com latência sub-segundo; nesta implementação, a latência média foi de 139 ms. O sistema proposto integra uma resposta embarcada por meio do acionamento de um LED conectado ao microcontrolador, que é ativado imediatamente após a detecção de uma sirene. Ao ultrapassar um limiar de probabilidade pré-definido, o modelo embarcado dispara uma saída digital, acionando o LED com padrão intermitente. Essa resposta ocorre com latência inferior a 1 segundo, sem necessidade de conectividade com a nuvem.

Essa atuação local amplia a utilidade do sistema em ambientes urbanos ruidosos ou com visibilidade reduzida, fun-

cionando como reforço visual em situações de emergência. A abordagem é especialmente relevante para integração com sinalizadores urbanos, veículos de emergência ou dispositivos de acessibilidade, alinhando-se ao paradigma de cidades inteligentes baseadas em AIoT.

V. Experimentos

Este trabalho caracteriza-se como um estudo de caso experimental em ambiente simulado. A dificuldade de capturar sons reais de sirenes em vias urbanas, devido à variabilidade das condições acústicas e à dificuldade de reprodutibilidade, motivou a adoção de uma base pública de sons rotulados (Kaggle e YouTube), simulando diferentes cenários urbanos de forma controlada e técnica.

A. Objetivo da Avaliação

O objetivo principal é demonstrar que modelos de aprendizado profundo podem ser executados de forma embarcada, com autonomia, em microcontroladores com restrições de memória e processamento, para a detecção de sons de sirenes em ambientes urbanos. A proposta insere-se no contexto de sistemas **AIoT** (Artificial Intelligence of Things), onde sensores e inteligência artificial são combinados em dispositivos embarcados com capacidade de resposta autônoma.

B. Configuração Experimental

O modelo de rede neural convolucional, treinado previamente em ambiente desktop, foi convertido para o formato *TF Lite* , com quantização para representação em 8 bits. Em seguida, foi implantado no microcontrolador **Arduino Nano 33 BLE Sense Rev2**, que possui um processador ARM Cortex-M4 de 64 MHz, 256 KB de RAM e 1 MB de memória Flash. Esse dispositivo também conta com um microfone digital MEMS com modulação PDM, adequado para captura de áudio embarcada.

A captura do áudio, a geração do espectrograma e a inferência do modelo são realizadas localmente no microcontrolador, em tempo real. Cada janela de 2 segundos de áudio é processada imediatamente após a sua aquisição. O sistema foi programado em C++ utilizando a biblioteca TF Lite.

C. Procedimento Experimental

Os testes foram realizados em ambiente controlado, com reprodução de sons representativos das duas classes do problema: *(i)* sirenes de ambulância e polícia e *(ii)* ruídos urbanos diversos, como buzinas, motores e conversas. Os sons foram reproduzidos por alto-falantes posicionados próximos ao microcontrolador.

Durante a execução, o microcontrolador captava continuamente o som ambiente, processava os dados, gerava o espectrograma correspondente, e utilizava o modelo embarcado para classificar o trecho de áudio. A resposta do sistema (detecção ou não de sirene) era registrada e validada com base no som reproduzido naquele instante.

As métricas de desempenho coletadas incluíram tempo médio de inferência, uso de memória RAM e Flash, além das métricas de classificação: acurácia, precisão, revocação e F1-Score, que foram medidas por meio da contagem de acertos e erros comparando as previsões do modelo com as classes verdadeiras dos áudios testados.

VI. Discussão dos Resultados

Os resultados obtidos na execução embarcada do modelo indicam que o sistema proposto é tecnicamente viável e eficaz para a detecção de sirenes. Neste acurácia de 96,83%, associada a uma precisão de 94,49% e sensibilidade de 99,04%, revela um desempenho estável e confiável mesmo após o processo de quantização do modelo, que o adaptou para execução em dispositivos com recursos computacionais restritos.

A métrica F1-Score, que harmoniza precisão e revocação, também permaneceu elevada (96,70%), evidenciando que o modelo mantém um bom equilíbrio entre a capacidade de detectar verdadeiros positivos e a limitação de falsos alarmes. Essa característica é particularmente importante em contextos urbanos, onde o sistema pode ser exposto a ruídos diversos e imprevisíveis.

O tempo médio de inferência de 139 milissegundos representa o intervalo entre a captura do áudio, o processamento do espectrograma e a classificação pelo modelo, medida por carimbos de tempo no firmware (da captura ao término da inferência), acionando uma saída digital ao detectar a sirene. Essa latência está abaixo do limite considerado aceitável para aplicações IAoT em ambientes urbanos, que toleram até 500 ms de resposta para manter eficácia e segurança, de acordo com estudos de latência em IoT (100–500 ms) [12], [13].

Em termos de recursos, o modelo embarcado utilizou aproximadamente 127,76 KB de memória RAM, o que corresponde a cerca de **49,9%** dos 256 KB disponíveis no microcontrolador. Para a memória Flash, o consumo foi de 699,88 KB, representando aproximadamente **68,5%** do total de 1 MB (1024 KB). Esses valores demonstram que o sistema opera com folga suficiente para execução estável, mantendo espaço residual para outras funcionalidades, sensores e controle do dispositivo.

A partir disso, observa-se que, com o uso de técnicas adequadas como quantização do modelo e compressão do espectrograma, é possível realizar inferência local eficiente mesmo com arquiteturas simples, como redes neurais convolucionais rasas, em plataformas embarcadas de baixo consumo energético.

Além do bom desempenho técnico, os resultados reforçam o caráter **AIoT** do sistema desenvolvido. O dispositivo é capaz de capturar o som ambiente, extrair características relevantes, processar um modelo de aprendizado profundo e tomar decisões localmente, sem qualquer dependência de conectividade com a nuvem. Esse tipo de abordagem favorece maior privacidade, redução de latência e maior autonomia operacional, características desejáveis em sistemas de monitoramento inteligente distribuído.

Do ponto de vista aplicado, o sistema pode ser integrado a semáforos inteligentes, postes urbanos, veículos autônomos

ou dispositivos vestíveis para alertas situacionais. O sucesso na execução de um pipeline completo de IA embarcada com desempenho competitivo representa um avanço concreto na direção da popularização de soluções de aprendizado de máquina no contexto da borda (edge computing).

VII. Conclusão e Trabalhos Futuros

Este trabalho apresentou o desenvolvimento e avaliação de um sistema embarcado capaz de realizar a detecção de sirenes em tempo real utilizando aprendizado profundo. O sistema foi implementado em um microcontrolador de baixo consumo energético, explorando técnicas de pré-processamento de áudio, geração de espectrogramas e inferência de modelos quantizados via *TF Lite*.

A proposta insere-se no contexto de **AIoT** (Artificial Intelligence of Things), evidenciando que é possível combinar sensores, conectividade e inteligência artificial embarcada para criar soluções autônomas, responsivas e eficientes. O modelo de rede neural convolucional treinado foi convertido com sucesso para execução local, tendo desempenho expressivo: acurácia de 96,83%, F1-score de 96,70% e tempo médio de inferência de 139 ms, com utilização de menos de 50% da RAM e aproximadamente 69% da memória Flash disponível no dispositivo.

A principal contribuição deste trabalho é demonstrar, de forma prática, que técnicas modernas de aprendizado de máquina podem ser aplicadas de forma embarcada, sem dependência de nuvem, mesmo em plataformas com restrições severas de hardware. Essa abordagem favorece aplicações em cenários com baixa conectividade, necessidade de privacidade ou demanda por respostas imediatas, como semáforos inteligentes, veículos autônomos e dispositivos vestíveis de alerta.

Embora os resultados sejam promissores, esta pesquisa possui algumas limitações. O ambiente de teste foi simulado e controlado, o que pode não refletir todas as complexidades acústicas de um ambiente urbano real. Além disso, a base de dados utilizada não contemplou variações geográficas ou culturais de sinais sonoros, o que pode impactar a generalização do modelo em larga escala.

Como trabalhos futuros, pretende-se:

- Realizar experimentos em ambientes reais com sons urbanos captados diretamente em vias públicas, explorando diferentes níveis de ruído de fundo e distâncias variáveis para ampliar a qualidade do dataset.
- Expandir o conjunto de classes sonoras, incluindo outros sons de emergência (como sirenes de bombeiros e polícia), buzinas, alarmes e apitos industriais.
- Explorar modelos mais leves e eficientes, como redes otimizadas com técnicas de *pruning* e *knowledge distillation*, a fim de reduzir ainda mais o consumo de recursos e permitir escalabilidade.
- Investigar a integração do sistema de detecção de sirenes com plataformas de automação urbana e veículos inteligentes, permitindo ações automáticas como priorização de semáforos, notificações em tempo real para motoristas e integração com sistemas de resposta emergencial.
- Avaliar a integração com mecanismos de resposta automática, como sinalização luminosa, envio de notificações ou comunicação com outros dispositivos IoT.
- Investigar a viabilidade de atualização dinâmica do modelo embarcado (aprendizado contínuo na borda) e a segurança na transmissão de novos parâmetros para dispositivos em campo.

Com essas direções, espera-se contribuir para o avanço das soluções de inteligência artificial embarcada e consolidar o papel do *AIoT* na construção de cidades mais inteligentes, seguras e responsivas.

Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo suporte concedido por meio do Código de Financiamento 001. Agradecem também ao Instituto de Pesquisas Eldorado, cujo suporte técnico, infraestrutura e incentivo à inovação tecnológica foram decisivos para o desenvolvimento desta pesquisa. Estende-se o agradecimento à Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM), através do projeto POSGRAD, que viabilizou a execução desta iniciativa. O apoio dessas instituições foi essencial para a concretização deste trabalho.

Referências

[1] R. Sellami and M. Hammami, "Urban sound recognition in smart cities using an iot–fog based deep learning architecture," *Applied Sciences*, vol. 15, no. 3, p. 1201, 2023.

[2] M. J. Baucas and P. Spachos, "Using cloud and fog computing for large scale iot-based urban sound classification," *arXiv preprint arXiv:1910.07652*, 2019.

[3] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, S. Regev *et al.*, "Tensorflow lite micro: Embedded machine learning on tinyml systems," *arXiv preprint arXiv:2010.08678*, 2020.

[4] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers*. O'Reilly Media, 2019.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[7] F. Meucci, L. Pierucci, E. Del Re, and P. Desii, "A real-time siren detector to improve safety of guide in traffic environment," *Transactions on Emerging Telecommunications Technologies*, vol. 19, no. 6, pp. 676–684, 2008.

[8] J. Nordby, "Environmental sound classification on microcontrollers using convolutional neural networks," Master's thesis, Norwegian University of Life Sciences, 2019. [Online]. Available: http://hdl.handle.net/11250/2611624

[9] M. Mohaimenuzzaman, C. Bergmeir, I. West, and B. Meyer, "Environmental sound classification on the edge: a pipeline for deep acoustic networks on extremely resource-constrained devices," *Pattern Recognition*, vol. 133, p. 109025, 2023, also available as ArXiv:2103.03483.

[10] I. AB. (2023) Siren detection ready model. [Online]. Available: https://www.imagimob.com/siren-ready-model

[11] V. Unnikrishnan, "Emergency vehicle siren sounds," 2020, acesso em: 6 maio 2024. [Online]. Available: https://www.kaggle.com/datasets/vishnu0399/emergency-vehicle-siren-sounds

[12] Nabto, "Iot latency: The power of real-time communication," 2025, relatório online.

[13] P. Muralidhara, "Edge and cloud integration: Optimizing latency and resource allocation for iot applications," *IJECS*, vol. 5, no. 7, pp. 17 388–17 406, 2016.