Avaliação de Desempenho e Consumo Energético de TinyML em Dispositivos de Borda para Previsão de Precipitação

Clariele Almeida[1], **Rafael José Moura**[1], **Danilo Araújo**[1], **Ermeson Andrade**[1]

[1]Universidade Federal Rural de Pernambuco (UFRPE)
Recife, PE – Brasil

{clariele.almeida, rafael.moura, danilo.araujo, ermeson.andrade}@ufrpe.br

Abstract. *In the Internet of Things (IoT) ecosystem, edge devices operate with limited computational and energy resources, requiring efficient machine learning models. In this work, a rainfall prediction system based on Tiny Machine Learning (TinyML) models is developed and executed on a resource constrained edge device. Convolutional Neural Network (CNN) and Multilayer Perceptron (MLP) models are implemented on the Arduino Nano 33 BLE Sense. Based on this system, the energy efficiency during inference is evaluated, as well as the impact of model optimization techniques including pruning, quantization, and knowledge distillation. The results enable the comparison of energy consumption and model performance, supporting the selection of more efficient TinyML solutions for IoT applications.*

Resumo. *No ecossistema da Internet das Coisas (IoT), dispositivos de borda operam com recursos computacionais e energéticos limitados, exigindo modelos de aprendizado de máquina eficientes. Neste trabalho, é desenvolvido um sistema de predição de chuva baseado em modelos Tiny Machine Learning (TinyML) executados em um dispositivo de borda com recursos limitados. São implementados modelos Convolutional Neural Network (CNN) e Multilayer Perceptron (MLP) no Arduino Nano 33 BLE Sense. A partir desse sistema, avalia-se a eficiência energética durante a inferência e o impacto de técnicas de otimização de modelos, incluindo poda, quantização e knowledge distillation. Os resultados permitem comparar consumo energético e desempenho, contribuindo para a escolha de soluções mais eficientes para aplicações IoT baseadas em TinyML.*

1. Introdução

As mudanças climáticas representam uma ameaça global, cujos impactos tendem a ser mais severos em países em desenvolvimento, que geralmente apresentam maior vulnerabilidade a desastres ambientais [Kumar et al. 2024]. Nesse contexto, eventos climáticos extremos, como chuvas intensas e enchentes, podem causar impactos significativos em áreas urbanas e rurais, afetando a infraestrutura, a segurança alimentar e a gestão de recursos hídricos.

O crescimento da IoT tem impulsionado a implantação de sensores e dispositivos inteligentes capazes de coletar dados ambientais em diferentes contextos, como monitoramento climático, agricultura de precisão e sistemas de alerta meteorológico. Nesse

cenário, a IoT conecta sensores, atuadores e dispositivos embarcados que permitem monitoramento e controle de ambientes físicos por meio da coleta e análise de dados em tempo real [Choudhary 2024]. Muitos desses dispositivos operam em cenários distribuídos e com recursos limitados de processamento, memória e energia, frequentemente alimentados por bateria. Exemplos incluem estações meteorológicas remotas, sensores instalados em áreas agrícolas ou regiões suscetíveis a enchentes e dispositivos embarcados em sistemas de monitoramento ambiental, nos quais a conectividade com servidores centrais pode ser intermitente ou apresentar latência elevada, tornando necessário o processamento local dos dados. Dessa forma, a eficiência computacional torna-se requisito central para aplicações inteligentes executadas na borda [Kallimani et al. 2024].

O avanço das técnicas de *Machine Learning* (ML) tem ampliado as possibilidades de análise e previsão em sistemas ambientais baseados em dados [Kumar et al. 2024]. Modelos de ML têm sido amplamente utilizados em tarefas de previsão meteorológica, incluindo estimativas de precipitação a partir de variáveis atmosféricas históricas e observações coletadas por sensores e estações meteorológicas [Scheifer et al. 2025, Vieira et al. 2025, Breder et al. 2025]. Esses modelos são capazes de capturar padrões complexos e relações não lineares presentes nos dados climáticos, frequentemente superando abordagens estatísticas tradicionais em termos de desempenho preditivo. Entretanto, muitos desses métodos foram originalmente projetados para ambientes computacionais com alta disponibilidade de recursos, o que dificulta sua aplicação direta em dispositivos de borda sujeitos a restrições de processamento, memória e consumo energético.

Dito isso, as abordagens baseadas em Inteligência Artificial de Borda (Edge AI) têm sido exploradas para permitir a execução de modelos de ML diretamente em dispositivos próximos à fonte de dados. Ao deslocar parte do processamento da nuvem para a borda da rede, essa abordagem pode reduzir latência, tráfego de dados e dependência de conectividade contínua [Bourechak et al. 2023]. Entretanto, a adoção de técnicas de ML em dispositivos embarcados impõe desafios relacionados ao consumo energético e à eficiência computacional. Esse cenário tem motivado o desenvolvimento de abordagens como TinyML, voltadas à execução de modelos de ML em plataformas com restrições severas de memória, processamento e energia [Abadade et al. 2023, Kallimani et al. 2024].

Apesar dos avanços recentes, grande parte dos estudos sobre previsão meteorológica com ML concentra-se principalmente na melhoria da acurácia preditiva [Scheifer et al. 2025, Vieira et al. 2025, Breder et al. 2025, Sham et al. 2025, Liyew and Melese 2021], enquanto aspectos relacionados ao custo computacional e ao consumo energético dos modelos ainda são pouco explorados, especialmente em cenários de computação de borda. Essa lacuna torna relevante investigar como diferentes modelos de ML se comportam quando executados em dispositivos com recursos limitados, considerando não apenas o desempenho preditivo, mas também sua eficiência energética.

Diante desse contexto, este trabalho tem como objetivo avaliar a eficiência energética de diferentes modelos de Machine Learning, MLP e CNN, aplicados à tarefa de previsão diária de precipitação. Os modelos são executados em um microcontrolador Arduino Nano 33 BLE Sense e avaliados considerando simultaneamente métricas de desempenho preditivo, latência de inferência e consumo energético. Essa análise conjunta permite investigar o *trade-off* entre acurácia, custo computacional e eficiência energética em ambientes de computação de borda. Os resultados obtidos fornecem subsídios para a

seleção de modelos de ML mais adequados para aplicações IoT com restrições de recursos, como sistemas de monitoramento ambiental e estações meteorológicas inteligentes.

O restante do artigo está organizado da seguinte forma: a Seção 2 discute os trabalhos relacionados; a Seção 3 descreve a metodologia adotada; a Seção 4 apresenta os resultados experimentais. Por fim, a Seção 5 traz as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

Diversos estudos têm investigado estratégias para viabilizar a execução de modelos de ML em dispositivos embarcados com restrições de memória, processamento e energia. Nesse contexto, técnicas de compressão e otimização de modelos têm sido amplamente exploradas para reduzir o custo computacional sem comprometer significativamente o desempenho preditivo. Por exemplo, [Bhushan et al. 2025] investigam a implementação de um sistema de detecção de objetos baseado em MobileNetV2 em um microcontrolador ARM Cortex-M4, analisando o impacto de técnicas como quantização, poda e *Knowledge Distillation* (KD) no uso de memória, latência de inferência e consumo energético. Os resultados indicam que a quantização pós-treinamento em 8 bits foi a única abordagem capaz de viabilizar a implantação do modelo dentro das restrições de memória do dispositivo. De forma semelhante, [Mnif et al. 2024] propõem uma abordagem combinada de compressão de modelos, envolvendo poda de filtros, quantização e KD para otimizar uma CNN aplicada ao reconhecimento de gestos em dispositivos IoT de borda, obtendo reduções expressivas no tamanho do modelo e no tempo de inferência. Entretanto, esses estudos concentram-se principalmente na otimização de uma única arquitetura de rede neural, sem realizar comparações entre diferentes algoritmos de ML.

Para a análise da eficiência energética de modelos de ML executados em dispositivos de borda, [Rahman et al. 2024] comparam diferentes microcontroladores, incluindo STM32F446RE, Arduino Nano 33 BLE Sense, Raspberry Pi RP2040 e ESP32, na implementação de um sistema vestível para classificação de arritmia em tempo real. O estudo avalia métricas como consumo de potência, tempo de inferência, uso de memória e custo, demonstrando que a eficiência energética pode variar significativamente entre plataformas de hardware. De forma complementar, [Tao et al. 2022] investigam o impacto combinado de poda e quantização na eficiência energética de uma rede ResNet-18 implantada em um microcontrolador STM32H7A3, analisando diferentes configurações de compressão e seu efeito no consumo de energia, tempo de inferência e tamanho do modelo. Embora esses estudos forneçam evidências importantes sobre o impacto das técnicas de otimização e do hardware no consumo energético, a maioria das análises considera apenas uma arquitetura de modelo, limitando a compreensão de como diferentes algoritmos influenciam a eficiência energética em cenários de Edge AI.

No contexto da previsão de precipitação, diversos estudos têm aplicado técnicas de ML para modelar variáveis meteorológicas e estimar a ocorrência ou quantidade de chuva. O estudo de Sham et al. (2025) apresenta uma revisão das principais técnicas de IA utilizadas para previsão de precipitação, discutindo o uso de algoritmos como *Support Vector Machine* (SVM), Redes Neurais Artificiais (ANNs) e *Random Forest* (RF) na modelagem de dados climáticos. Os autores destacam que esses métodos são capazes de capturar relações não lineares presentes em variáveis meteorológicas complexas, frequentemente superando abordagens estatísticas tradicionais em desempenho preditivo. Liyew

e Melese (2021) também investigam a aplicação de diferentes algoritmos de ML para prever a quantidade diária de precipitação a partir de variáveis atmosféricas históricas, incluindo temperatura, umidade, evaporação e velocidade do vento. O estudo compara modelos como *Multiple Linear Regression*, RF e XGBoost, demonstrando que técnicas baseadas em ML podem extrair padrões complexos dos dados meteorológicos e melhorar a precisão das previsões. Entretanto, essas abordagens são avaliadas em ambientes computacionais com maior disponibilidade de recursos, sem considerar restrições de processamento, memória ou consumo energético típicas de dispositivos de borda.

Apesar dos avanços, a literatura ainda carece de análises que comparem múltiplos algoritmos de ML sob condições idênticas de borda, considerando simultaneamente precisão preditiva, latência e eficiência energética. Muitos trabalhos focam em aprimorar a acurácia de modelos de previsão de chuva sem avaliar o custo computacional em ambientes embarcados. Nesse sentido, este trabalho preenche essa lacuna ao comparar diferentes modelos TinyML (MLP e CNN) aplicados à previsão diária de precipitação em um dispositivo de borda. Avaliamos acurácia, latência de inferência, uso de recursos e consumo de energia, bem como o impacto de técnicas de otimização (poda, quantização e KD). Assim, buscamos mostrar o *trade-off* entre desempenho preditivo e eficiência energética em aplicações ambientais de Edge AI, orientando a seleção de soluções TinyML adequadas para cenários com recursos limitados.

3. Metodologia

Esta seção descreve as etapas adotadas para a previsão de precipitação em dispositivos de borda, incluindo a definição do problema, o tratamento dos dados, os modelos utilizados, o processo de treinamento, a configuração experimental e as métricas de avaliação.

3.1. Definição do Problema

O problema investigado consiste na previsão de precipitação a partir de variáveis meteorológicas históricas coletadas por uma estação meteorológica. A tarefa foi formulada como um problema de regressão temporal *one-step-ahead*, utilizando janelas deslizantes com contexto fixo de 7 observações anteriores para prever a próxima medição de precipitação, caracterizando um horizonte de curto prazo. Assume-se que apenas dados históricos observados estão disponíveis durante a inferência.

3.2. Dataset e Preprocessamento de Dados

O *dataset* utilizado nesta pesquisa foi desenvolvido por pesquisadores da Universidade Federal Rural de Pernambuco (UFRPE) e utilizado no estudo apresentado em [de Albuquerque and de Araujo 2025]. A base contém registros meteorológicos provenientes de duas estações: Estação Pesca UFRPE e Estação Vargem Fria. Neste trabalho foi utilizada apenas a base da Estação Pesca UFRPE, composta por 20 028 registros coletados entre outubro de 2023 e junho de 2024, correspondendo a aproximadamente oito meses de medições com frequência média de cerca de dez medições diárias. As variáveis incluem temperatura, umidade relativa do ar, velocidade do vento, rajada de vento e precipitação diária.

Os dados foram organizados cronologicamente e submetidos a procedimentos de tratamento, incluindo verificação de consistência, conversão de tipos e interpolação tem-

poral para tratamento de valores ausentes. Em seguida, foi aplicada normalização utilizando o método *StandardScaler*, que padroniza as variáveis com base na média e no desvio padrão do conjunto de treinamento. Para modelar dependências temporais, os dados foram estruturados em janelas deslizantes contendo sete observações consecutivas. O *dataset* foi dividido temporalmente em conjuntos de treinamento e validação, reservando aproximadamente 10% das observações mais recentes para validação.

3.3. Modelos Edge AI

Foram avaliados dois modelos adequados para execução em dispositivos de borda: uma MLP e uma CNN de 1 Dimensão (CNN-1D). Esses modelos foram selecionados por apresentarem baixa complexidade computacional, número reduzido de parâmetros e compatibilidade com ferramentas de implantação em microcontroladores.

A MLP recebe como entrada as janelas temporais geradas no pré-processamento, com dimensão $(7, 5)$, correspondendo a sete observações consecutivas de cinco variáveis meteorológicas. Cada janela é convertida em um vetor unidimensional por meio de uma operação de *flatten*, resultando em um vetor de 35 características. O modelo consiste em uma única camada densa com 8 neurônios e ativação ReLU, seguida por uma camada de saída linear responsável pela estimativa da precipitação. A CNN-1D, por outro lado, processa diretamente as janelas temporais no formato sequencial $(7, 5)$, sem necessidade de *flatten*. A arquitetura é composta por duas camadas convolucionais unidimensionais com 16 filtros cada e tamanho de *kernel* igual a 5, utilizando *padding* do tipo *same* e ativação ReLU. Em seguida, é aplicada uma camada de *Global Average Pooling* para agregação das características ao longo da dimensão temporal, reduzindo a saída para um vetor de dimensão fixa. Por fim, uma camada densa com saída linear é utilizada para realizar a regressão da precipitação.

3.4. Treinamento do Modelo e Configuração

Após o treinamento dos modelos base, foram aplicadas três técnicas de otimização: quantização pós-treinamento, poda de pesos e KD. A quantização converte pesos e ativações de ponto flutuante para representações inteiras de 8 bits, reduzindo o uso de memória e o custo computacional. A poda remove gradualmente pesos da rede durante o treinamento, produzindo modelos mais esparsos. Já a KD transfere conhecimento de um modelo maior (*teacher*) para um modelo menor (*student*), permitindo que arquiteturas mais compactas reproduzam o comportamento de modelos mais complexos. Após essas etapas, os modelos foram convertidos para o formato TensorFlow Lite (TFLite) para execução em dispositivos embarcados. Para garantir a reprodutibilidade dos experimentos, o código utilizado neste trabalho está disponível publicamente no repositório[1].

3.5. Configuração Experimental e Medição de Energia

Os experimentos foram conduzidos em um ambiente de computação de borda utilizando o microcontrolador Arduino Nano 33 BLE Sense. Os modelos convertidos para TFLite foram executados no dispositivo utilizando a biblioteca *TensorFlow Lite for Microcontrollers*, permitindo a realização das inferências diretamente no hardware embarcado.

[1] `https://github.com/clariele/tinyml-precipitation-energy`

O consumo de energia foi medido utilizando o sensor de corrente e tensão INA219, responsável por registrar continuamente os valores de corrente e tensão ao longo da execução dos experimentos. A partir dessas medições, foi estimada a potência instantânea e, posteriormente, calculada a energia consumida. Ressalta-se que as medições correspondem ao consumo total do sistema embarcado, incluindo o microcontrolador, periféricos e bibliotecas de suporte, não sendo restritas ao modelo. Para melhor estimar o impacto da inferência, cada rodada foi composta por duas fases distintas: uma fase de repouso (*baseline*), sem execução do modelo, e uma fase de inferência, permitindo estimar o consumo adicional associado à execução do modelo por diferença entre essas condições.

Cada modelo foi executado repetidamente com o objetivo de garantir maior estabilidade estatística nas medições. O código de aquisição foi configurado para executar 10 rodadas independentes por experimento, sendo que, em cada rodada, o processo de inferência foi repetido 60.000 vezes consecutivas. Entre rodadas consecutivas foi inserido um intervalo de 60 segundos, permitindo a estabilização da placa e do sensor INA219 antes do início da próxima medição, reduzindo possíveis efeitos térmicos e de sobrecarga do sistema que poderiam influenciar o consumo energético e o desempenho computacional.

3.6. Métricas de Avaliação

O desempenho dos modelos foi avaliado considerando três aspectos principais: desempenho preditivo, custo computacional (latência e utilização de recursos) e eficiência energética. Para avaliar a qualidade das previsões foram utilizadas métricas de regressão amplamente empregadas em problemas de previsão contínua: RMSE e MAE.

O custo computacional foi analisado por meio do tempo de inferência necessário para gerar uma previsão no dispositivo embarcado. Por fim, a eficiência energética foi avaliada pela energia consumida durante o processo de inferência, calculada a partir das medições de corrente e tensão obtidas com o sensor INA219. Essas métricas permitem comparar o impacto de diferentes arquiteturas e técnicas de otimização no desempenho e no consumo energético de aplicações TinyML em dispositivos de borda.

4. Resultados

Esta seção apresenta os resultados obtidos da execução dos modelos TinyML no dispositivo de borda. A análise considera três aspectos principais: desempenho preditivo, custo computacional e consumo energético. Esses resultados permitem avaliar o impacto de técnicas de otimização e analisar o *trade-off* entre precisão e eficiência energética.

4.1. Desempenho Preditivo

A Figura 1 apresenta os resultados para os modelos TinyMLP. O modelo *baseline* alcançou RMSE de 0,701 e MAE de 0,170. A quantização pós-treinamento preservou o desempenho preditivo, mantendo os mesmos valores de RMSE e MAE do modelo original, indicando que a redução da precisão numérica dos pesos não impactou significativamente as previsões. O modelo podado apresentou pequena melhora no desempenho, mantendo RMSE de aproximadamente 0,701 e reduzindo o MAE para 0,128. Esse comportamento sugere que o processo de poda pode ter atuado como mecanismo de regularização, removendo pesos redundantes e favorecendo a capacidade de generalização. Já o modelo *KD Student* apresentou leve degradação no desempenho, com RMSE de 0,704 e MAE de 0,182, o que é esperado devido à redução da capacidade representacional do modelo.

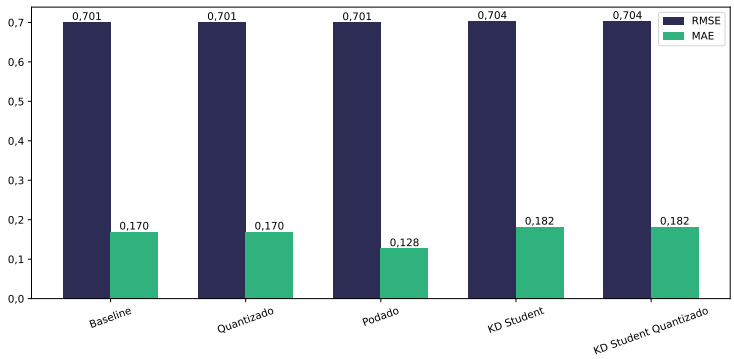

Figura 1. Desempenho preditivo das variantes do TinyMLP

A Figura 2 apresenta o desempenho preditivo obtido para a arquitetura TinyCNN. O modelo *baseline* apresentou RMSE de 0,740 e MAE de 0,222. Assim como observado na arquitetura MLP, a versão quantizada manteve praticamente o mesmo desempenho preditivo do modelo original. Por outro lado, o modelo podado apresentou uma leve degradação nas métricas, com RMSE aumentando para 0,763 e MAE para 0,282, sugerindo que a arquitetura convolucional pode ser mais sensível à remoção de parâmetros. De forma semelhante, o modelo *KD Student* apresentou erros mais elevados, com RMSE de 0,801 e MAE de 0,342, refletindo a redução no número de parâmetros e, consequentemente, na capacidade de representação do modelo.

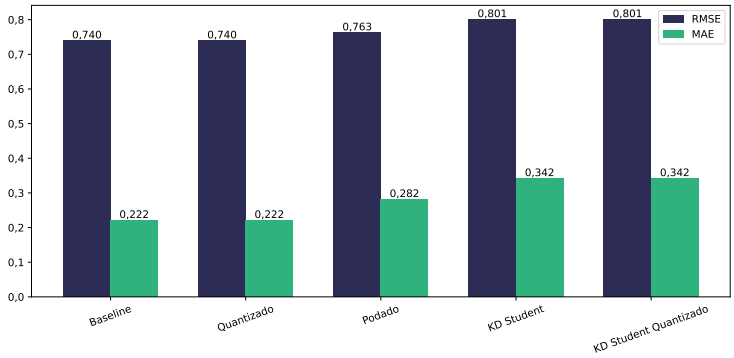

Figura 2. Desempenho Preditivo das Variantes do TinyCNN

Esses resultados indicam que a quantização preservou o desempenho preditivo em ambas as arquiteturas, enquanto poda e KD introduziram diferentes *trade-offs* entre complexidade e precisão, evidenciado os desafios da aplicação de técnicas de compressão em cenários TinyML, nos quais memória e capacidade computacional são restrições críticas para implantação embarcada. Entre os modelos avaliados, o TinyMLP podado apresentou o melhor desempenho, com RMSE de 0,701 e MAE de 0,128, indicando boa capacidade para previsões de precipitação de curto prazo no contexto investigado.

4.2. Latência e Utilização de Recursos

A Tabela 1 apresenta a latência média de inferência das arquiteturas TinyMLP e TinyCNN sob diferentes estratégias de otimização, juntamente com o desvio padrão das execuções repetidas, evidenciando a consistência das medições. Por utilizar operações mais comple-

xas para um problema simples de regressão, o TinyCNN apresentou maior custo computacional. Em contraste, o TinyMLP executou a mesma tarefa com tempo 93,8% menor, mostrando-se mais eficiente nesse cenário. Entre as variantes otimizadas, a quantização foi a técnica que mais beneficiou o TinyMLP, reduzindo a latência em 13,26 s. Para o TinyCNN, essas otimizações tornam-se ainda mais relevantes devido ao elevado tempo de inferência do modelo *baseline*. A combinação de KD com quantização apresentou o melhor resultado para o TinyCNN, reduzindo a latência em 75,5%, representando um ganho significativo para aplicações TinyML em dispositivos com recursos limitados.

Tabela 1. Comparação da latência de inferência (s) entre TinyMLP e TinyCNN.

Otimização	TinyMLP (s)	TinyCNN (s)
Quantizado	**8,53 ± 0,005722**	131,52 ± 0,008306
KD Student Quantizado	20,44 ± 0,015182	**86,01 ± 0,042047**
KD Student	21,72 ± 0,040976	145,44 ± 0,038260
Podado	21,76 ± 0,207232	348,49 ± 0,828587
Baseline	21,79 ± 0,016156	351,26 ± 0,857785

O uso de recursos de memória também foi analisado por meio do tamanho final dos modelos convertidos para TFLite. A Figura 3 apresenta a comparação entre as diferentes variantes dos modelos para as arquiteturas (a) TinyMLP e (b) TinyCNN. Para a arquitetura TinyCNN, o modelo *baseline* apresentou o maior tamanho, com 10.508 bytes, enquanto o modelo *KD Student Quantized* apresentou o menor tamanho, com 5.248 bytes. Esse comportamento está alinhado com o princípio de compressão em TinyML, no qual o objetivo não é apenas reduzir o número de parâmetros, mas também diminuir o custo de representação dos pesos em memória. A KD favorece a aprendizagem de representações mais eficientes no modelo estudante, enquanto a quantização reduz a precisão numérica dos parâmetros. Em conjunto, essas técnicas permitem reduzir significativamente o tamanho do modelo sem modificar sua arquitetura fundamental, o que facilita a implantação em microcontroladores com recursos de memória extremamente limitados.

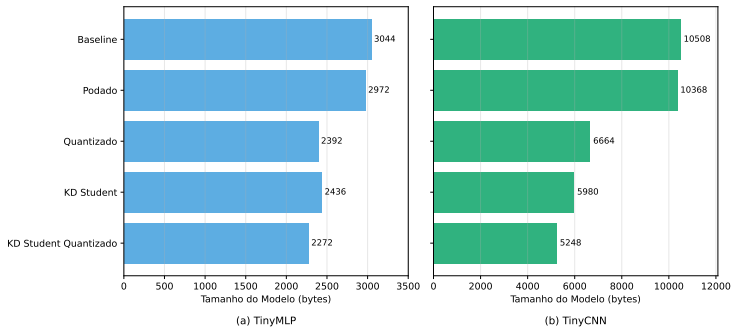

Figura 3. Comparação do tamanho dos modelos das variantes do TinyCNN.

Resultados semelhantes foram observados para a arquitetura TinyMLP, conforme mostrado na Figura 3. O modelo *baseline* apresentou tamanho de 2.972 bytes, enquanto o modelo *KD Student Quantized* apresentou o menor tamanho, com 2.272 bytes. De maneira geral, os resultados indicam que as técnicas de compressão aplicadas reduziram significativamente tanto a latência de inferência quanto o uso de memória, especialmente quando combinadas com quantização. Esses resultados são relevantes para aplicações de TinyML, nas quais restrições de memória e capacidade computacional representam fatores críticos para implantação de modelos em dispositivos embarcados.

4.3. Análise do Consumo de Energia

A Figura 4 apresenta o consumo energético dos modelos *baseline* e suas variantes otimizadas. Para o modelo TinyMLP, a quantização reduziu o consumo energético em 5,6 mJ, enquanto para o TinyCNN, a mesma técnica resultou em redução de 54,3 mJ, correspondendo a uma economia de energia de 90% e de 60%, respectivamente. Esses resultados evidenciam o impacto significativo da redução de precisão numérica dos pesos e ativações na eficiência energética durante a inferência. A maior redução observada para a arquitetura TinyCNN ocorreu com o modelo *KD Student Quantized*, que apresentou diminuição de aproximadamente 69,1 mJ no consumo energético em relação ao *baseline*, correspondendo a uma redução próxima de 77%. As demais técnicas apresentaram reduções mais modestas. No caso do TinyMLP, a poda resultou em redução de 0,5 mJ, enquanto o modelo *KD Student* apresentou redução de cerca de 0,6 mJ. A variação (*KD Student Quantized*) apresentou redução de 0,9 mJ. Para o TinyCNN, a poda teve impacto mínimo no consumo energético (redução inferior a 1%), enquanto o modelo *KD Student* apresentou redução semelhante à obtida com quantização, em torno de 54,6 mJ.

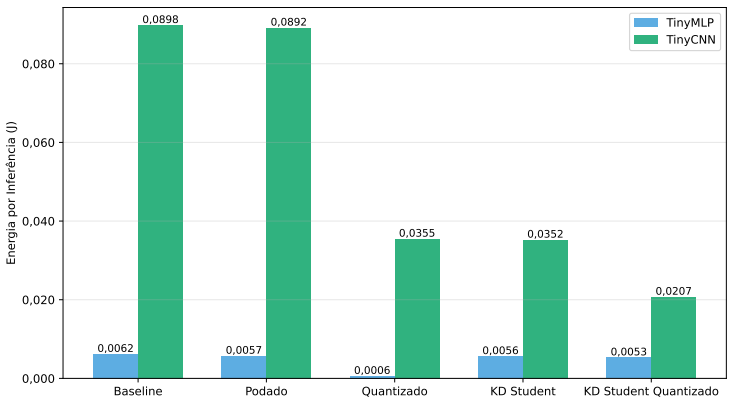

Figura 4. Consumo de energia por inferência dos modelos TinyMLP e TinyCNN sob diferentes estratégias de otimização.

De maneira geral, os resultados indicam que técnicas baseadas apenas em compressão estrutural, como poda ou destilação, tendem a produzir ganhos energéticos limitados quando aplicadas isoladamente em microcontroladores. Em particular, a poda apresentou economias de energia praticamente negligenciáveis nos experimentos realizados. Esse comportamento pode ser explicado pelo fato de que, em muitas plataformas de hardware, pesos zerados gerados pela poda ainda exigem a execução das mesmas operações aritméticas durante a inferência, a menos que o hardware ou a biblioteca de inferência possua suporte específico para computação esparsa. Em contraste, técnicas que reduzem a precisão numérica dos parâmetros, como a quantização, apresentam impacto mais significativo no consumo energético durante a inferência. Esse comportamento é consistente com o fato de que a redução da precisão numérica diminui o custo computacional das operações aritméticas e o volume de dados movimentados em memória, fatores que influenciam diretamente o consumo energético em dispositivos embarcados.

4.4. Trade-off entre Energia e Desempenho

A Figura 5 apresenta gráficos de dispersão relacionando o erro preditivo (MAE) e a energia consumida por inferência para as arquiteturas TinyMLP e TinyCNN. As linhas trace-

jadas indicam o desempenho do modelo *baseline*, dividindo o espaço em quadrantes que permitem identificar modelos que simultaneamente reduzem erro e consumo energético, bem como aqueles que priorizam apenas um desses objetivos.

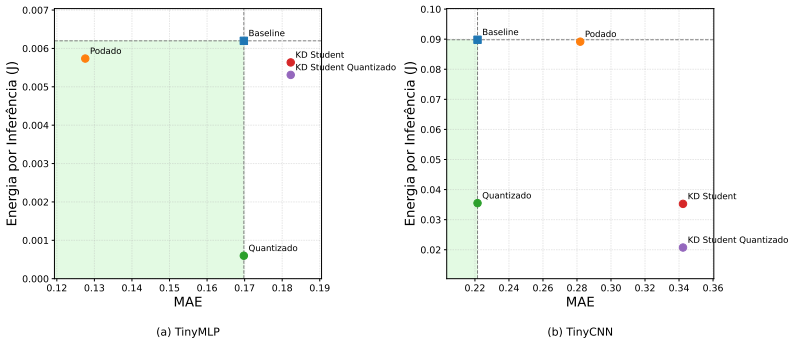

Figura 5. Trade-off entre Energia e Desempenho de Modelos TinyML

Para a arquitetura TinyMLP, a quantização apresenta a maior redução no consumo energético, diminuindo a energia por inferência em relação ao *baseline*, sem alterar o erro preditivo. Já o modelo podado apresenta o menor MAE entre os modelos avaliados, além de pequena redução no consumo de energia. Em contraste, as variantes baseadas em KD apresentam leve aumento no erro preditivo, apesar de pequenas reduções energéticas em relação ao *baseline*. Esses resultados indicam que, para TinyMLP, a quantização apresenta melhor desempenho quando a prioridade é a eficiência energética.

Para arquitetura TinyCNN, as técnicas de otimização produzem reduções mais expressivas no consumo energético. A quantização reduz substancialmente a energia por inferência mantendo o mesmo nível de erro do *baseline*. O modelo *KD Student Quantized* apresenta a maior redução energética entre todos os modelos avaliados, porém com aumento perceptível no erro preditivo. Já o modelo podado não apresenta ganhos relevantes em eficiência energética e ainda apresenta pior desempenho preditivo, o que reduz sua atratividade para implantação em dispositivos de borda. Em geral, os resultados evidenciam que diferentes técnicas de otimização favorecem diferentes objetivos. A quantização se destaca como a abordagem mais eficaz para reduzir o consumo energético mantendo desempenho preditivo semelhante ao modelo original, enquanto outras técnicas podem oferecer melhorias em determinadas métricas. Esses resultados reforçam a importância de considerar explicitamente o *trade-off* entre desempenho preditivo e eficiência energética ao selecionar modelos para aplicações de TinyML em dispositivos embarcados.

5. Conclusão e Trabalhos Futuros

Este trabalho analisou o impacto de diferentes técnicas de otimização de modelos de Edge AI no desempenho preditivo, na latência de inferência e no consumo energético em dispositivos embarcados. Para isso, foram avaliadas duas arquiteturas leves, TinyMLP e TinyCNN, aplicadas à previsão de precipitação a partir de dados meteorológicos reais. Após o treinamento dos modelos base, foram investigadas três técnicas amplamente utilizadas em TinyML: quantização pós-treinamento, poda de pesos e KD. Os modelos foram convertidos para TFLite e executados em um Arduino Nano 33 BLE Sense, enquanto o consumo energético foi medido por meio do sensor INA219.

Os resultados mostram que a quantização apresentou os maiores ganhos em eficiência energética e redução de latência, mantendo praticamente inalterado o desempenho preditivo. No caso do TinyMLP, a quantização reduziu o consumo energético em aproximadamente 90% e diminuiu significativamente o tempo de inferência em relação ao *baseline*. Para o TinyCNN, também foram observadas reduções expressivas de energia e latência. Em contraste, a poda apresentou impacto limitado no consumo energético, enquanto os modelos baseados em KD reduziram o custo computacional, mas com leve degradação do desempenho. Esses resultados evidenciam o equilíbrio entre acurácia, custo computacional e eficiência energética em aplicações TinyML e indicam que a quantização constitui uma estratégia particularmente eficaz para implantação de modelos de ML em dispositivos de borda com recursos restritos.

Os resultados obtidos fornecem evidências relevantes sobre o impacto de técnicas de otimização na eficiência de modelos TinyML executados em microcontroladores. Em particular, os ganhos observados com a quantização indicam que essa técnica pode viabilizar a implantação de modelos de ML em dispositivos de borda, reduzindo significativamente o consumo energético e o tempo de inferência. Essas evidências são especialmente relevantes para aplicações IoT sensíveis a energia, como estações meteorológicas inteligentes, sistemas de monitoramento ambiental e redes de sensores distribuídas, nas quais a execução local dos modelos pode aumentar a autonomia energética e reduzir a dependência de processamento em nuvem. Entretanto, os resultados estão condicionados ao ambiente experimental adotado, incluindo o uso do Arduino Nano 33 BLE Sense, um conjunto específico de dados meteorológicos e medições realizadas com o sensor INA219, além da análise do consumo energético restrita à fase de inferência, o que pode limitar a generalização para outros cenários de hardware e aplicações.

Como trabalhos futuros, pretende-se expandir esta investigação avaliando os modelos em outras plataformas de Edge AI, como ESP32 e Raspberry Pi. Outra direção promissora consiste em explorar diferentes modelos TinyML e técnicas adicionais de compressão, incluindo quantização sensível à camada, poda estruturada e métodos automatizados de busca de arquitetura. Esse tipo de avaliação pode contribuir para o desenvolvimento de soluções mais eficientes para implantação em dispositivos de borda.

Agradecimentos

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brasil, pelo apoio financeiro por meio do projeto nº 401147/2025-8.

Referências

Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., and Hafid, A. S. (2023). A comprehensive survey on tinyml. *IEEE access*, 11:96892–96922.

Bhushan, C. M., Koppuravuri, P., Prasanthi, N., Gazi, F., Hussain, M. M., Abdussami, M., Devi, A. A., and Faizi, J. (2025). Deploying tinyml for energy-efficient object detection and communication in low-power edge ai systems. *Scientific Reports*.

Bourechak, A., Zedadra, O., Kouahla, M. N., Guerrieri, A., Seridi, H., and Fortino, G. (2023). At the confluence of artificial intelligence and edge computing in iot-based applications: A review and new perspectives. *Sensors*, 23(3):1639.

Breder, G. B., Vasconcelos, G., and Ferro, M. (2025). Beyond accuracy: A comparative study of machine learning models for extreme weather forecasting in rio de janeiro with a green ai perspective. In *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*, pages 1996–2007. SBC.

Choudhary, A. (2024). Internet of things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions. *Discover Internet of Things*, 4(1):31.

de Albuquerque, A. S. and de Araujo, D. R. (2025). Short-term rainfall forecasting aided by iot and deep learning. In *2025 IEEE Latin Conference on IoT (LCIoT)*, pages 107–110. IEEE.

Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S., and López, O. L. (2024). Tinyml: Tools, applications, challenges, and future research directions. *Multimedia Tools and Applications*, 83(10):29015–29045.

Kumar, R., Goel, R., Sidana, N., Sharma, A. P., ghai, S., Singh, T., singh, R., Priyadarshi, N., Twala, B., and Ahmad, V. (2024). Enhancing climate forecasting with ai: Current state and future prospect. *F1000Research*, 13:1094.

Liyew, C. M. and Melese, H. A. (2021). Machine learning techniques to predict daily rainfall amount. *Journal of Big Data*, 8(1):153.

Mnif, M., Sahnoun, S., Saad, Y. B., Fakhfakh, A., and Kanoun, O. (2024). Combinative model compression approach for enhancing 1d cnn efficiency for eit-based hand gesture recognition on iot edge devices. *Internet of things*, 28:101403.

Rahman, H. U., Ritu, A. T., Rasha, H. J., and Anower, M. S. (2024). Comparative analysis of microcontrollers in wearable devices for real-time arrhythmia classification: an embedded machine learning approach. In *2024 6th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT)*, pages 1327–1331. IEEE.

Scheifer, E. L., de Queiroz Pereira, G., Guimarães, A. A., Souza, L. F., and Lorenzetto, J. A. S. (2025). Uso de dados climáticos e machine learning na previsão de chuvas para gestão sustentável do ecoturismo em francisco beltrão (pr). *Revista Brasileira de Ecoturismo (RBEcotur)*, 18(5):706–717.

Sham, F. A. F., El-Shafie, A., Jaafar, W. Z. W., Sherif, M., Ahmed, A. N., et al. (2025). Advances in ai-based rainfall forecasting: A comprehensive review of past, present, and future directions with intelligent data fusion and climate change models. *Results in Engineering*, 27:105774.

Tao, Y., Hioka, Y., and Lu, Y. (2022). Experimental energy consumption analysis of neural network model compression methods on microcontrollers with applications in bird call classification. In *2022 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pages 1–8. IEEE.

Vieira, J. A., Campos, S. R., and de Sousa, G. V. M. (2025). Previsão climática no nordeste de goias: Aplicações de big data e machine learning. *Iesgo Science*, 1(1).