PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

MARCELO LUIS WALTER

ESTRATÉGIAS OTIMIZADAS DE *MACHINE LEARNING* PARA IDENTIFICAÇÃO DE DÍGITOS EM SISTEMAS EMBARCADOS PARA *SMART METERING*

MESTRADO EM INFORMÁTICA
PUCPR

CURITIBA
2025

ESTRATÉGIAS OTIMIZADAS DE *MACHINE LEARNING* PARA IDENTIFICAÇÃO DE DÍGITOS EM SISTEMAS EMBARCADOS PARA *SMART METERING*

2025

MARCELO LUIS WALTER

Estratégias Otimizadas de *Machine Learning* para Identificação de Dígitos em Sistemas Embarcados para *Smart Metering*

dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná para obtenção do título de mestre em Informática.

Pontifícia Universidade Católica do Paraná - PUCPR

Programa de Pós-Graduação em Informática - PPGIa

Orientador: MARCELO EDUARDO PELLENZ

Coorientador: EDSON EMILIO SCALABRIN

Curitiba - PR, Brasil

2025

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

W231e
2025

Walter, Marcelo Luis
Estratégias otimizadas de *machine learning* para identificação de dígitos em sistemas embarcados para *smart metering* / Marcelo Luis Walter ; orientador: Marcelo Eduardo Pellenz ; coorientador: Edson Emilio Scalabrin. – 2025.
lxxx f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2025
Bibliografia: f. lxxvii-lxxx

1. Informática. 2. Sistemas de computadores embarcados. 3. Visão por computador. 4. Eficiência energética. 5. Aumento de dados. 6. Reconhecimento de dígitos. I. Pellenz, Marcelo Eduardo. II. Scalabrin, Edson Emilio. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD. 20. ed. – 004

Bibliotecária: Luci Eduarda Wielganczuk – CRB 9/1118

MARCELO LUIS WALTER

Estratégias Otimizadas de *Machine Learning* para Identificação de Dígitos em Sistemas Embarcados para *Smart Metering*

MARCELO LUIS WALTER . – Curitiba - PR, Brasil, 2025-

lxxxi p. : il. (algumas color.) ; 30 cm.

Orientador: MARCELO EDUARDO PELLENZ

Dissertação –
Pontifícia Universidade Católica do Paraná - PUCPR
Programa de Pós-Graduação em Informática - PPGIa, 2025.

1. Data Augmentation. 2. Computer Vision. 2. Digit Recognizition.
I. Prof. Dr. Marcelo Pellenz. II. Pontifícia Universidade Católica do Paraná - PUCPR. III. Programa de Pós-Graduação em Informática - PPGIa IV. Título

MARCELO LUIS WALTER

Estratégias Otimizadas de *Machine Learning* para Identificação de Dígitos em Sistemas Embarcados para *Smart Metering*

dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná para obtenção do título de mestre em Informática.

Trabalho aprovado. Curitiba - PR, Brasil, 27 de agosto de 2025:

MARCELO EDUARDO PELLENZ
Orientador(a)

EDSON EMILIO SCALABRIN
Coorientador(a)

Professor(a)
Convidado 1

Professor(a)
Convidado 2

Professor
Convidado 3

Curitiba - PR, Brasil
2025

Este trabalho é dedicado a meu filho, minha esposa e minha família,
que sempre apoiaram meus sonhos e escolhas
e foram os que mais dedicaram esforços para que
eu atingisse meus objetivos

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 13 de agosto de 2025.

84-2025

DECLARAÇÃO

Declaro para os devidos fins, que **MARCELO LUIS WALTER** defendeu a dissertação de Mestrado intitulada "**Estratégias Otimizadas de ML para Identificação de Dígitos em Sistemas Embarcados para Smart Metering"**, na área de concentração Ciência da Computação no dia 03 de junho de 2025, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade, firmo a presente declaração.

gov.br
Documento assinado digitalmente
JEAN PAUL BARDDAL
Data: 15/08/2025 14:49:52-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Jean Paul Barddal
Coordenador do Programa de Pós-Graduação em Informática

Agradecimentos

Agradeço a Deus, que me guiou nesse caminho. Também agradeço a PUCPR por essa oportunidade, assim como ao programa de bolsa CAPES e Fundação Araucária. Da mesma forma agradeço ao meu orientador e meu co-orientador que me guiaram com tanta dedicação, à empresa Sumersoft que proporcionou o projeto e ambiente que são base deste trabalho e por fim aos meus amigos e colegas participantes dessa jornada. Sem eles, o trabalho seria muito mais árduo e penoso.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

"Eu acredito em uma coisa -
que apenas uma vida vivida para os outros
é uma vida que vale a pena ser vivida."
Albert Einstein

Resumo

O avanço das cidades inteligentes demanda redes heterogêneas de medidores de água, gás e eletricidade capazes de operar de forma autônoma, segura e com consumo mínimo de energia. Nessas redes, milhares de dispositivos AIoT — fabricados por diferentes fornecedores e sujeitos a condições ambientais adversas — precisam interpretar localmente dígitos em mostradores analógicos, reduzindo o tráfego de dados e prolongando a vida útil da bateria.

Diante desse cenário, esta dissertação investiga estratégias de *machine learning* (ML) para medição inteligente (*smart metering*) em microcontroladores ESP32-S3 de baixo custo. Inicialmente avaliou-se um conjunto de arquiteturas convolucionais (CNN): Lenet5, TinyML, MobileNet, MiniVGG e SqueezeNet, sendo que a versão quantizada do *LeNet-5* obteve **98,9 %** de acurácia com **apenas 300kB** de memória e **21 min** para inferir 10.000 imagens. Esta versão configurou-se com o melhor compromisso entre precisão, latência e consumo energético.

Uma vez encontrado o melhor modelo CNN para o ambiente restritivo escolhido, o estudo concentrou-se em determinar a eficiência de técnicas de *data augmentation* (DA) para a busca de acurácia na tarefa de deteção de dígitos.

Assim, aplicaram-se nove técnicas clássicas de *data augmentation* a três bases de dígitos — Kaggle NumOCR, SCUT-WMN e o conjunto proprietário SmartEyes — variando-se parâmetros de cada técnica bem como o volume de novas imagens geradas (0–500%). O protocolo padronizado permitiu medir o *coeficiente angular da tendência* da acurácia; ruído gaussiano moderado e translação horizontal curta proporcionaram ganhos de até +1,33%, enquanto rotações amplas degradaram o desempenho. Verificou-se ponto de saturação após ∼400 % de aumento, quando surgem sobreajuste e custos computacionais adicionais.

Como contribuição, o trabalho entrega um *framework* replicável para otimizar modelos leves sob severas restrições de memória e energia, além de diretrizes para o emprego futuro de técnicas automáticas de DA (*AutoDA*) parametrizadas por classe.

Palavras-chave: Aumento de dados, AIoT, Reconhecimento de Dígitos, Medição Inteligente, Visão Computacional, Sistemas Embarcados, Eficiência Energética.

Abstract

The advancement of smart cities requires heterogeneous networks of water, gas, and electricity meters that operate autonomously, securely, and with minimal energy consumption. Within these networks, thousands of AIoT devices—manufactured by different vendors and exposed to adverse environmental conditions—must locally interpret the digits displayed on analog dials, thereby reducing data traffic and extending battery life.

This dissertation investigates machine-learning strategies for smart metering on low-cost ESP32-S3 microcontrollers. An initial evaluation of several convolutional architectures showed that a quantised *LeNet-5* achieves **98.9 %** accuracy while occupying only **300 kB** of memory and requiring **21 min** to infer 10 000 images, representing the best trade-off between precision, latency, and energy use.

Subsequently, nine classical data-augmentation techniques were applied to three digit datasets—Kaggle NumOCR, SCUT-WMN, and the proprietary SmartEyes—sweeping intensities and volumes (0–500 %). A standardised protocol enabled the measurement of the *trend angular coefficient* of accuracy; moderate Gaussian noise and short horizontal translations yielded gains of up to **+1.33 p.p.**, whereas large rotations degraded performance. A saturation point emerged after approximately 400 % expansion, where overfitting and additional computational costs appear.

The dissertation delivers a replicable framework for optimising lightweight models under severe memory and energy constraints and provides guidelines for the future adoption of automatic augmentation techniques (AutoDA) parameterised by class.

Keywords: Data Augmentation, Machine Learning, Digit Recognition, Smart Metering, Computer Vision, Embedded Systems, Energy Efficiency.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AIoT	Internet das Coisas com Inteligência Artifical, do inglês *Artificial Inteligence / Internet of Things*
CNN	Rede Neural Convulacional, do inglês *Convulational Neural Network*
DA	Aumento do volume de dados, do inglês *Data Augmentation*
IA	Inteligência Artificial
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos, do inglês *Institute of Electrical and Electronics Engineers*
IoT	Internet das Coisas, do inglês *Internet of Things*
LoRa	Dispositivo de rádio freqência de Longo Alcance, do inglês *Long Range*
LoRaWAN	Rede de dispositivos LoRa, do inglês *Long Range Wide-Area Networks*
MB	Megabytes (1024 kilobytes)
ML	Aprendizado de Máquina, do inglês *Machine Learning*
SRAM	Memória de Acesso Randômica Estática, do inglês *Static Random Access Memory*

Sumário

1 Introdução

A consolidação do paradigma de *cidades inteligentes* pressiona serviços públicos a adotarem redes de monitoramento capazes de operar de forma autônoma, segura e energeticamente eficiente. Nesse contexto, milhares de medidores de água, gás ou eletricidade — fabricados por fornecedores distintos, instalados em ambientes adversos e alimentados por baterias de longa duração — precisam extrair localmente a informação de consumo e transmiti-la de modo esparso (AWAN, 2023). A convergência entre **Inteligência Artificial** e **Internet das Coisas** (**AIoT**) oferece a esses dispositivos a capacidade de interpretar, *in loco*, os dígitos impressos nos mostradores analógicos, reduzindo tráfego de dados, latência e custos operacionais.

Embora CNNs de alto desempenho resolvam o problema em servidores ou *cloud*,portar para microcontroladores de baixo custo impõe severas restrições de *hardware* (2–8 MB de *SRAM*, clock abaixo de 240 MHz) e de orçamento energético. Portanto, podemos destacar dois desafios:

I. **Selecionar arquiteturas** que mantenham acurácia competitiva sem exceder a memória disponível nem deteriorar o tempo de inferência;

II. **Ampliar a robustez** dos modelos frente a variações de iluminação, ruído, ângulo e desgaste mecânico dos hidrômetros, mesmo quando se dispõe de conjuntos de treinamento limitados.

O objetivo central desta dissertação é investigar estratégias e desenvolver uma metodologia que combinem *machine learning* leve com técnicas de *data augmentation* detalhadamente parametrizadas, de modo a maximizar a acurácia do reconhecimento de dígitos em microcontroladores ESP32-S3 — escolhidos por serem amplamente adotados em soluções comerciais de *smart metering*.

Dessa forma, a dissertação contribui para o avanço do estado da arte em *smart metering* ao oferecer soluções replicáveis e energeticamente viáveis para leitura automática de dígitos em ambientes reais de cidades inteligentes.

Contribuições

1) Protocolo padronizado que mede o *coeficiente angular da tendência* para quantificar o ganho de acurácia em função do volume sintético gerado.

2) Demonstração de que o *LeNet-5* quantizado atinge 98,9 % de acurácia ocupando apenas 300 kB — valor compatível com ESP32-S3 — e inferindo 10.000 imagens em 21 min.

3) Identificação de faixas ótimas: ruído gaussiano moderado e translações horizontais curtas adicionam até +1,33%, enquanto rotações abundantes degradam o desempenho.

4) Diretrizes para adoção futura de AutoDA parametrizado por classe, reduzindo ensaios empíricos e consumo de energia durante o treinamento *on-device*.

Estrutura da Dissertação

A pesquisa foi estruturada em duas frentes complementares:

- **Avaliação de modelos**: comparação experimental de nove arquiteturas (LeNet-5, TinyML-CNN, MobileNet, MiniVGG, entre outras) antes e depois da quantização para TensorFlow Lite Micro;
- **Otimização do espaço de entrada**: aplicação sistemática de nove técnicas clássicas de aumento de dados, cada qual em três faixas de intensidade e cinco incrementos de volume (0–500 %), sobre três bases de dígitos distintas (Kaggle NumOCR (NAVARRO; JAISWAL,), SCUT-WMN (YANG et al., 2019) e o conjunto proprietário *SmartEyes*).

Além deste capítulo introdutório, o trabalho organiza-se da seguinte forma:

Chapter 2 Fundamentação Teórica — apresenta os princípios de visão computacional, redes neurais convolucionais, quantização de modelos e protocolos de comunicação em AIoT que dão suporte à pesquisa.

Chapter 3 Revisão do Estado da Arte — discute trabalhos recentes sobre leitura automática de hidrômetros, portabilidade de modelos para microcontroladores e técnicas de *data augmentation* em dispositivos de borda.

Chapter 4 Contexto e Objetivos — descreve o projeto *SmartEyes*, delimita as perguntas de pesquisa e estabelece os critérios de avaliação empregados ao longo do estudo.

Chapter 5 Avaliação de Modelos de *Machine Learning* — compara nove arquiteturas CNN executadas em ESP32-S3, analisando acurácia, uso de memória, tempo de inferência e viabilidade de implantação.

Chapter 6 Aprimoramento via *Data Augmentation* — investiga, de forma sistemática, nove técnicas clássicas de aumento de dados em três faixas de intensidade e cinco volumes, identificando o ponto de saturação e as combinações que maximizam a performance.

Chapter 7 Escaneamento de Características de Imagens — apresenta um estudo exploratório que cataloga variáveis fotométricas e geométricas dos dígitos capturados, formando a base para futuras estratégias de *AutoDA* com parametrização adaptativa por classe.

Chapter 8 Conclusões e Trabalhos Futuros — resume as contribuições, aponta limitações, e propõe extensões, incluindo o uso de AutoDA com parametrização distinta por classe baseada em escaneamento prévio das imagens.

1.1 Motivação

A leitura automática de hidrômetros insere-se em um panorama mais amplo de *smart cities*, onde milhares de dispositivos — muitas vezes heterogêneos em arquitetura e fabricante — precisam operar de forma autônoma, por longos períodos e com consumo mínimo de energia. Esses nós AIoT (ZHANG; TAO, 2020) coletam, processam e transmitem dados de consumo por redes de baixa potência, como a LoRaWAN (ZOU et al., 2021a), sustentando decisões de gestão hídrica em tempo quase real.

Embora a visão computacional resolva o problema de reconhecimento de dígitos em servidores ou na nuvem, transferir essa tarefa para microcontroladores ESP32-S3 — dotados de apenas 2 a 8MB de SRAM e orçados em poucos miliwatts — impõe desafios específicos:

(a) restrição severa de memória e latência para execução de CNNs;

(b) limitação de energia em cenários alimentados a bateria;

(c) bases de imagens pequenas e enviesadas, sobretudo quando capturadas em campo sob iluminação e desgaste variáveis.

Tais fatores ampliam o risco de sobreajuste e motivam o uso de *data augmentation* (DA) cuidadosamente calibrado.

Assim, o trabalho que orienta essa dissertação configurando o Problema/Questão de Pesquisa é: **"Levantar, analisar resultados experimentais e discutir diretrizes que indiquem como selecionar um modelo convolucional leve apropriado, bem como parametrizar as transformações de DA de modo a maximizar a acurácia**

de detecção de dígitos em ambientes IoT sem sacrificar o orçamento limitado de recursos de memória, processamento e energia."

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral desta dissertação é desenvolver uma metodologia eficiente para a aplicação de técnicas de *data augmentation* em *datasets* de imagens, visando otimizar o desempenho de modelos de *machine learning* embarcados em sistemas AIoT para *Smart Metering*. A proposta inclui a identificação e aplicação dos melhores parâmetros de *data augmentation* para melhorar a precisão e a robustez do reconhecimento de dígitos em medidores inteligentes de água.

1.2.2 Objetivos Específicos

- **Analisar a Variabilidade dos Dados**: Realizar um estudo detalhado sobre a variabilidade presente nas imagens dos medidores de água, incluindo fatores como ruído, brilho, contraste, translações, rotações, desfoque e deformações.
- **Identificar Parâmetros de Data Augmentation**: Desenvolver uma abordagem sistemática para escanear *datasets* de imagens e identificar os melhores parâmetros de *data augmentation* que possam ser aplicados para melhorar a qualidade dos dados de treinamento.
- **Implementar Técnicas de Data Augmentation**: Aplicar as técnicas de *data augmentation* identificadas nos *datasets* de imagens, criando um conjunto diversificado de dados de treinamento para os modelos de *machine learning*.
- **Avaliar o Desempenho dos Modelos**: Treinar modelos de *machine learning* com os *datasets* aumentados e avaliar o desempenho dos modelos em termos de precisão, robustez e capacidade de generalização.
- **Desenvolver um Framework de Automatização**: Criar um *framework* que automatize o processo de identificação dos melhores parâmetros de *data augmentation* e a aplicação dessas técnicas em novos *datasets* de imagens.
- **Validar a Metodologia em Cenários Reais**: Realizar testes práticos e validações da metodologia desenvolvida em cenários reais de *Smart Metering*, verificando a eficácia das técnicas propostas na melhoria do reconhecimento de dígitos.

- **Documentar e Divulgar os Resultados**: Documentar todo o processo, desde a análise inicial até os resultados finais, e divulgar as conclusões em publicações científicas e apresentações em conferências, contribuindo para o avanço do conhecimento na área de visão computacional e *machine learning* em sistemas AIoT.

2 Fundamentação Teórica

Antes de abordar os aspectos do trabalho, é necessário definir e entender alguns princípios que complementam a proposta. Portanto esta seção define os conceitos básicos de AIoT, modelos de *machine learning* para identificação de dígitos e técnicas de *data augmentation*.

2.1 AIoT

A convergência entre a Inteligência Artificial (IA) e a Internet das Coisas (IoT) deu origem ao conceito de AIoT (*Artificial Intelligence of Things*). Este termo refere-se à integração de capacidades de IA em dispositivos IoT, proporcionando a esses dispositivos a habilidade de processar dados e tomar decisões de forma autônoma e em tempo real. A AIoT representa uma evolução significativa na tecnologia, permitindo novas aplicações para dispositivos inteligentes e conectados, especialmente em ambientes com restrições de recursos, como é o caso de sistemas embarcados para medição inteligente (*smart metering*). Esses sistemas são capazes de operar de forma autônoma e eficiente em uma ampla variedade de aplicações, incluindo saúde inteligente, cidades inteligentes, agricultura de precisão e manufatura avançada.

Segundo (ZHANG; TAO, 2020), a AIoT marca um avanço tecnológico que habilita dispositivos inteligentes a realizar processamento de dados imediato, beneficiando-se do *edge computing* para melhorar a privacidade e reduzir tempos de resposta. Ao contrário do *machine learning* tradicional, que requer grande poder computacional, o *machine learning* orientado para AIoT adapta essas capacidades para dispositivos de borda com baixa potência e recursos limitados.

Em aplicações de medição inteligente, como a leitura automática de dígitos de medidores de utilidades, a AIoT possibilita a implementação de técnicas de visão computacional diretamente em dispositivos embarcados. Este processo envolve a adaptação e a avaliação de modelos de *machine learning* para serem executados eficientemente em dispositivos de baixo custo e com limitações de memória, como microcontroladores ESP32-S3.

A fusão de IA com IoT na AIoT não só melhora a eficiência operacional dos dispositivos, mas também amplia as capacidades dos sistemas de medição inteligente, permitindo que operem de maneira mais autônoma e precisa. Esta integração é crucial para o desenvolvimento de sistemas inteligentes que requerem respostas rápidas e seguras, utilizando recursos computacionais limitados de forma eficaz (SS; Z; IA, 2023). Uma arquitetura AIoT integrando leitores de água e aplicação de reconhecimento de dígitos

através de visão computacional é apresentada por (ZOU et al., 2021a) demonstrando como imagens podem ser carregadas nos devices e processadas por um servidor, conforme ilustrado na Figura (1).

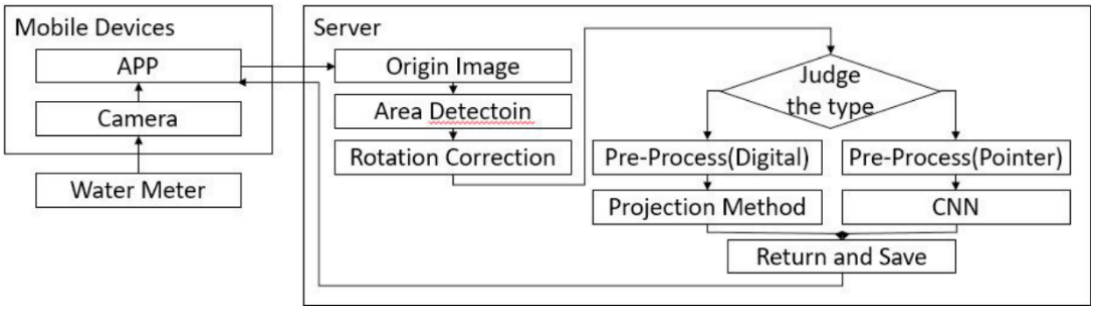


Figura 1 – Arquitetura exemplo de um sistema AIoT (ZOU et al., 2021a)

2.2 Machine Learning Aplicado à Visão Computacional

Uma técnica de IA que vem sendo adotada atualmente em dispositivos IoT é a aprendizagem de máquina embarcada, configurando o que se entende pela definição AIoT. E, sob este contexto, uma das técnicas que mais se faz necessária é a aplicação de visão computacional para a interpretação de características visuais das informações coletadas. A capacidade de aplicar *machine learning* diretamente nos dispositivos de borda, em vez de depender exclusivamente da nuvem, reduz a latência, melhora a privacidade dos dados e aumenta a eficiência energética, já que reduz o gasto de eletricidade para o processamento e transmissão das imagens.

Um dos principais desafios enfrentados pela AIoT é a limitação de recursos dos dispositivos de borda, que frequentemente possuem capacidade de processamento e memória limitadas. Isso torna a adoção de modelos de *machine learning* compactos essencial para garantir a eficiência desses sistemas. Entre os modelos de *machine learning* compactos mais relevantes estão LeNet, MobileNet, TinyML, MiniVGG e SqueezeNet (AL, 2017). Esses modelos foram projetados para serem eficientes em termos de computação e uso de memória, permitindo sua implementação em dispositivos embarcados com recursos limitados.

2.2.1 LeNet5

LeNet5, desenvolvido por (HAFFNER, 1998), é um dos primeiros modelos de rede neural convolucional (CNN) amplamente utilizados para reconhecimento de dígitos manuscritos. Estudos recentes como o de (SAINI; TRIPATHI; MIRZA, 2023) demonstram ainda sua grande acuracidade mesmo frente a métodos contemporâneos. Ele possui uma arquitetura simples, composta por duas camadas convolucionais seguidas por camadas de *pooling* e, finalmente, camadas totalmente conectadas. As principais vantagens do LeNet5

incluem sua eficiência em termos de computação e sua capacidade de ser treinado com um conjunto de dados relativamente pequeno, tornando-o adequado para dispositivos embarcados com recursos limitados. Conforme demonstrado na Figura 2 a arquitetura do modelo de LeNet-5 recebe uma imagem de 32x32 pixeis a qual é tratada pelas seguintes camandas:

1. Camada C1 (Convolucional): 6 mapas de características (28x28), com receptivos de 5x5. Total: 156 parâmetros treináveis.

2. Camada S2 (Subamostragem): 6 mapas de características (14x14), com receptivos de 2x2. Total: 12 parâmetros treináveis.

4. Camada C3 (Convolucional): 16 mapas de características, conectados a subconjuntos dos mapas da S2. Total: 1.516 parâmetros treináveis.

4. Camada S4 (Subamostragem): 16 mapas de características (5x5), conectados a 2x2 receptivos de C3. Total: 32 parâmetros treináveis.

5. Camada C5 (Convolucional): 120 mapas de características (1x1), conectados a todos os mapas da S4. Total: 48.120 parâmetros treináveis.

6. Camada F6 (Totalmente Conectada): 84 unidades, conectadas à C5. Total: 10.164 parâmetros treináveis.

7. Camada de Saída: Unidades RBF Euclidianas, uma para cada classe, calculando a distância Euclidiana.

A LeNet-5 utiliza funções sigmoides e unidades RBF para a classificação final das imagens

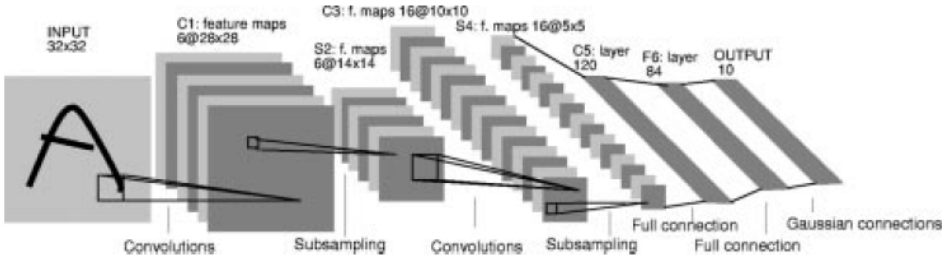


Figura 2 – Arquitetura do modelo LeNet conforme (HAFFNER, 1998)

2.2.2 TinyML

TinyML refere-se ao uso de modelos de *machine learning* em dispositivos extremamente limitados em termos de energia e recursos computacionais. Um exemplo notável é o artigo de (RAY, 2022), que discute a aplicação de TinyML em dispositivos IoT. Eles destacam a eficiência energética e a capacidade de processamento de modelos compactos, como o TensorFlow Lite, que permite a execução de inferências de *machine learning*

diretamente em microcontroladores. A Figura 3 do artigo de (RAY, 2022) ilustra o uso de TinyML de forma eficiente, destacando como os modelos podem ser otimizados para operar em dispositivos com recursos extremamente limitados.

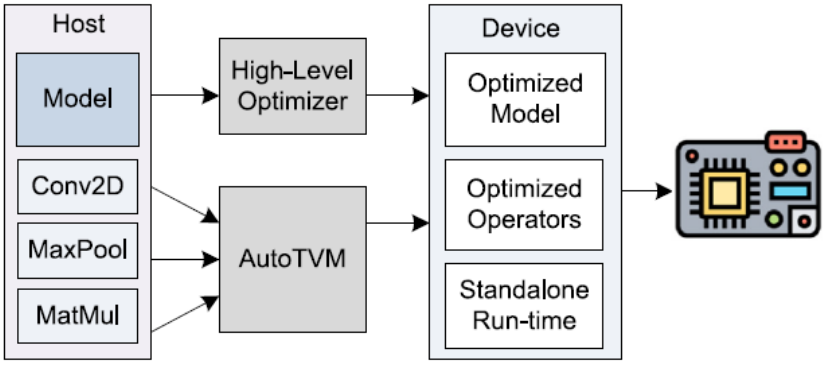


Figura 3 – Arquitetura de um sistema TinyML eficiente, destacando a otimização de modelos para operar em dispositivos com recursos limitados (RAY, 2022).

2.2.3 MobileNet

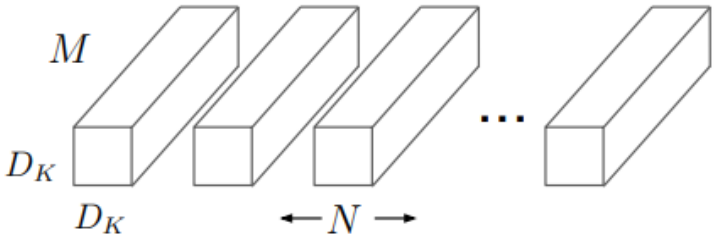
MobileNet, desenvolvido por (ADAM, 2017), é uma classe de modelos de rede neural projetados especificamente para aplicações móveis e embarcadas. Várias aplicações reais de tal modelo utilizando dispositivos com recursos limitados podem ser encontrados conforme demonstrado na Figura 5. Utiliza convoluções separáveis em profundidade para reduzir significativamente a complexidade computacional e o tamanho do modelo, mantendo uma alta precisão. As principais características do MobileNet incluem dois hiperparâmetros globais que permitem ajustar o modelo para diferentes restrições de recursos, proporcionando um equilíbrio eficiente entre latência e precisão. A Figura 4 do artigo de (ADAM, 2017) ilustra a estrutura de convoluções separáveis em profundidade, chamada de *Depthwise Separable Convolution* demonstrando como essa abordagem reduz a carga computacional e o tamanho dos modelos de redes neurais.

2.2.4 MiniVGG

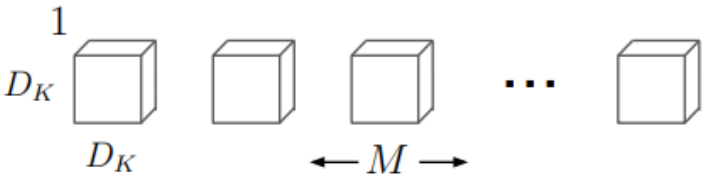
MiniVGG proposto por (KUMAR et al., 2021) é uma versão compacta do famoso modelo VGG, projetado para ser eficiente em termos de computação sem comprometer significativamente a precisão. A principal vantagem do MiniVGG é sua capacidade de ser implementado em dispositivos embarcados com memória limitada, mantendo uma estrutura de rede profunda que facilita a captura de características complexas nas imagens.

2.2.5 SqueezeNet

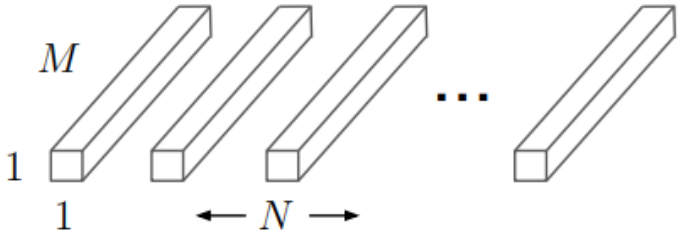
SqueezeNet, desenvolvido por (KEUTZER, 2016), foi projetado para atingir a precisão do AlexNet, mas com 50 vezes menos parâmetros, o que resulta em um modelo



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figura 4 – Filtros de convulucoes conforme desctito por (ADAM, 2017)

de menos de 0,5 MB. Utiliza a estratégia de *squeeze* para reduzir o número de filtros nas camadas convolucionais e *expand* para aumentar a complexidade da rede apenas quando necessário. Essa abordagem resulta em um modelo altamente compacto e eficiente, ideal para dispositivos embarcados. A Figura 6 do artigo de (KEUTZER, 2016) ilustra

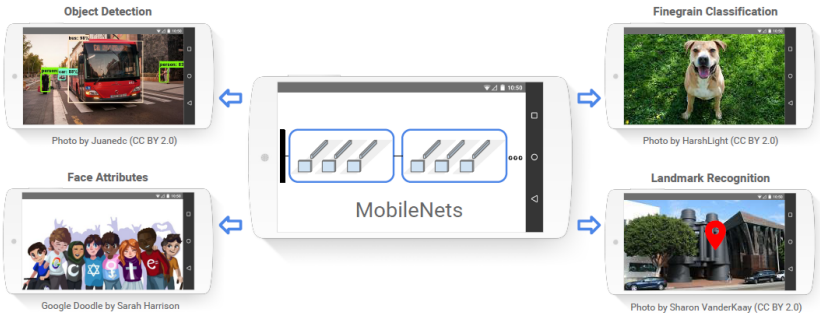


Figura 5 – Exemplo de arquitetura MobileNet aplicada em tarefas de visão computacional conforme (ADAM, 2017)

a arquitetura do SqueezeNet, mostrando como as camadas de squeeze e expand são combinadas para formar a rede.

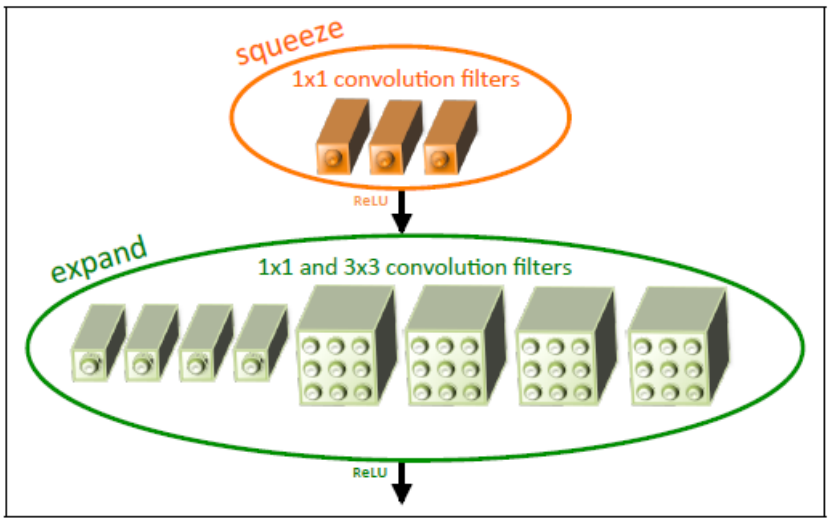


Figura 6 – Organização de convoluções do SqueezeNet conforme (KEUTZER, 2016)

2.3 Data Augmentation

Data Augmentation é uma técnica essencial no treinamento de modelos de machine learning, especialmente quando os dados disponíveis são limitados em termos de volume ou com muitas características não padronizadas. Esta técnica visa aumentar a quantidade e a diversidade dos dados de treinamento, aplicando transformações aos dados originais. A seguir, detalhamos algumas das técnicas mais comuns de data augmentation, conforme demonstrado pela Figura 7:

Noise - Adicionar ruído às imagens é uma técnica comum de data augmentation que ajuda os modelos a generalizarem melhor, tornando-os robustos a pequenas variações na entrada. O ruído pode ser Gaussiano, Poisson ou Speckle, entre outros. Segundo (KHOSH-GOFTAAR, 2019), adicionar ruído pode melhorar significativamente o desempenho do modelo em cenários do mundo real uma vez que geram conjuntos de imagens com graus similares de ruído.

Brightness - Ajustar o brilho das imagens pode simular diferentes condições de iluminação. Esta técnica envolve aumentar ou diminuir os valores dos pixels de forma uniforme. Estudos de (AKBıYıK; ÖZYURT; ÇELIK, 2019) demonstram que variar os níveis de brilho pode ajudar os modelos a se tornarem invariantes às mudanças nas condições de iluminação.

Contrast - Ajustar o contraste modifica a diferença na luminância ou na cor que torna um objeto distinguível. Aumentar o contraste pode tornar as características mais proeminentes, enquanto diminuí-lo pode tornar o modelo mais sensível a diferenças sutis.

(GULSHAD, 2021) descobriram que ajustes de contraste podem melhorar a robustez do modelo em diversas condições ambientais.

Translation - A translação envolve deslocar o conteúdo da imagem ao longo do eixo x ou y, simulando mudanças na posição do objeto dentro do quadro. Esta técnica ajuda os modelos a aprenderem a invariância posicional. (VEELING et al., 2018) discutem como a translação pode ser crucial para aplicações de imagem médica, onde a posição das características pode variar entre os exames.

Rotation - A rotação das imagens em ângulos aleatórios pode ajudar os modelos a aprenderem a reconhecer objetos independentemente da sua orientação. Esta técnica é especialmente útil em aplicações como patologia, onde a orientação das células pode variar. (ZHANG, 2019) mostram que a rotação pode melhorar significativamente o desempenho das CNNs na detecção de objetos rotacionados.

Blur - A aplicação de desfoque nas imagens pode simular condições de captura de imagem não ideais. Técnicas como o desfoque Gaussiano são comumente utilizadas. (HOLZINGER, 2019) destacam que a augmentação com desfoque pode ajudar os modelos a aprenderem a reconhecer objetos mesmo quando não estão em foco perfeito, melhorando sua capacidade de generalização.

Elastic Deformation - A deformação elástica envolve distorcer a imagem de maneira não rígida, o que pode ser particularmente útil para aumentar conjuntos de dados em imagens médicas. (NITSCHKE, 2018) demonstram que deformações elásticas podem ajudar a melhorar o aprendizado de formas e estruturas, aumentando a robustez dos modelos de segmentação.

Shearing - O cisalhamento envolve distorcer a imagem ao longo de um eixo, criando um efeito de inclinação. Esta técnica pode ajudar os modelos a aprenderem invariâncias geométricas. Segundo(NANNI et al., 2021), o cisalhamento pode melhorar a robustez dos modelos em cenários onde os objetos podem ser visualizados de diferentes ângulos.

Grayscale - Converter imagens para escala de cinza pode ajudar os modelos a focarem na textura e na forma, em vez de na cor. Esta técnica é particularmente útil em ambientes onde a informação de cor é menos confiável. Estudos de (HINTON, 2012) indicam que a augmentação com escala de cinza pode ser benéfica para tarefas de classificação baseadas em textura.

Binarization - A binarização converte as imagens em formato binário (preto e branco), o que pode ser útil para certos tipos de tarefas de reconhecimento de imagens, como o reconhecimento de texto. (AKBıYıK; ÖZYURT; ÇELIK, 2019) explicam que a binarização pode melhorar a detecção de bordas e simplificar a estrutura da imagem, facilitando o aprendizado de características importantes pelos modelos.

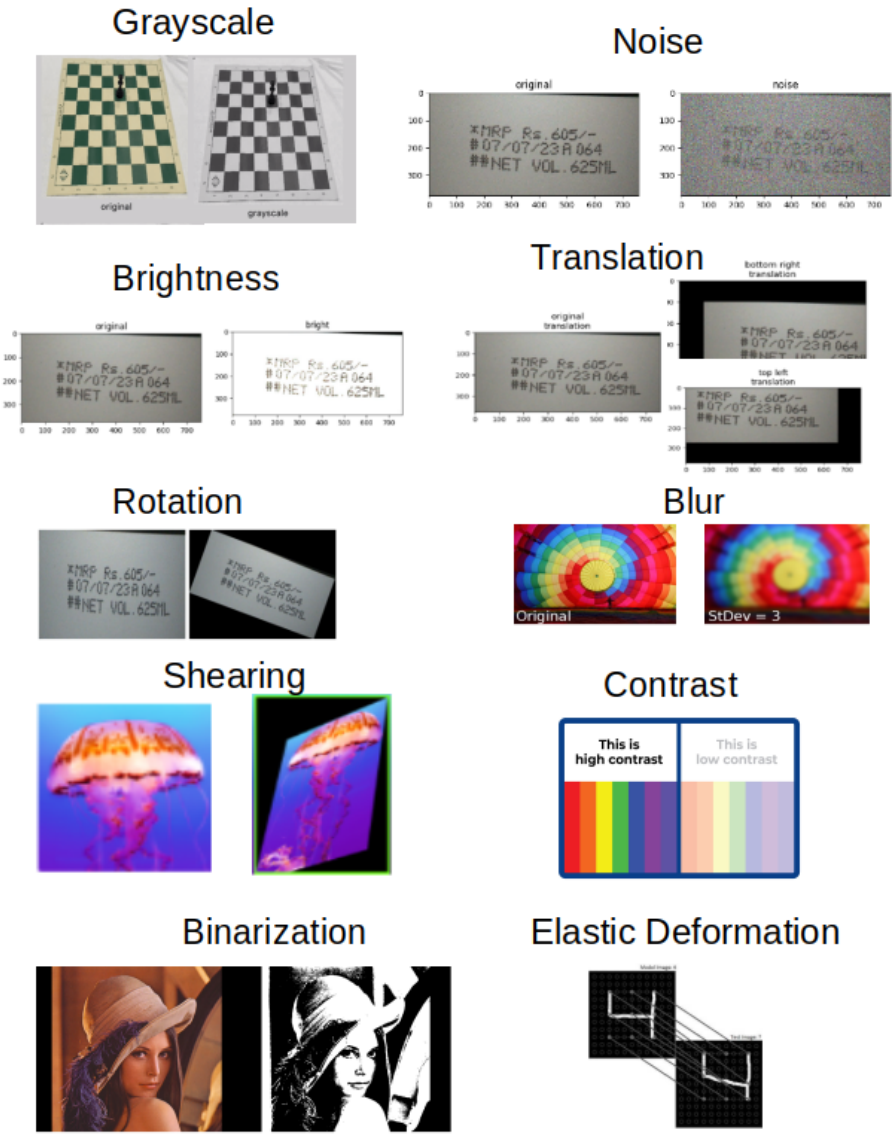


Figura 7 – Algumas Técnicas de Data Augmentation Fonte: O Autor

2.4 Smart Metering

2.4.1 Introdução

Smart metering, ou medição inteligente, é uma tecnologia fundamental na era das cidades inteligentes, proporcionando uma maneira eficiente e precisa de monitorar o consumo de recursos como água, gás e eletricidade. Esta tecnologia utiliza dispositivos de medição avançados que não apenas registram o consumo, mas também transmitem os dados coletados para sistemas centrais em tempo real. Isso permite uma gestão mais eficiente dos recursos, ajudando na redução de desperdícios e no melhor planejamento de consumo e distribuição.

2.4.2 Tecnologias de Comunicação

Os sistemas de medição inteligente (smart meterings) têm transformado a gestão de recursos como energia e água, permitindo comunicação bidirecional entre medidores e centrais de controle para coleta de dados em tempo real e gestão eficiente do consumo. Para isso, várias tecnologias de comunicação sem fio são empregadas, incluindo RF Mesh, Zigbee, Cellular (3G/4G/5G) e LoRa/LoRaWAN (ZORZI, 2016).

RF Mesh cria uma rede de malha onde cada medidor retransmite dados até a central, ideal para grandes áreas, mas com desafios de interferência e consumo de energia. Zigbee, com baixo consumo de energia e curto alcance, é ideal para medidores próximos, mas menos eficaz para longas distâncias. As redes celulares (3G/4G/5G) oferecem alta largura de banda e cobertura, mas com custos operacionais e de energia mais elevados.

LoRa (Long Range) e o protocolo LoRaWAN (Long Range Wide Area Network), por sua vez, destacam-se por permitir comunicação a longas distâncias com baixo consumo de energia, utilizando frequências não licenciadas.

2.4.2.1 LoRa e LoRaWAN

LoRa é uma tecnologia de modulação que permite a comunicação a longas distâncias com baixo consumo de energia. LoRaWAN é um protocolo de comunicação que opera em camadas superiores, permitindo a conexão de dispositivos IoT a uma rede LoRa. Essas tecnologias são amplamente utilizadas em aplicações de smart metering devido à sua capacidade de suportar um grande número de dispositivos com baixa taxa de transferência de dados, características detalhadas por (ZORZI, 2016).

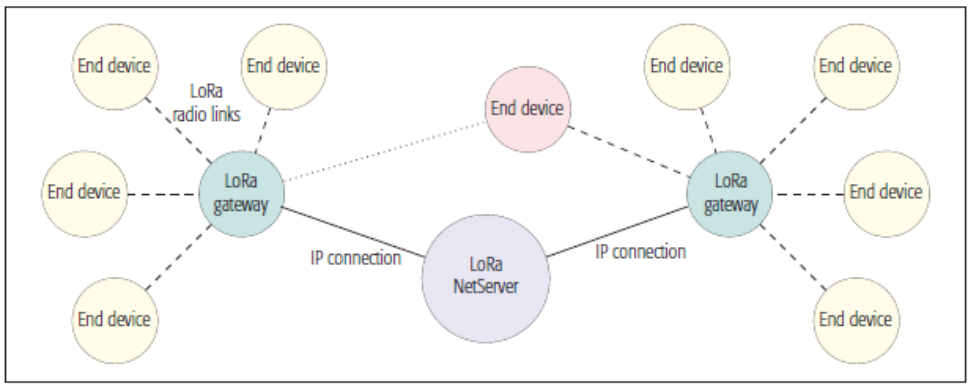


Figura 8 – Arquitetura LoRaWAN para medição inteligente. Fonte: (ZORZI, 2016)

Esta tecnologia é especialmente eficaz em áreas rurais e suburbanas, onde a cobertura de redes celulares é limitada. LoRa/LoRaWAN proporciona vantagens como cobertura ampla com poucos gateways e longa duração da bateria, tornando-se uma solução promissora para medição inteligente eficiente e de baixo custo.

2.4.3 Medição com Visão Computacional

Para medidores analógicos tradicionais, uma solução eficiente é a utilização de visão computacional para ler automaticamente os valores exibidos. Isso requer o desenvolvimento de modelos de *machine learning* capazes de realizar reconhecimento de dígitos em condições variadas de iluminação e posicionamento. (ZOU et al., 2021b) propõem um método robusto para reconhecimento de leituras de medidores em cenários complexos, utilizando técnicas de data augmentation para melhorar a performance do modelo.

2.4.4 Aplicações de Data Augmentation

A aplicação de técnicas de data augmentation é essencial para melhorar a robustez dos modelos de *machine learning* utilizados em smart metering. Técnicas como rotação, adição de ruído, ajuste de brilho e contraste, e deformações elásticas são utilizadas para aumentar a diversidade do conjunto de dados de treinamento, como descrito por (GULSHAD, 2021) e (ZHANG, 2019).A Tabela 1 traz um resumo da aplicação de técnicas clássicas de DA segundo os artigos citados anteriormente.

Tabela 1 – Algumas Técnicas de Data Augmentation e seus Efeitos na Precisão de Reconhecimento

Técnica	**Descrição**	**Efeito na Precisão**
Rotação (*Rotation*)	Alteração do ângulo de visão	Aumenta a acurácia contra diferentes ângulos
Ruído (*Noise*)	Adição de ruído aleatório à imagem	Melhora a resistência a interferências
Brilho/Contraste (*Brightness/Contrast*)	Ajustes no brilho e contraste	Melhora a adaptação a diversas condições de iluminação
Deformação Elástica (*Elastic Deformation*)	Aplicação de deformações aleatórias	Aumenta a variabilidade do conjunto de dados

3 Estado da Arte

3.1 Reconhecimento de Dígitos com Redes Neurais Convolucionais

O reconhecimento de dígitos utilizando Redes Neurais Convolucionais (CNNs) tem sido amplamente explorado em diversas aplicações, desde sistemas bancários até dispositivos IoT. Este capítulo revisa os principais modelos de CNNs aplicados ao reconhecimento de dígitos, focando em técnicas que otimizam desempenho e eficiência, fundamentais para aplicações em dispositivos IoT.

3.1.1 Modelos Principais de CNNs

Conforme vimos no Capítulo 2 da fundamentação teórica, modelos de aprendizagem de máquina já foram amplamente pesquisados e existem modelos citados acima como Lenet-5, MobileNet, SqueezeNet e TinyML, que foram desenvolvidos levando em consideração ambientes com restrição de capacidade de processamento e memória.

3.1.2 Técnicas de Otimização e Desempenho

Convoluções Separáveis em Profundidade - A técnica de convoluções separáveis em profundidade, utilizada no MobileNet, separa a operação de convolução em duas etapas: convolução espacial e convolução ponto a ponto. Isso reduz drasticamente o custo computacional sem sacrificar significativamente a precisão do modelo (ADAM, 2017).

Compressão e Quantização - SqueezeNet e TinyML empregam técnicas de compressão e quantização para reduzir o número de parâmetros e operações necessárias. Estas técnicas são cruciais para a implementação em dispositivos IoT, onde a memória e o poder de processamento são limitados (KEUTZER, 2016; RAY, 2022).

Data Augmentation - Aumentar os dados de treinamento através de técnicas de data augmentation, como rotação, escala e distorção, é uma prática comum para melhorar a robustez dos modelos de CNN. Isso é especialmente importante em cenários de IoT, onde os dados podem ser escassos e variados (AWAN, 2023).

3.2 Data Augmentation para Visão Computacional

A aplicação de *Data Augmentation* é essencial para melhorar a robustez e a capacidade de generalização dos modelos de visão computacional, especialmente quando se trata de conjuntos de dados limitados. Aumentando artificialmente o tamanho do conjunto

Tabela 2 – Comparação de técnicas de *Data Augmentation*

Técnica	**Categoria**	**Precisão Original (%)**	**Melhoria (%)**	**Ref.**
Espelhamento	Geométrica	90.00	+2.10	(NANNI et al., 2021)
Rotação	Geométrica	90.00	+2.70	(NITSCHKE, 2018)
Redimensionamento	Geométrica	90.00	+2.50	(NANNI et al., 2021)
Recorte	Geométrica	90.00	+3.20	(NITSCHKE, 2018)
Color Jittering	Fotométrica	90.00	+1.80	(KHOSHGOFTAAR, 2019)
PCA Jittering	Fotométrica	90.00	+2.00	(KHOSHGOFTAAR, 2019)

de dados de treinamento com transformações que preservam os rótulos, é possível reduzir o *overfitting* e melhorar o desempenho dos modelos de *Deep Learning* (NITSCHKE, 2018; KHOSHGOFTAAR, 2019; NANNI et al., 2021).

3.2.1 Técnicas de Data Augmentation

As técnicas de *Data Augmentation* podem ser categorizados em transformações geométricas e fotométricas (NITSCHKE, 2018). As transformações geométricas alteram a geometria da imagem, enquanto as transformações fotométricas modificam os canais de cor:

Transformações Geométricas - As transformações geométricas incluem operações como espelhamento, rotação, redimensionamento e recorte. Essas técnicas são amplamente utilizadas devido à sua eficiência computacional e simplicidade de implementação. A rotação, por exemplo, pode ser aplicada girando a imagem em torno do seu centro em ângulos específicos, gerando novas amostras invariantes (NANNI et al., 2021).

Transformações Fotométricas - As transformações fotométricas ajustam os canais de cor da imagem, alterando atributos como brilho, contraste e saturação. Um exemplo é a técnica de *Color Jittering*, que aplica manipulações aleatórias aos canais de cor, tornando o modelo mais robusto a variações de iluminação (KHOSHGOFTAAR, 2019).

A Tabela 2 traz um resumo da melhoria de acurácia observada utilizando diferentes técnicas de DA.

3.2.2 Estudos de Caso

Os artigos analisados demonstram que a aplicação de *Data Augmentation* pode levar a melhorias significativas no desempenho de modelos de CNN em várias tarefas de visão computacional. (NITSCHKE, 2018) mostraram que a técnica de recorte aumenta a precisão top-1 em 13,82% . (NANNI et al., 2021) destacaram a eficácia das transformações geométricas e fotométricas em vários conjuntos de dados, incluindo dados biomédicos e de textura.

Em resumo, a *Data Augmentation* é uma técnica crucial para melhorar o desempe-

nho e a robustez dos modelos de visão computacional, permitindo a criação de sistemas mais eficazes e generalizáveis.

3.3 Reconhecimento de Dígitos em Medidores: Trabalhos Relacionados e Data Augmentation

O reconhecimento de dígitos em sistemas de medição automática (AMR - Automatic Meter Reading) é uma tarefa desafiadora que se beneficia significativamente das técnicas de *data augmentation* para melhorar a precisão e a robustez dos modelos de CNN aplicados. Estudos recentes destacam várias abordagens eficazes para abordar essa questão, explorando técnicas de *data augmentation* específicas e modelos de CNN ajustados para essa finalidade.

Contextualização dos trabalhos de AMR. Antes de focar nos modelos de CNN e nas técnicas de *data augmentation*, é relevante contextualizar brevemente os avanços em leitura automática de medidores (AMR). Laroca et al. (LAROCA et al., 2021) propõem um sistema AMR completo para cenários não controlados, inserindo uma etapa de detecção dos cantos e classificação do contador para retificar a região e filtrar medidores ilegíveis. Eles disponibilizaram o conjunto Copel-AMR com 12 500 imagens de medidores reais (incluindo casos ilegíveis) e demonstraram que o sistema alcança taxas de reconhecimento superiores a 99 %, processando cerca de 55 quadros por segundo. Em um trabalho anterior, Laroca et al. (LAROCA et al., 2019) adotaram uma abordagem de duas etapas: utilizaram o detector Fast-YOLO para localizar o visor e avaliaram diferentes redes para reconhecer os dígitos; além disso, introduziram o conjunto público UFPR-AMR com 2 000 imagens e mostraram que, combinando o Fast-YOLO com a rede CR-NET e técnicas de aumento de dados, é possível alcançar 97,3 % de acerto em um conjunto de avaliação, mantendo alta velocidade de processamento. Esses trabalhos ilustram a evolução das soluções de AMR e justificam a análise de modelos compactos e estratégias de *data augmentation* abordadas a seguir.

3.3.1 Modelos de CNN

Os modelos de CNN são amplamente utilizados para o reconhecimento de dígitos devido à sua capacidade de aprender características discriminativas diretamente das imagens. A seguir, destacamos os modelos mais relevantes e suas adaptações para a leitura de medidores:

- **LeNet-5**: Desenvolvido por (HAFFNER, 1998), este modelo clássico de CNN é eficiente para tarefas de classificação de imagens, incluindo o reconhecimento de dígitos. Sua arquitetura simples e eficiente o torna uma escolha viável para dispositivos embarcados com recursos limitados.

- **MobileNet**: Este modelo, proposto por (ADAM, 2017), é otimizado para dispositivos móveis e embarcados, oferecendo um bom equilíbrio entre precisão e eficiência computacional. A utilização de convoluções separáveis em profundidade reduz significativamente o número de parâmetros e o custo computacional.

- **SqueezeNet**: (KEUTZER, 2016) introduziram este modelo que atinge precisão comparável ao AlexNet, mas com 50 vezes menos parâmetros. Sua arquitetura compacta é ideal para implementação em dispositivos com capacidade de processamento e memória restritas.

- **TinyML:** O modelo TinyML (RAY, 2022) se destaca no banco de dados MNIST, oferecendo uma precisão comparável a modelos mais robustos, enquanto requer recursos computacionais mínimos. Isso destaca a adequação do TinyML para aplicações de *machine learning* com restrições de recursos.

Trabalhos em ambientes controlados. Além das abordagens robustas em cenários não controlados discutidas anteriormente, a literatura também apresenta soluções em ambientes mais controlados. Gao et al. (GAO et al., 2018) propõem um sistema para reconhecimento de dígitos de hidrômetros em dispositivos móveis: a detecção dos dígitos é feita com um classificador do tipo AdaBoost e a subsequente classificação utiliza uma rede de baixo custo computacional, adequada para hardware portátil. Já Gómez, Rusiñol e Karatzas (GóMEZ; RUSIñOL; KARATZAS, 2018) apresentam uma abordagem de leitura de texto em cenas sem segmentação, aplicada a medidores utilitários, em que a sequência de dígitos é reconhecida diretamente pelo modelo sem separar cada caractere individualmente. Embora esses trabalhos foquem em cenários controlados ou com menos restrições de hardware, eles ilustram a evolução das técnicas de visão computacional para AMR e fornecem um contexto adicional para a escolha das estratégias adotadas nesta dissertação.

Mais recentemente, Xie et al. (XIE et al., 2025) apresentaram o algoritmo ASS-YOLO para detecção de dígitos em imagens de medidores digitais, juntamente com o conjunto de dados Dig21000, que contém cerca de 21 000 imagens de medidores. Embora o foco desse trabalho seja em medidores digitais e a abordagem utilize redes profundas mais pesadas, ele evidencia a tendência atual de construção de grandes bases públicas e o avanço contínuo das técnicas de detecção de dígitos. Os resultados obtidos reforçam a importância de considerar tanto a robustez quanto a eficiência computacional nas soluções de AMR desenvolvidas para dispositivos embarcados.

3.4 Aplicação de Data Augmentation em Sistemas AIoT

A integração de técnicas de *Data Augmentation* em sistemas de AIoT é crucial para melhorar a robustez e a precisão dos modelos de *machine learning* implantados em dispositivos embarcados. Esses sistemas são particularmente relevantes em contextos como medição inteligente (*smart metering*), onde a precisão e a confiabilidade dos dados são cruciais. Este capítulo aborda os avanços no estado da arte relacionados à aplicação de *Data Augmentation* em AIoT, com base nos estudos mais recentes.

3.4.1 Integração de AIoT com Data Augmentation

A combinação de técnicas de *Data Augmentation* com AIoT permite o desenvolvimento de modelos de *machine learning* mais robustos e eficientes, capazes de lidar com a diversidade e a variabilidade dos dados capturados por dispositivos IoT. Em (ZHANG; TAO, 2020) os autores exploram como a AIoT pode beneficiar-se de técnicas avançadas de *Data Augmentation* para melhorar a performance de sistemas de reconhecimento de dígitos em medidores inteligentes. A aplicação de modelos de *machine learning* em dispositivos embarcados permite a análise em tempo real e a tomada de decisões autônomas.

As técnicas de *Data Augmentation*, como deformação elástica, desfoque e rotação, são adaptadas para serem executadas em dispositivos de baixa capacidade de recursos, proporcionando melhorias significativas na acurácia dos modelos. Em (ZHANG; TAO, 2020) os autores demonstram a eficiência dessas técnicas na melhoria da robustez dos modelos através de exemplos práticos em diferentes aplicações de AIoT.

3.4.2 Benefícios do Data Augmentation em AIoT

O estudo apresentado em (SS; Z; IA, 2023) destaca a importância do *Data Augmentation* em AIoT, apontando que a diversidade gerada no conjunto de dados de treinamento pode compensar a falta de dados reais e melhorar a generalização dos modelos. Eles mostram como técnicas como a rotação e a escala podem ser aplicadas para aumentar a variabilidade dos dados de entrada, resultando em modelos mais robustos e precisos.

Em (MOHAMMADI et al., 2018) os autores também enfatizam que a aplicação de *Data Augmentation* em sistemas AIoT pode reduzir significativamente os erros de classificação e aumentar a eficiência do sistema. Eles demonstram que, ao aplicar técnicas de *Data Augmentation*, é possível melhorar a detecção de anomalias e a manutenção preditiva em sistemas IoT industriais, proporcionando uma maior confiabilidade e desempenho dos sistemas.

A Tabela 3 ilustra a comparação entre diferentes técnicas de Data Augmentation aplicadas ao reconhecimento de dígitos, conforme discutido por (AWAN, 2023). Os re-

sultados apresentados por (AWAN, 2023) demonstram que a aplicação de técnicas de Data Augmentation pode significativamente melhorar a precisão dos modelos de reconhecimento de dígitos em cenários AIoT. A integração de métodos como EfficientNet-B4 e EfficientDet-D4, que são otimizados para execução em dispositivos de recursos limitados, proporciona uma solução robusta e eficiente para o reconhecimento de dígitos em ambientes com variabilidade de dados.

Tabela 3 – Comparação entre técnicas de DA no reconhecimento de dígitos manuscritos

Técnica de Data Augmentation	**Precisão**
Deformações Elásticas	99.83%
Desfoque	99.10%
Rotação	99.25%
Ruído	98.75%
Variações de Intensidade	98.50%

A aplicação de Data Augmentation em sistemas AIoT oferece um avanço significativo na criação de modelos de reconhecimento de dígitos mais precisos e robustos. As técnicas discutidas neste capítulo são essenciais para lidar com a diversidade e a variabilidade dos dados capturados por dispositivos IoT, melhorando a performance geral dos sistemas AIoT.

3.4.3 Aplicações em *Smart Metering*

A aplicação de técnicas de *data augmentation* e modelos de CNN em *smart metering* tem mostrado resultados promissores. Em (SRIPANUSKUL; BUAYAI; MAO, 2022) os autores destacam o uso de *data augmentation* generativa para melhorar a precisão na leitura de medidores, demonstrando um aumento significativo na robustez dos modelos treinados. Em (HIDAYAT et al., 2021) e (HOLZINGER, 2019) também são apresentadas abordagens complementares que podem ser adaptadas para melhorar a precisão e a eficiência dos sistemas de AMR.

Estudos como os de (NITSCHKE, 2018) fornecem uma análise detalhada dos impactos de diversas técnicas de Data Augmentation em diferentes modelos de Machine Learning. Comparações entre métodos mostram que a combinação de várias técnicas pode melhorar a melhoria da performance de classificação durante o treinamento de um modelo. A Tabela 4 traz um compilado do resultado apontado por (NITSCHKE, 2018) onde é possível comprovar a melhoria da eficiência com o uso de técnicas de *Data Augmentation*

Tabela 4 – Comparação de Técnicas de Data Augmentation

Técnica de Data Augmentation	**Top-1 Acc (%)**	**Top-5 Acc (%)**	**Ganho (Top-1)**	**Ganho (Top-5)**
Baseline (Sem Augmentation)	48.13 ± 0.42	64.50 ± 0.65	-	-
Flipping (Espelhamento)	49.73 ± 1.13	67.36 ± 1.38	+1.60	+2.86
Rotating (Rotação)	50.80 ± 0.63	69.41 ± 0.48	+2.67	+4.91
Cropping (Recorte)	61.95 ± 1.01	79.10 ± 0.80	+13.82	+14.60
Color Jittering (Variação de Cor)	49.57 ± 0.53	67.18 ± 0.42	+1.44	+2.68
Edge Enhancement (Realce de Bordas)	49.29 ± 1.16	66.49 ± 0.84	+1.16	+1.99
Fancy PCA	49.41 ± 0.84	67.54 ± 1.01	+1.28	+3.04

4 Otimização de Reconhecimento de Dígitos em Dispositivos IoT

4.1 Introdução

A proposta deste trabalho de dissertação de mestrado visa desenvolver e validar estratégias otimizadas de reconhecimento de dígitos em sistemas embarcados, com foco em aplicações de *smart metering*. Dada a importância crescente dos sistemas de medição inteligente para a gestão eficiente de recursos em cidades inteligentes, este estudo busca não apenas melhorar a precisão e a robustez dos modelos de machine learning aplicados, mas também garantir que esses modelos operem eficientemente dentro das limitações impostas pelo hardware dos dispositivos embarcados.

Para alcançar esses objetivos, propomos uma metodologia que abrange desde a construção e preparação de datasets específicos até a implementação e avaliação de técnicas avançadas de *Data Augmentation*. O estudo também inclui a otimização de modelos de *machine learning* para que sejam compatíveis com as restrições de processamento e energia dos dispositivos baseados em microcontroladores ESP-32, utilizados nos medidores inteligentes.

Este capítulo detalha a proposta de trabalho, dividida em várias etapas que incluem a coleta de dados, a validação de modelos de *machine learning*, a aplicação de técnicas de *Data Augmentation*, e a avaliação dos resultados. O objetivo final é estabelecer uma metodologia que permita a generalização dos métodos desenvolvidos para outras aplicações em sistemas AIoT, promovendo avanços significativos na área de reconhecimento de padrões em dispositivos com recursos limitados.

As seções a seguir apresentam uma descrição detalhada do problema, as etapas propostas para o desenvolvimento do trabalho, e os estudos e resultados preliminares que fundamentam esta pesquisa.

4.2 Descrição do Problema

A crescente urbanização e a necessidade de uma gestão eficiente dos recursos naturais têm impulsionado a adoção de tecnologias de cidades inteligentes. Dentro deste contexto, os sistemas de medição inteligente (smart metering) emergem como uma solução essencial para monitorar e gerenciar o consumo de água, gás e eletricidade. Esses sistemas permitem a coleta automática e contínua de dados de consumo, facilitando a detecção de anomalias, a otimização do uso de recursos e a redução de desperdícios. No entanto, a

implementação eficaz de smart metering, especialmente em medidores analógicos, enfrenta diversos desafios técnicos e operacionais.

4.2.1 Desafios Técnicos

Um dos principais desafios é a **limitação de energia e processamento**. Os dispositivos de medição frequentemente operam com baterias e possuem capacidade de processamento limitada, o que exige que os modelos de *machine learning* sejam extremamente eficientes em termos de consumo de energia e uso de recursos computacionais. Isso é crucial para garantir a longevidade dos dispositivos e a viabilidade econômica do sistema.

Outro desafio significativo é a **variabilidade dos dados** coletados. As imagens capturadas dos medidores analógicos podem variar amplamente devido a diferentes condições de iluminação, ângulos de visualização e desgaste dos medidores. Essa variabilidade dificulta a criação de modelos robustos que possam generalizar bem para diferentes cenários, tornando essencial o uso de técnicas avançadas de tratamento de dados, como o *data augmentation*, para aumentar a diversidade do conjunto de dados de treinamento e melhorar a robustez dos modelos.

A **necessidade de dados de treinamento** também representa um obstáculo considerável. A obtenção de grandes volumes de dados rotulados para treinamento é uma tarefa complexa e demorada. Além disso, os dados disponíveis podem não abranger toda a gama de variações presentes no ambiente real, o que pode limitar a eficácia dos modelos de *machine learning*. Técnicas como o *data augmentation* são essenciais para simular essas variações e criar um conjunto de dados de treinamento mais representativo.

Por fim, a **precisão na identificação de dígitos** é crucial para o sucesso dos sistemas de smart metering. Erros na leitura dos dígitos dos medidores podem resultar em medições incorretas, afetando a gestão de recursos e a faturação. Portanto, é fundamental desenvolver modelos de *machine learning* que sejam não apenas precisos, mas também robustos o suficiente para lidar com a variabilidade dos dados e as limitações dos dispositivos.

4.2.2 Desafio Social

A presente dissertação é parte de um trabalho conjunto de desenvolvimento de um dispositivo de leitura de relógios de utilidades (água, gás, energia, etc.). Este dispositivo é composto de um hardware específico que se encaixa na tampa do relógio, possuindo uma câmera que tira algumas fotos ao longo do dia, realiza o reconhecimento dos dígitos no próprio processador (baseado em ESP-32) e envia esta informação em um pequeno pacote LoRaWAN para uma central. Assim, espera-se a transformação de relógios normais em relógios inteligentes apenas com a troca da tampa, sem a necessidade de maiores ajustes

Figura 9 – Medidor 1 Fonte: O Autor

Figura 10 – Medidor 2 Fonte: O Autor

ou conexão de energia elétrica. Este dispositivo será alimentado por uma bateria com expectativa de uso para aproximadamente 2,5 anos. Essa configuração proposta permite uma grande economia de recursos ambientais uma vez que não serão necessárias produção nem tampouco troca de infraestrutura já existente.

As Figuras 9, 10 e 11 trazem fotos reais do dispositivo que se demonstra como uma tampa acoplada a um medidor de água comum. Este dispositivo possui uma camêra direcionada aos dígitos de leitura. A câmera está configurada para tirar fotos periódicas que são processadas pela rede CNN rodando em um circuito interno baseado em ESP32-S3.

O presente trabalho tem por finalidade explorar estratégias de otimização do algoritmo de detecção de dígitos com alta acurácia (acima de 98%) sem comprometer o uso dos recursos físicos disponíveis. Essa exploração irá se concentrar em alguns aspectos principais (Figura 12):

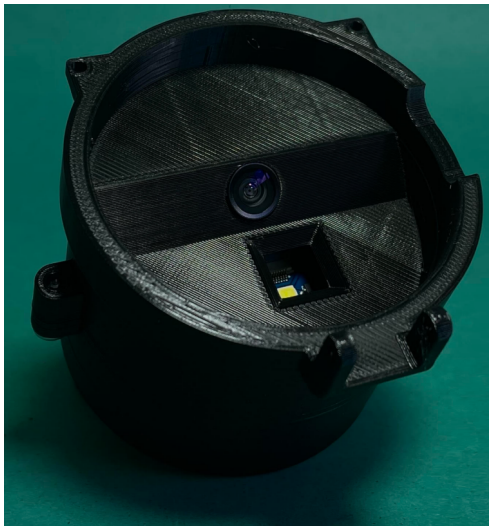


Figura 11 – Medidor 3 Fonte: O Autor

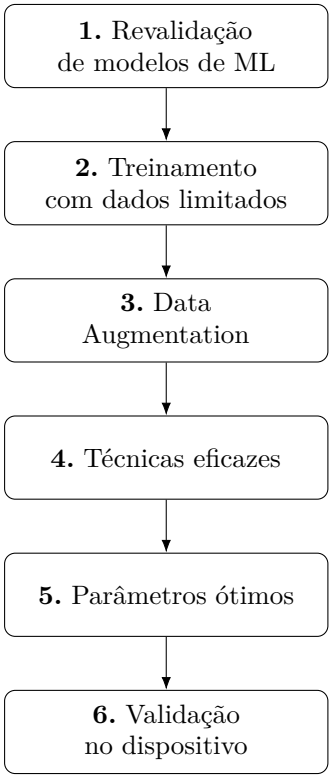


Figura 12 – Fluxo de melhorias para reconhecimento de dígitos em hidrômetros.

1. **Revalidação dos modelos de ML**: otimizar modelos existentes para a detecção de dígitos, adequando-os às características dos relógios medidores.
2. **Treinamento com datasets limitados**: desenvolver métodos robustos quando a coleta de imagens é restrita.
3. **Uso extensivo de Data Augmentation**: aplicar agressivamente técnicas de aumento de dados para elevar a acurácia.
4. **Detecção das técnicas mais efetivas**: identificar quais formas de augmentation

realmente trazem ganho.

5. **Detecção programática de parâmetros ótimos**: automatizar a escolha dos melhores hiperparâmetros de augmentation.
6. **Validação no dispositivo**: confirmar que a acurácia vista em laboratório se mantém em campo.

5 Análise Comparativa de Modelos de *Machine Learning* em Medidores de Água Inteligentes

No contexto de sistemas de medição inteligente — especialmente voltados ao monitoramento de bens de consumo — a aplicação de técnicas de visão computacional associadas ao *machine learning* permite automatizar a leitura de mostradores analógicos, extraindo dígitos por meio de reconhecimento de imagens. Este trabalho explora a adaptação e execução de modelos de *machine learning* em microcontroladores ESP32-S3, avaliando seu desempenho em tarefas de reconhecimento de dígitos com base na base de dados MNIST, tradicionalmente utilizada em experimentos de classificação de dígitos manuscritos (conforme citado em (HAFFNER, 1998),(SAINI; TRIPATHI; MIRZA, 2023),(MAJUMDER et al., 2018),(WONG et al., 2016),(ASISH et al., 2023),(SUN; FARD; MAHOOR, 2023)).

Inicialmente, os modelos são desenvolvidos e treinados em ambientes de alto desempenho com suporte a GPU, utilizando bibliotecas como Keras e TensorFlow (INC,). Posteriormente, são convertidos para o formato TensorFlow Lite e implantados em dispositivos com restrições de memória e processamento. O processo envolve a avaliação da precisão dos modelos, tempo de inferência e eventuais perdas decorrentes da conversão ou das limitações de hardware.

Este capítulo apresenta os fundamentos da pesquisa aplicada a sistemas de medição inteligente com reconhecimento de dígitos, com o objetivo de avaliar a viabilidade de sua implementação embarcada. A investigação foi orientada por perguntas centrais relacionadas à eficácia da execução de inferência em dispositivos de baixo custo, os desafios da conversão de modelos, e o impacto dessa transição na acurácia final.

As principais questões investigadas incluem:

- É viável executar inferências de *machine learning* em dispositivos com recursos restritos, utilizando metodologias semelhantes às aplicadas em estações de trabalho?
- Quais desafios surgem ao portar modelos desenvolvidos para workstations para microcontroladores como o ESP32-S3?
- Há perda de precisão durante o processo de conversão dos modelos para o formato TensorFlow Lite?

- A acurácia dos modelos se mantém ao serem executados diretamente nos dispositivos de borda?
- Quais são os tamanhos finais dos modelos após a conversão, e eles são compatíveis com os recursos de memória disponíveis?
- Existem modelos cuja conversão se torna inviável para uso embarcado?
- Caso desafios sejam identificados, quais estratégias podem ser adotadas para mitigar perdas de desempenho?

A partir dessas questões, este trabalho busca compreender os limites e as possibilidades da execução embarcada de modelos de aprendizado, oferecendo contribuições práticas para o desenvolvimento de soluções AIoT aplicadas à medição inteligente de água.

5.1 Tecnologias Selecionadas

Para alcançar os objetivos propostos neste trabalho, foi construído um conjunto de dados contendo imagens reais dos mostradores de medidores de água utilizados no sistema em desenvolvimento. Essas imagens apresentam uma grande variedade de cores, fundos, formatos e resoluções, exigindo técnicas robustas de reconhecimento de padrões para que os dígitos sejam corretamente identificados em condições reais de operação.

Dentre as abordagens possíveis, optou-se pela utilização de Redes Neurais Convolucionais (CNNs), dada sua consagrada aplicação em tarefas de reconhecimento de dígitos e imagens. Como referência metodológica inicial, utilizou-se a base de dados MNIST, composta por imagens em escala de cinza de 28x28 pixels contendo dígitos manuscritos de 0 a 9. Essa base é amplamente reconhecida na literatura e permite o desenvolvimento de modelos com resultados comparáveis entre diferentes arquiteturas.

A plataforma de hardware selecionada foi o microcontrolador ESP32-S3, mais especificamente os kits de desenvolvimento N8R2 e N16R8. Essa foi a escolha feita pela empresa parceira nesse trabalho por representar uma boa relação de custo e benefício com suporte a bibliotecas de *machine learning* embarcado e recursos adequados para inferência local, ainda que com restrições significativas de memória e processamento. Para viabilizar a execução dos modelos nesses dispositivos, foi utilizada a biblioteca TensorFlow Lite Micro, que permite a conversão e execução de redes neurais treinadas em ambientes de alto desempenho para dispositivos com capacidade computacional reduzida.

Esse conjunto tecnológico — modelos baseados em CNNs, base MNIST, TensorFlow e microcontroladores ESP32-S3 — foi escolhido por representar uma abordagem robusta, reprodutível e alinhada com os requisitos reais do projeto, especialmente quanto às

limitações de recursos e à necessidade de operação em ambientes sem conectividade permanente.

5.2 Modelos Selecionados

Na etapa de seleção de modelos, foram priorizadas arquiteturas que já apresentavam bons resultados na literatura com a base MNIST, além de atenderem a critérios de simplicidade estrutural, tamanho reduzido e compatibilidade com dispositivos embarcados de baixa capacidade. Após uma análise comparativa, foram selecionados os seguintes modelos para os experimentos:

- **LeNet5**: Modelo clássico de rede convolucional, amplamente utilizado como referência em tarefas de classificação de dígitos.
- **TinyML**: Modelo leve oriundo da comunidade TinyML, otimizado para execução em dispositivos com restrições severas de memória.
- **MobileNet**: Arquitetura conhecida pela eficiência em dispositivos móveis, com bom compromisso entre acurácia e tamanho do modelo.
- **MiniVGG**: Versão reduzida da VGGNet, buscando manter alto desempenho com estrutura simplificada.
- **AlexNet**: Modelo histórico, introduzido na ImageNet, utilizado aqui por seu desempenho e por fins de comparação.
- **VGG16 e VGG19**: Modelos profundos da família VGG, com elevado número de parâmetros, incluídos na análise para avaliar os limites de viabilidade de execução embarcada.
- **SqueezeNet**: Modelo otimizado para reduzir drasticamente o número de parâmetros mantendo boa acurácia.
- **CapsNet**: Arquitetura baseada em redes de cápsulas, com proposta inovadora de representação espacial de características.

Esses modelos representam uma diversidade de abordagens arquiteturais e níveis de complexidade, permitindo avaliar não apenas a acurácia, mas também o impacto do tamanho do modelo, do tempo de inferência e da viabilidade de implantação em dispositivos de borda como o ESP32-S3.

5.3 Metodologia

Esta etapa do trabalho consistiu na preparação, conversão e avaliação de diferentes modelos de *machine learning* com base em redes neurais convolucionais. O objetivo foi analisar sua viabilidade de execução em dispositivos embarcados de baixo custo, mais especificamente no microcontrolador ESP32-S3. Para isso, foi seguido um conjunto estruturado de procedimentos aplicados a cada modelo selecionado.

A seguir, são descritas as etapas executadas para cada um dos modelos escolhidos (Figura 13):

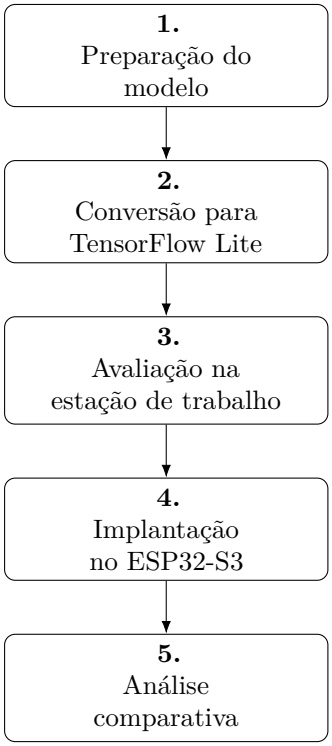


Figura 13 – Fluxo de implantação e validação do modelo de reconhecimento de dígitos.

1. **Preparação do modelo**: carregamento ou treinamento do modelo em uma estação de trabalho com GPU, utilizando Python + TensorFlow.
2. **Conversão**: transformação do modelo treinado para o formato *TensorFlow Lite*, compatível com dispositivos embarcados.
3. **Avaliação na estação de trabalho**: execução da inferência com o modelo convertido sobre 10 000 imagens separadas da base MNIST, seguida do cálculo da acurácia e da matriz de confusão.
4. **Implantação no ESP32-S3**: carregamento do modelo convertido em um kit de desenvolvimento ESP32-S3, com posterior inferência sobre o mesmo conjunto de 10 000 imagens, utilizando C + TensorFlow Lite Micro.

5. **Análise comparativa**: cálculo da acurácia e da matriz de confusão no ambiente embarcado e comparação aos resultados obtidos na estação de trabalho.

O código Python utilizado para treinar e converter os modelos foi padronizado, com variações apenas na definição das camadas e hiperparâmetros específicos de cada arquitetura. A base MNIST foi dividida em 80% para treinamento e 20% para validação, mantendo o conjunto de teste isolado conforme padrão da base.

No lado embarcado, o código em linguagem C também foi unificado para todos os modelos, utilizando o mesmo conjunto de imagens e estrutura de execução, respeitando a ausência de pré-processamento adicional tanto na fase de treinamento quanto de teste. O uso da biblioteca *TensorFlow Lite Micro* (INC,) garantiu a compatibilidade entre o modelo convertido e o ambiente de execução no ESP32-S3, respeitando os limites de memória e armazenamento disponíveis no dispositivo.

Foram utilizados três kits de desenvolvimento ESP32-S3 para execução dos testes, com o intuito de verificar a consistência dos resultados e garantir maior confiabilidade aos dados obtidos. Todas as bibliotecas utilizadas foram empregadas em suas versões mais recentes disponíveis no momento do estudo, e o hardware escolhido representava os modelos mais atualizados no mercado voltados para aplicações com restrição de recursos computacionais.

Esse conjunto de tecnologias e procedimentos ofereceu um ambiente controlado para avaliação dos modelos, garantindo reprodutibilidade dos testes e permitindo a análise detalhada de desempenho, acurácia e viabilidade da implantação em dispositivos de borda.

5.3.1 Preparação dos Modelos na Estação de Trabalho

O processo de construção e treinamento dos modelos foi conduzido por meio de um script em Python disponível publicamente[1], utilizando a biblioteca Keras para definição da arquitetura, treinamento e avaliação das redes neurais convolucionais.

As etapas do script incluem:

- **Importação de bibliotecas**: carregamento de pacotes para manipulação de dados, visualização e *machine learning*.
- **Construção e compilação do modelo**: definição das camadas convolucionais por meio da arquitetura sequencial do Keras.
- **Carregamento e pré-processamento da base MNIST**: normalização das imagens, codificação dos rótulos e divisão do conjunto de dados em treinamento, validação e teste.

[1] <https://github.com/julianoribeiro/MLonEsp32.git>

- **Treinamento do modelo**: realização do treinamento utilizando 15 épocas e *batch size* de 512 amostras, conforme boas práticas descritas na documentação oficial do TensorFlow e experiência prévia dos desenvolvedores.
- **Avaliação do modelo**: teste de predição com o conjunto de teste, ainda na estação de trabalho.

O código desenvolvido mantém uma estrutura simples, porém robusta, seguindo as práticas recomendadas para classificação de imagens por meio de CNNs. Não foram aplicados processos adicionais de pré-processamento de imagem ou manipulação de dados antes do treinamento ou da inferência.

5.3.2 Conversão dos Modelos para TensorFlow Lite

A conversão dos modelos foi realizada com as funcionalidades nativas da biblioteca TensorFlow, utilizando os métodos recomendados na documentação oficial. O processo de conversão foi direto e eficiente, permitindo armazenar e executar os modelos em dispositivos ESP32-S3 com tempo reduzido de preparação.

Essa etapa foi fundamental para garantir que os modelos treinados em ambiente de alto desempenho pudessem ser utilizados em dispositivos com recursos limitados, respeitando restrições de memória e compatibilidade com bibliotecas de inferência embarcadas.

5.3.3 Testes com o ESP32-S3

A execução dos testes no ESP32-S3 foi realizada por meio da biblioteca TensorFlow Lite Micro, escrita em linguagem C. O código embarcado foi desenvolvido com base na estrutura sugerida pela documentação da própria biblioteca, mantendo simplicidade e clareza no processo de carregamento e inferência dos modelos.

Durante os testes, foram utilizadas as mesmas 10.000 imagens do conjunto de teste da base MNIST, garantindo comparabilidade com os resultados obtidos na estação de trabalho. Assim como na fase em Python, não foi aplicado nenhum tipo de pré-processamento de imagem adicional no ambiente embarcado.

Esse ambiente controlado de testes permitiu avaliar com precisão o impacto da conversão e da execução embarcada sobre a acurácia e o tempo de inferência.

5.4 Resultados e Discussão

Com base nos testes realizados tanto na estação de trabalho quanto nos dispositivos embarcados, foram obtidas diversas estatísticas que auxiliam na análise da viabilidade de execução de modelos de *machine learning* em ambientes com recursos computacionais

limitados. Esta seção apresenta os principais resultados obtidos e busca responder às questões de pesquisa apresentadas anteriormente, oferecendo uma visão abrangente dos desafios e oportunidades no uso de modelos embarcados para reconhecimento de dígitos em medidores inteligentes de água.

A Tabela 5 apresenta os tamanhos, em megabytes, dos modelos selecionados, considerando suas versões originais (em TensorFlow) e convertidas (em TensorFlow Lite). Essa análise foi fundamental para a seleção dos modelos que poderiam ser efetivamente executados nos kits de desenvolvimento ESP32-S3, uma vez que a memória RAM e o armazenamento interno desses dispositivos são bastante restritos.

Tabela 5 – Tamanho dos modelos em MB (TensorFlow e TensorFlow Lite)

Modelo	**TensorFlow**	**TensorFlow Lite**
LeNet5	1,08	0,30
TinyML	0,77	0,21
MobileNet	3,26	1,55
MiniVGG	19,71	6,40
SqueezeNet	9,32	2,80
VGG16*	57,36	56,42
VGG19*	229,79	76,42
AlexNet*	247,53	82,40
CapsNet*	37,63	12,50

* Modelos marcados excedem a memória disponível do ESP32.

Observa-se que os modelos não apresentam um padrão uniforme de compressão ao serem convertidos para o formato TensorFlow Lite. Embora esse não tenha sido um objetivo direto da pesquisa, verificou-se que o grau de compressão está fortemente associado à arquitetura interna de cada modelo, especialmente ao tipo e à quantidade de camadas utilizadas.

Essas variações indicam a importância de investigar, em estudos futuros, quais tipos de camadas influenciam diretamente a eficiência de compressão, possibilitando o desenvolvimento de estratégias de otimização mais eficazes.

Outro aspecto relevante refere-se às limitações físicas dos dispositivos embarcados. Os kits ESP32-S3 utilizados possuem, no máximo, 8MB de memória RAM adicional, sendo que a biblioteca TensorFlow Lite Micro exige que todo o modelo esteja carregado em memória. Por essa razão, os modelos VGG16, VGG19, AlexNet e CapsNet foram desconsiderados na fase de testes embarcados, devido ao tamanho final do modelo convertido ultrapassar a capacidade do dispositivo.

Nos testes conduzidos na estação de trabalho, foram realizados dois conjuntos de inferência para cada modelo: o primeiro utilizando o modelo treinado em TensorFlow, e o segundo com o modelo convertido para TensorFlow Lite. Ambos foram avaliados com

o mesmo conjunto de 10.000 imagens previamente separadas da base MNIST. Em todos os casos, as matrizes de confusão obtidas para os dois conjuntos de teste foram idênticas, indicando que a conversão para o formato Lite não impactou negativamente a acurácia dos modelos. Esses resultados apontam para três conclusões importantes:

- **Alta acurácia**: Todos os modelos testados apresentaram excelente desempenho na tarefa de classificação de dígitos.
- **Consistência entre versões**: A equivalência entre as matrizes de confusão indica que a versão convertida manteve o mesmo desempenho da versão original.
- **Ausência de perda significativa**: A conversão para o formato TensorFlow Lite não causou perda de precisão nos modelos testados.

A Tabela 6 resume os principais resultados dos modelos testados no ESP32-S3, incluindo o tempo total de inferência e a acurácia obtida com o conjunto de 10.000 imagens da base MNIST.

Tabela 6 – Resumo dos Resultados de Inferência no ESP32-S3

Modelo	**Tempo de Inferência (s)**	**Acurácia (%)**
LeNet5	8,35	98,7
TinyML	6,81	98,5
MobileNet	14,74	98,8
MiniVGG	25,10	98,6
SqueezeNet	21,95	98,5

Cabe destacar que, apesar de não terem sido testados no ESP32-S3 por questões de tamanho, os modelos VGG16, VGG19, AlexNet e CapsNet apresentaram acurácias acima de 98% quando executados na estação de trabalho. Isso demonstra que a limitação não está na qualidade dos modelos, mas sim na compatibilidade com os recursos físicos do hardware embarcado.

5.4.1 Análise do Modelo LeNet5

Dentre os modelos testados, o LeNet5 destacou-se por apresentar uma combinação bastante equilibrada entre precisão, tamanho e tempo de execução. Após a conversão para o formato TensorFlow Lite, o modelo foi implantado no ESP32-S3 e submetido ao mesmo conjunto de 10.000 imagens da base MNIST utilizado na estação de trabalho.

A Tabela 7 apresenta a matriz de confusão resultante da execução do modelo no ambiente embarcado.

Tabela 7 – Matriz de confusão – Modelo LeNet5 (Execução no ESP32-S3)

Dígito	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	974	0	0	0	0	0	0	0	4	2
1	0	1130	0	0	0	0	0	1	4	0
2	4	1	1013	2	1	0	0	2	9	0
3	1	0	1	1000	0	1	0	1	5	1
4	0	0	2	0	968	0	1	0	3	8
5	2	0	0	9	0	868	2	0	8	3
6	7	3	0	0	2	1	940	0	5	0
7	1	0	11	1	0	0	0	1006	4	5
8	3	0	1	3	1	1	2	1	960	2
9	3	3	0	2	6	1	0	2	5	987

A análise da matriz de confusão obtida no ESP32-S3 demonstra que o modelo LeNet5 manteve elevado nível de acurácia, mesmo em um ambiente com severas restrições de processamento. A maioria das previsões está concentrada na diagonal principal da matriz, o que indica classificações corretas para os respectivos dígitos.

Foram observadas variações sutis nos erros de classificação entre o ambiente embarcado e a estação de trabalho. Por exemplo, o dígito “6” foi confundido com “0” em sete ocasiões no ESP32-S3, enquanto no teste realizado na estação de trabalho essa confusão ocorreu seis vezes. Esses desvios, ainda que pequenos, revelam que o hardware pode introduzir pequenas flutuações nos resultados da classificação.

Adicionalmente, o dígito “5” apresentou desempenho ligeiramente inferior no dispositivo embarcado quando comparado à estação de trabalho, sugerindo que o ambiente de execução pode influenciar a estabilidade da inferência para determinados padrões. Por outro lado, o dígito “9” apresentou maior número de erros na estação de trabalho, enquanto no ESP32-S3 esses erros foram reduzidos.

Esses resultados reforçam a importância de realizar testes diretamente em dispositivos de borda, visto que a simples conversão do modelo pode não refletir completamente seu comportamento em execução real. Apesar das pequenas variações, o desempenho geral do LeNet5 no ESP32-S3 foi considerado satisfatório e compatível com os objetivos do projeto.

5.4.2 Análise do Modelo TinyML

O modelo TinyML, oriundo da comunidade da plataforma Kaggle, foi selecionado por sua leveza e adequação a dispositivos embarcados com recursos extremamente limitados. O desempenho do modelo foi avaliado por meio de sua matriz de confusão após execução no microcontrolador ESP32-S3, como mostrado na Tabela 8.

Tabela 8 – Matriz de confusão – Modelo TinyML (Execução no ESP32-S3)

Dígito	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	974	0	0	2	0	1	2	0	1	0
1	0	1129	1	3	0	1	0	0	1	0
2	3	1	1013	5	1	0	0	4	5	0
3	0	0	1	998	0	4	0	2	2	3
4	0	0	0	0	968	0	1	1	1	11
5	2	0	1	7	0	879	1	1	0	1
6	6	2	0	0	4	6	939	0	1	0
7	0	1	3	3	0	1	0	1015	1	4
8	3	0	1	1	1	4	1	2	958	3
9	1	4	1	1	1	7	0	6	4	984

Os resultados obtidos demonstram que o modelo TinyML apresentou bom desempenho geral, com acurácia próxima a 98,5%. No entanto, observou-se certa dificuldade na diferenciação de dígitos visualmente semelhantes — por exemplo, erros entre os dígitos “8” e “9”, ou entre “3” e “5”.

Apesar dessas imprecisões pontuais, o modelo se mostrou uma opção bastante adequada para aplicações embarcadas, principalmente considerando sua leveza e baixo consumo de memória. Trata-se de uma solução viável para cenários em que o equilíbrio entre acurácia e limitação de recursos é um fator crítico.

5.4.3 Análise do Modelo MobileNet

A arquitetura MobileNet, amplamente utilizada em dispositivos móveis por sua eficiência e baixo consumo de recursos, foi incluída na avaliação para verificar sua performance em dispositivos ainda mais restritos, como o ESP32-S3.

A Tabela 9 apresenta a matriz de confusão resultante da execução do modelo convertido no ambiente embarcado.

Tabela 9 – Matriz de confusão – Modelo MobileNet (Execução no ESP32-S3)

Dígito	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	977	0	0	0	1	0	0	1	1	0
1	0	1132	0	2	0	0	1	0	0	0
2	3	5	1013	3	1	0	1	4	2	0
3	0	0	0	1002	0	3	0	2	2	1
4	0	0	1	0	971	0	4	0	0	6
5	1	0	0	3	0	885	1	1	0	1
6	7	2	0	0	1	11	937	0	0	0
7	1	12	5	3	0	0	0	1001	1	5
8	7	1	5	4	1	7	0	3	942	4
9	1	4	0	1	8	4	0	7	2	982

A análise mostra que o modelo MobileNet apresentou desempenho elevado no ESP32-S3, com destaque para os dígitos “0”, “1” e “3”, que obtiveram taxas de acerto superiores a 99%. Embora algumas confusões entre dígitos tenham sido observadas, como “7” e “1”, ou “4” e “9”, essas ocorrências foram relativamente pontuais.

Comparativamente, a execução do MobileNet no ESP32 apresentou maior estabilidade do que na estação de trabalho, com redução de erros em alguns dígitos específicos, como “9”. Isso pode estar relacionado ao comportamento do otimizador durante a execução embarcada ou a diferenças de precisão no cálculo das operações numéricas.

Entretanto, vale destacar que, apesar da boa acurácia, o tempo de execução da inferência no ESP32-S3 foi significativamente maior em comparação com modelos mais simples, o que pode limitar seu uso em aplicações que demandam resposta em tempo real. Ainda assim, trata-se de uma arquitetura promissora para aplicações embarcadas, desde que as restrições de tempo e consumo sejam aceitáveis.

5.4.4 Análise do Modelo MiniVGG

O modelo MiniVGG foi proposto como uma versão reduzida da arquitetura VGGNet, visando equilibrar profundidade e capacidade de generalização com um tamanho compatível com dispositivos embarcados. A Tabela 10 apresenta a matriz de confusão gerada a partir da execução do modelo no ESP32-S3.

Tabela 10 – Matriz de confusão – Modelo MiniVGG (Execução no ESP32-S3)

Dígito	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	975	0	0	0	0	0	2	1	1	1
1	0	1132	1	0	0	0	0	1	1	0
2	3	1	1009	1	1	0	1	4	8	0
3	0	0	1	996	0	4	0	2	5	2
4	1	0	1	0	968	0	4	1	1	8
5	1	0	1	3	0	882	2	0	1	0
6	3	2	1	0	1	2	945	0	4	0
7	1	2	3	1	0	0	0	1006	2	6
8	1	0	1	1	1	1	0	3	961	5
9	1	1	0	1	2	4	0	4	3	993

Os resultados mostram que o modelo MiniVGG manteve bom desempenho, com acurácia geral em torno de 98,6%. As classificações incorretas ocorreram em padrões visuais tipicamente ambíguos, como entre os dígitos “8” e “9”, ou entre “4” e “9”.

Apesar de seu tamanho superior ao LeNet5 e ao TinyML, o MiniVGG ainda pôde ser executado com sucesso no ESP32-S3. Seu desempenho justifica o uso de arquiteturas um pouco mais profundas, desde que otimizadas, especialmente em cenários em que a acurácia é mais relevante do que o tempo de resposta.

5.4.5 Análise do Modelo SqueezeNet

A arquitetura SqueezeNet foi projetada com o objetivo de reduzir drasticamente o número de parâmetros, mantendo uma acurácia comparável a modelos maiores. A Tabela 11 apresenta a matriz de confusão do modelo convertido e executado no microcontrolador ESP32-S3.

Tabela 11 – Matriz de confusão – Modelo SqueezeNet (Execução no ESP32-S3)

Dígito	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	973	0	0	0	1	0	3	1	1	1
1	0	1131	1	0	0	0	0	0	3	0
2	1	1	1006	2	2	0	0	4	10	1
3	0	0	1	991	0	6	0	3	6	3
4	1	0	1	0	964	0	3	0	1	10
5	1	0	2	5	0	879	2	0	1	1
6	3	2	1	0	2	4	944	0	2	0
7	0	1	3	0	0	1	0	1011	3	9
8	2	0	1	1	1	3	1	1	958	2
9	2	3	0	1	4	4	0	6	3	986

O desempenho do SqueezeNet foi consistente com o esperado, apresentando acurácia próxima a 98,5% e baixa taxa de erros em geral. A compactação do modelo não comprometeu significativamente sua capacidade de generalização, o que reforça a viabilidade de uso dessa arquitetura em sistemas embarcados com restrições de espaço e memória.

Embora o tempo de inferência tenha sido superior ao de modelos menores como o TinyML, os ganhos em acurácia tornam o SqueezeNet uma alternativa interessante em aplicações onde o tempo de resposta pode ser levemente flexibilizado.

Em síntese, tanto o MiniVGG quanto o SqueezeNet se mostraram modelos robustos e adequados para uso em sistemas embarcados, representando um bom compromisso entre precisão e eficiência computacional.

5.4.6 Síntese dos Resultados e Outras Considerações

Os testes realizados revelaram que é possível executar modelos de *machine learning* em dispositivos com recursos restritos como o ESP32-S3, mantendo níveis elevados de acurácia. No entanto, a viabilidade da execução embarcada depende diretamente do tamanho final do modelo após a conversão, bem como do consumo de memória e tempo de inferência.

Os modelos mais leves, como LeNet5 e TinyML, apresentaram tempo de execução reduzido, com médias inferiores a 10 segundos para a classificação de 10.000 imagens, o que equivale a aproximadamente 1 milissegundo por inferência. Já modelos maiores, como

MobileNet e MiniVGG, demandaram entre 10 e 25 segundos para processar o mesmo volume de dados. O modelo SqueezeNet, mesmo com sua proposta de compactação, apresentou tempo de execução mais elevado do que os modelos menores, mas ainda compatível com a operação embarcada.

Outro aspecto relevante diz respeito à variabilidade dos tempos de inferência observados mesmo dentro do mesmo dispositivo. Foram detectadas pequenas flutuações na ordem de centésimos de segundo entre execuções consecutivas, sugerindo a influência de fatores como alocação de memória, interferências do sistema operacional e características específicas do compilador. Embora essas variações não impactem significativamente os resultados, elas reforçam a importância de testes múltiplos para garantir consistência e confiabilidade nas medições.

Por fim, observou-se que o processo de conversão dos modelos para o formato TensorFlow Lite não provocou perda significativa de acurácia, o que valida a eficácia da conversão e da biblioteca utilizada. Em todos os casos, as matrizes de confusão do ambiente embarcado apresentaram comportamento semelhante às obtidas na estação de trabalho, com variações marginais que não comprometem a funcionalidade do sistema.

Estes resultados comprovam a viabilidade técnica de executar modelos de *machine learning* em dispositivos de borda, contribuindo para soluções AIoT mais autônomas e eficientes. O modelo LeNet5 destacou-se como o mais equilibrado entre desempenho, precisão e tempo de inferência, enquanto o TinyML apresentou a melhor eficiência em termos de velocidade e consumo de memória.

A escolha do modelo ideal dependerá, portanto, das prioridades da aplicação, sejam elas acurácia, tempo de resposta ou limitação de recursos. Esse estudo serve como base para decisões de projeto em sistemas de medição inteligente, especialmente em contextos onde a conectividade é limitada e a inferência precisa ocorrer localmente.

5.5 Conclusão do Capítulo

Este capítulo apresentou uma análise comparativa da execução de diferentes modelos de *machine learning* em dispositivos embarcados voltados à medição inteligente de água. O objetivo foi investigar a viabilidade de se utilizar redes neurais convolucionais embarcadas em microcontroladores como o ESP32-S3, mantendo a acurácia dos modelos e respeitando as limitações de hardware.

Os resultados obtidos demonstraram que modelos clássicos, como o LeNet5 e variações leves como o TinyML, apresentam desempenho satisfatório em termos de precisão e tempo de inferência, mesmo em dispositivos com recursos bastante restritos. Modelos mais complexos, como MobileNet, MiniVGG e SqueezeNet, também apresentaram acurácia

elevada, embora com tempos de inferência maiores. Já arquiteturas mais pesadas, como AlexNet, VGG16, VGG19 e CapsNet, mostraram-se inviáveis para execução no ESP32-S3, principalmente devido ao tamanho excessivo dos modelos após a conversão.

A conversão dos modelos para o formato TensorFlow Lite não implicou perdas significativas de acurácia, o que evidencia a robustez do processo e a viabilidade da biblioteca TensorFlow Lite Micro para aplicações embarcadas. As pequenas variações observadas nas matrizes de confusão entre os ambientes de teste indicam que, embora a precisão dos cálculos no hardware embarcado possa introduzir pequenas flutuações, o desempenho geral permanece estável e confiável.

Este estudo contribui com subsídios práticos para o desenvolvimento de soluções AIoT voltadas ao monitoramento de consumo de água em tempo real, reforçando a importância da escolha consciente de modelos conforme o contexto e as restrições do projeto. Além disso, os experimentos realizados apontam caminhos para futuras investigações, como a compressão de modelos mais complexos, o uso de técnicas de quantização e poda, e a otimização de tempo de inferência para cenários em que a resposta em tempo real é um requisito crítico.

A partir da experiência adquirida nesta avaliação, a etapa seguinte da pesquisa focará na coleta de dados reais com imagens dos mostradores dos hidrômetros utilizados no sistema em desenvolvimento, empregando técnicas de *data augmentation* e redes neurais personalizadas para lidar com as especificidades do ambiente real.

6 Aprimoramento do Reconhecimento de Dígitos por meio de DA em Dispositivos AIoT

A implementação de modelos de *machine learning* em dispositivos de borda no contexto da Internet Artificial das Coisas (AIoT) está transformando a forma como coletamos e processamos dados. Uma aplicação chave dessa abordagem é o reconhecimento automático de dígitos em hidrômetros, tarefa historicamente realizada de forma manual. Com a crescente proliferação de dispositivos inteligentes, cresce também a demanda por sistemas eficientes e autônomos, capazes de interpretar e analisar dados diretamente na borda, reduzindo a dependência de processamento remoto.

Este capítulo apresenta uma investigação abrangente sobre a aplicação de técnicas de *data augmentation* como forma de melhorar a acurácia de modelos de reconhecimento de dígitos executados em dispositivos de borda com recursos computacionais limitados. Diferentemente de estudos anteriores focados em ambientes com alto poder computacional, esta pesquisa aborda explicitamente os desafios impostos por implantações em infraestrutura urbana inteligente, onde o processamento local enfrenta severas restrições de memória e desempenho, e a coleta de dados em campo é uma tarefa desafiadora.

Motivados por essas limitações, experimentamos um conjunto variado de técnicas de aumento de dados, incluindo injeção de ruído, ajustes de brilho e contraste, além de transformações geométricas como translação, rotação e cisalhamento. O estudo avalia o impacto dessas técnicas sobre a acurácia do modelo, utilizando dois conjuntos de dados públicos e um conjunto proprietário composto por imagens reais de dígitos em hidrômetros.

A contribuição deste capítulo está na formulação de um protocolo padronizado para avaliação de parâmetros de *data augmentation*, oferecendo evidências empíricas e diretrizes para o desenvolvimento de aplicações de *machine learning* embarcado mais eficientes e precisas no contexto de cidades inteligentes.

6.1 Trabalhos Relacionados

O uso de *data augmentation* tem sido amplamente estudado como forma de mitigar a escassez de dados e melhorar a capacidade de generalização de modelos de visão computacional. Um levantamento abrangente sobre as abordagens existentes é apresentado em (MUMUNI; MUMUNI, 2022), que organiza as técnicas em quatro grupos:

- Transformações geométricas e fotométricas simples;

- Estratégias baseadas em síntese 3D;
- Métodos baseados em GANs;
- Abordagens de aprendizado de políticas de aumento.

Entre os principais desafios citados, destacam-se (i) a escolha apropriada de transformações para o domínio em questão e (ii) o custo computacional, especialmente quando múltiplos exemplos são gerados por imagem de entrada.

Na fronteira das abordagens automáticas, métodos como *AutoAugment* (CUBUK et al., 2019) e *RandAugment* (CUBUK et al., 2020) realizam busca no espaço de transformações para selecionar combinações otimizadas de técnicas e seus respectivos parâmetros. Apesar dos resultados impressionantes alcançados em bases de domínio geral, (YANG et al., 2023) destacam que esses métodos tendem a falhar quando aplicados a dados ruidosos ou enviesados — como é comum em sistemas do mundo real — além de requererem alto custo computacional.

Com relação a aplicações embarcadas, poucos estudos investigam o impacto de técnicas de *data augmentation* sobre modelos executados em dispositivos de borda. A maioria das análises concentra-se em ambientes com GPU, negligenciando fatores como latência, footprint de memória e estabilidade sob execução embarcada. O capítulo anterior desta dissertação já discutiu a viabilidade de execução de arquiteturas leves em microcontroladores ESP32-S3, em linha com os achados descritos no capítulo anterior, que demonstram que modelos como LeNet-5 e TinyML mantêm boa acurácia com footprint reduzido. No entanto, tais estudos assumem um conjunto de dados fixo, sem explorar intervenções sobre o espaço de entrada.

Este capítulo avança o estado da arte ao avaliar nove técnicas clássicas de *data augmentation*, testadas sob três intensidades distintas, mensurando seu impacto sobre a acurácia em três bases diferentes. Além disso, analisamos a influência do volume de imagens geradas (0 a 500 % do conjunto original), conectando os resultados à execução embarcada validada no capítulo anterior.

6.2 Contexto da Pesquisa e Objetivos

Este estudo está inserido no contexto do projeto *SmartEyes*, que visa desenvolver dispositivos de leitura visual inteligente para hidrômetros residenciais, com execução local de modelos de reconhecimento de dígitos em microcontroladores de baixo custo. Ao contrário das abordagens convencionais, que transmitem imagens para servidores remotos, a proposta do projeto é realizar a inferência diretamente no dispositivo embarcado — neste

caso, utilizando o ESP32-S3 — evitando dependência de rede, reduzindo o consumo de energia e viabilizando aplicações em larga escala.

No entanto, esse cenário apresenta dois desafios principais. Primeiro, os dados reais são escassos: embora o projeto conte com um banco de imagens próprio, a coleta é onerosa, dependente de testes em campo e sujeita a variações ambientais (iluminação, posicionamento, foco etc.). Segundo, os modelos precisam ser suficientemente pequenos e rápidos para execução local, respeitando os limites de memória e tempo de resposta.

O capítulo anterior desta dissertação demonstrou que arquiteturas leves como o *LeNet-5* são capazes de atingir alta acurácia ($> 97\,\%$) mesmo após quantização para 8 bits, mantendo um tamanho de modelo inferior a 25 KB. Contudo, os testes foram conduzidos com conjuntos de dados fixos, sem intervenção no espaço de entrada.

Neste capítulo, investigamos se é possível melhorar o desempenho desses modelos embarcados por meio da aplicação sistemática de técnicas clássicas de *data augmentation*, com três objetivos principais:

- Avaliar o impacto de nove técnicas tradicionais (ruído, brilho, contraste, translações, rotações, shear, blur gaussiano e deformação elástica) sobre a acurácia do modelo;
- Medir o efeito de diferentes intensidades de cada transformação, de forma parametrizada;
- Analisar o comportamento da acurácia em função do volume de imagens geradas (0–500 %), identificando o ponto de saturação.

Todos os experimentos utilizam o modelo *LeNet-5* como referência, garantindo compatibilidade com o pipeline embarcado descrito anteriormente. Três bases são utilizadas para validação cruzada:

- **Kaggle NumOCR:** dígitos impressos com ruído artificial, resolução variável;
- **SCUT-WMN:** amostras recortadas de hidrômetros reais;
- **SmartEyes:** banco proprietário com 12.000 imagens coloridas de hidrômetros (resolução 64×96), adquiridas em campo.

A pergunta central é: até que ponto as técnicas clássicas de aumento de dados conseguem melhorar o desempenho de modelos embarcados em contextos restritos, sem comprometer a estabilidade ou exceder os recursos disponíveis?

As seções seguintes descrevem a seleção de tecnologias, o conjunto de transformações avaliadas, a metodologia experimental adotada e os resultados obtidos.

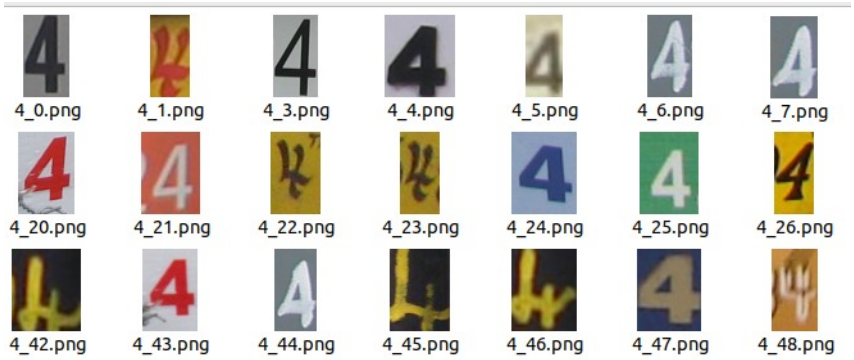


Figura 14 – Exemplo imagens dataset Kaggle NumOCR

6.3 Seleção de Tecnologias

Todos os experimentos foram implementados em Python utilizando a biblioteca Keras, que fornece uma interface de alto nível para definição e treinamento de modelos. O backend utilizado foi o TensorFlow 2.13, com suporte a GPU para acelerar os ciclos de treinamento.

Após o treinamento, os modelos foram convertidos para o formato TensorFlow Lite (`.tflite`) e quantizados para 8 bits, visando compatibilidade com o pipeline de inferência embarcada. A biblioteca `TensorFlow Lite for Microcontrollers` foi empregada para execução no firmware do ESP32-S3. Essa biblioteca é escrita em C/C++, não depende de sistema operacional e permite que os modelos sejam executados completamente em memória, o que é essencial para dispositivos com restrições severas de RAM.

O modelo de referência utilizado foi o **LeNet-5**, com a mesma topologia adotada no capítulo anterior. A arquitetura foi mantida fixa ao longo de todos os testes para garantir comparabilidade entre os diferentes cenários de *data augmentation*. Os hiperparâmetros de treinamento também foram padronizados: 30 épocas, *batch size* = 512, otimizador Adam com taxa de aprendizado padrão. A inicialização foi determinística, com semente global (`seed` = 42) aplicada a NumPy, TensorFlow e geradores pseudoaleatórios do sistema.

A arquitetura LeNet-5 foi escolhida por três motivos principais:

1. Apresenta desempenho robusto mesmo após quantização (acurácia $> 97\,\%$);
2. Seu tamanho e profundidade são adequados para execução em dispositivos com 2–8 MB de SRAM;
3. Serve como base confiável para avaliação isolada das técnicas de aumento de dados.

Três bases de dados foram utilizadas:

- **Kaggle NumOCR** – 785 imagens de dígitos impressos com resoluções variadas, ruído sintético e fundo claro 14;

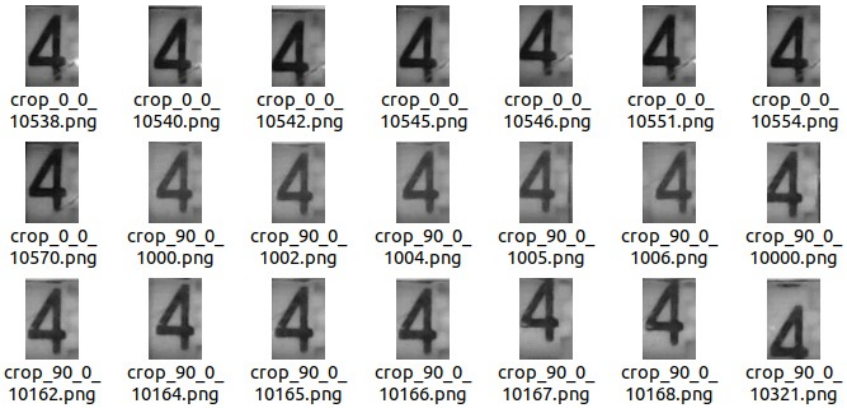


Figura 15 – Exemplo imagens dataset SCUT Water-Meter Number

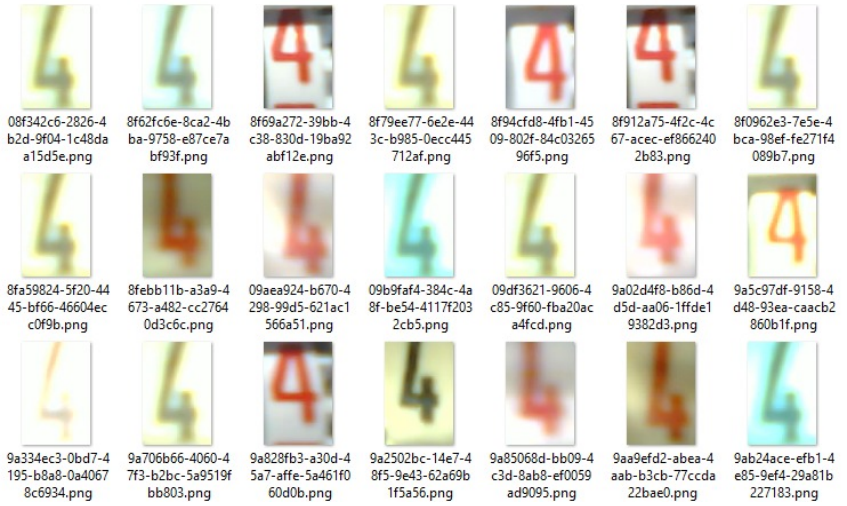


Figura 16 – Exemplo imagens dataset SmartEyes

- **SCUT-WMN** – 2.000 amostras da base SCUT Water-Meter Number, recortadas dos “easy samples” 15;
- **SmartEyes** – 12 000 imagens coloridas 64×96 px de hidrômetros reais. Dessas, 10.000 foram usadas para treino e validação (com *hold-out* de 20 %) e 2.000 para teste 16.

Todas as imagens foram convertidas para tons de cinza antes do treinamento e redimensionadas para 64×96 pixels, padronizando o pipeline de entrada.

6.4 Técnicas de Data Augmentation Selecionadas

Foram selecionadas nove técnicas de *data augmentation* clássicas, todas aplicadas diretamente no espaço de entrada e consideradas *label-preserving*, ou seja, não alteram semanticamente o conteúdo da imagem nem comprometem o rótulo. As técnicas selecionadas são:

a) **Ruído** – adição de ruído gaussiano ou ruído do tipo salt-and-pepper, para simular sensores ruidosos e condições de iluminação adversas;

b) **Ajuste de brilho** – modificação aditiva ou multiplicativa da intensidade global da imagem, para simular variações de iluminação;

c) **Ajuste de contraste** – amplificação ou compressão da faixa dinâmica da imagem;

d) **Translação horizontal** – deslocamento dos dígitos para a esquerda ou direita;

e) **Translação vertical** – deslocamento para cima ou para baixo;

f) **Rotação** – rotação da imagem em torno de seu centro;

g) **Cisalhamento (shear)** – deformação da imagem em uma direção horizontal ou vertical;

h) **Desfoque gaussiano (gaussian blur)** – aplicação de um filtro gaussiano para suavizar a imagem;

i) **Deformação elástica** – aplicação de campos de deslocamento suave que distorcem levemente a imagem mantendo a estrutura geral.

Transformações como *flip* horizontal ou zoom excessivo foram deliberadamente excluídas por seu potencial de alterar o valor semântico dos dígitos — por exemplo, um "6" invertido pode se assemelhar a um "9", invalidando a premissa de preservação de rótulo.

Cada técnica foi testada em três faixas de intensidade — *baixa*, *média* e *alta* — definidas empiricamente a partir de experimentos preliminares. Os parâmetros foram ajustados para cobrir uma gama ampla de perturbações, mas sem induzir distorções irreversíveis. Os exemplos de intensidade variam conforme a técnica (por exemplo, sigma do ruído, fator de brilho ou grau de rotação).

Todas as transformações foram implementadas por meio de funções personalizadas compatíveis com o *ImageDataGenerator* do Keras, permitindo controle determinístico sobre o processo de geração. Uma semente global foi utilizada em todos os casos para garantir reprodutibilidade dos resultados.

Os dados aumentados foram gerados em volumes crescentes — de 0 % até 500 % do tamanho original da base — em passos de 50 %. Ou seja, para cada base e cada configuração de técnica + intensidade, foram produzidos 11 conjuntos, totalizando mais de 300 combinações únicas ao longo dos experimentos.

6.5 Metodologia Experimental

Conforme descrito no capítulo 5, este trabalho busca um protocolo de uso de *data augmentation* para a melhoria da acurácia de modelos de detecção de dígitos, conforme explicado abaixo.

O protocolo experimental foi concebido para avaliar, de forma sistemática e parametrizada, os efeitos das técnicas de *data augmentation* sobre o desempenho do modelo LeNet-5, respeitando as restrições de execução embarcada em dispositivos ESP32-S3.

Para cada uma das três bases de dados — Kaggle NumOCR, SCUT-WMN e SmartEyes — foram realizados ciclos independentes de experimentação. Cada ciclo consistiu na aplicação de uma única técnica de *augmentation* (por exemplo, ruído ou rotação), em uma das três intensidades pré-definidas (baixa, média, alta). A cada combinação {técnica, intensidade}, foi gerado um número crescente de imagens sintéticas, de 0 % a 500 % do tamanho da base original, em passos de 50 %.

O modelo LeNet-5 foi treinado a partir do conjunto aumentado correspondente e avaliado sobre um conjunto de teste fixo, não afetado pelas transformações. O checkpoint de maior acurácia em *validation* foi selecionado e convertido para o formato `.tflite`, com quantização de 8 bits.

O modelo quantizado foi então testado em três placas ESP32-S3 N8R2 idênticas. Foram coletadas as seguintes métricas:

- **Acurácia (%)** no conjunto de teste fixo;
- **Tempo médio de inferência (ms)** por amostra, com desvio-padrão entre placas;
- **Tamanho do modelo** (em KB) após quantização;
- **Coeficiente angular da tendência** – inclinação da reta de regressão linear que relaciona o volume de dados sintéticos adicionados com a acurácia final.

A Figura 17 ilustra o fluxo de execução do experimento para cada configuração testada. Todas as execuções foram automatizadas por um único *script* Python, controlado por parâmetros que especificavam a base, técnica, intensidade e volume de dados.

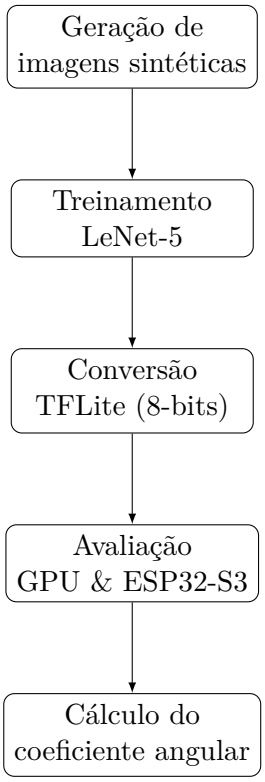


Figura 17 – Fluxo resumido dos experimentos de *data augmentation*

Algorithm 1 Geração, treinamento e avaliação para cada técnica de *augmentation*

Require: Bases D_1, D_2, D_3; Técnicas T_j; Intensidades I_k; Volumes V_ℓ

for cada base D_i **do**
  for cada técnica T_j **do**
    for cada intensidade I_k **do**
      for cada volume $V_\ell \in \{0, 50, \ldots, 500\}\%$ **do**
        Aplicar T_j com I_k para gerar $V_\ell\%$ de dados sintéticos
        Treinar modelo LeNet-5 e salvar melhor checkpoint
        Converter para TFLite e quantizar em 8 bits
        Avaliar modelo na GPU e nas três placas ESP32-S3
        Registrar acurácia, tempo médio e tamanho do modelo
      end for
      Calcular inclinação da curva de acurácia (coeficiente angular)
    end for
  end for
end for

Além das execuções padrão com o conjunto completo de treinamento, foi conduzido um experimento de varredura com subconjuntos de 25 %, 50 % e 100 % da base original. Cada subconjunto foi expandido com os mesmos incrementos percentuais de dados sintéticos, permitindo observar o comportamento de saturação da acurácia — isto é, identificar o ponto em que o aumento do volume de imagens não traz mais ganhos, ou até degrada o desempenho por sobreajuste.

Todas as configurações de compilação, versão do compilador (`xtensa-esp32-elf-gcc`) e bibliotecas embarcadas (`tflite-micro`) foram mantidas constantes em todos os experimentos para garantir comparabilidade.

6.6 Resultados e Análise

Após a execução dos modelos para os três *datasets*, foram obtidas estatísticas que auxiliam o processo de tomada de decisão. A análise a seguir apresenta um processo padronizado de testes com várias técnicas de *data augmentation* e seus respectivos intervalos de parâmetros.

6.6.1 Comparação das Técnicas de Data Augmentation

Conforme esperado, cada conjunto de dados analisado neste estudo apresenta variações próprias entre suas imagens; por isso, aplicar exatamente as mesmas técnicas e parâmetros de *data augmentation* a todos os *datasets* é inviável.

Conforme descrito na Seção 6.2, os resultados foram tabulados mostrando a variação das acurácias para cada técnica em diferentes parâmetros, aumentando o tamanho do *dataset* de 0 a 500%.

Nas Tabelas 12 e 13, observa-se a inclinação resultante da acurácia para cada conjunto, técnica e parâmetro empregado. Inclinação positiva indica tendência de aumento de acurácia; inclinação negativa, tendência de queda. Quanto maior o valor, mais impactante é a técnica para melhorar o desempenho.

Para avaliar o impacto do *data augmentation* no reconhecimento de dígitos, aplicamos as técnicas selecionadas a três *datasets* distintos: Kaggle NumOCR (dígitos impressos), SCUT (dígitos de hidrômetros públicos) e SmartEyes (conjunto proprietário).

A Tabela 12 apresenta os resultados de aumento para o Kaggle, cujo ponto de partida (sem aumento de dataset) era 85,26% de acurácia. A Tabela 13 traz os resultados para o SCUT, que iniciou em 98,85%. Cada técnica foi testada em vários parâmetros para identificar configurações ideais, medindo a maior acurácia obtida e a inclinação correspondente.

6.6.1.1 Análise Detalhada

Esta seção comenta o efeito de cada técnica sobre o desempenho nos *datasets* SCUT e Kaggle, destacando tendências e estratégias eficazes:

- **Blur**: o kernel **7×7** elevou levemente a acurácia em ambos os conjuntos. No SCUT chegou a **99,23%** (inclinação 0,006%), enquanto no Kaggle atingiu **91,58%** (incli-

Tabela 12 – Kaggle NumOCR – Acurácia Original: 85,26%

Técnica	**Métrica**	**Intervalo**	**Acurácia**	**Inclinação**
Blur	tamanho do kernel	7-7	91,58%	**0,7389%**
		15-15	91,58%	0,0139%
		30-30	86,32%	-1,9615%
Brilho	brilho	0.1-0.3	86,32%	**1,0119%**
		0.3-0.6	90,53%	0,4236%
		0.6-1.0	88,42%	0,6432%
Contraste	contraste	0-1	90,53%	0,4310%
		1-2	86,32%	-0,7933%
		2-3	89,47%	0,0268%
Deformação elástica	α e σ	100-6	93,68%	0,9155%
		500-12	93,68%	**1,3258%**
		1000-16	89,47%	0,1292%
Translação horizontal	percentual	0.3	93,68%	**1,2781%**
		0.6	87,37%	-0,1635%
		1	90,53%	0,7515%
Ruído	escala	5-85	91,58%	**0,6286%**
		85-170	89,47%	0,5399%
		170-255	89,47%	-1,1066%
Rotação	ângulo	30	90,53%	-1,1898%
		60	86,32%	-1,8662%
		120	87,37%	-2,2567%
Shear	percentual	0.1	90,53%	**0,7037%**
		0.3	89,47%	0,6004%
		0.6	90,53%	0,3628%
Translação vertical	percentual	0.3	89,47%	0,5674%
		0.6	88,42%	-0,6427%
		1	89,47%	**1,0797%**

nação 0,6018%), indicando redução de ruído útil sobretudo em imagens de menor qualidade.

- **Brilho**: ajustes moderados de brilho trouxeram ganhos expressivos — SCUT (**99,23%**, inclinação 0,0316%) e Kaggle (**90,53%**, inclinação 0,8751%) — sugerindo compensação de contraste insuficiente.

- **Contraste**: variações baixas pouco alteraram o SCUT (99,04%), mas aumentaram ligeiramente o Kaggle (**90,53%**, inclinação 0,2942%), mostrando que dígitos impressos se beneficiam de contraste realçado.

Tabela 13 – SCUT – Acurácia Original: 98,85%

Técnica	**Métrica**	**Intervalo**	**Acurácia**	**Inclinação**
Blur	tamanho do kernel	7-7	99,23%	0,0060%
		15-15	99,23%	0,0275%
		30-30	99,04%	0,0153%
Brilho	brilho	0.1-0.3	99,23%	0,0183%
		0.3-0.6	99,23%	**0,0316%**
		0.6-1.0	99,42%	**0,0480%**
Contraste	contraste	0-1	99,04%	-0,0374%
		1-2	99,04%	-0,0770%
		2-3	99,04%	-0,0153%
Deformação elástica	α e σ	100-6	99,42%	**0,0423%**
		500-12	99,42%	0,0214%
		1000-16	99,04%	0,0124%
Translação horizontal	percentual	0.3	99,62%	-0,0205%
		0.6	99,23%	0,0025%
		1	99,42%	0,0060%
Ruído	escala	5-85	99,04%	0,0151%
		85-170	98,85%	0,0091%
		170-255	98,85%	0,0116%
Rotação	ângulo	60	99,62%	-0,0011%
		120	98,65%	-0,0135%
		180	98,85%	-0,0080%
Shear	percentual	0.3	99,23%	0,0333%
		0.6	99,23%	**0,0363%**
		1	99,42%	-0,0030%
Translação vertical	percentual	0.3	99,04%	**0,0331%**
		0.6	99,04%	-0,0454%
		1	99,04%	-0,0020%

- **Deformação elástica**: ganhos moderados em ambos os casos — SCUT **99,42%** (0,0423%) e Kaggle **93,68%** — evidenciam a utilidade de deformações sutis.
- **Translação horizontal**: pequenos deslocamentos melhoraram SCUT para **99,62%** (inclinação negativa pequena) e Kaggle para **93,68%** (inclinação 1,1413%), sugerindo correção de desalinhamentos.
- **Ruído**: níveis controlados elevaram SCUT (**99,04%**, 0,0151%) e Kaggle (**91,58%**, 0,4918%), indicando maior robustez sem distorção excessiva.
- **Rotação**: ângulos maiores reduziram a acurácia. SCUT tolerou rotação moderada

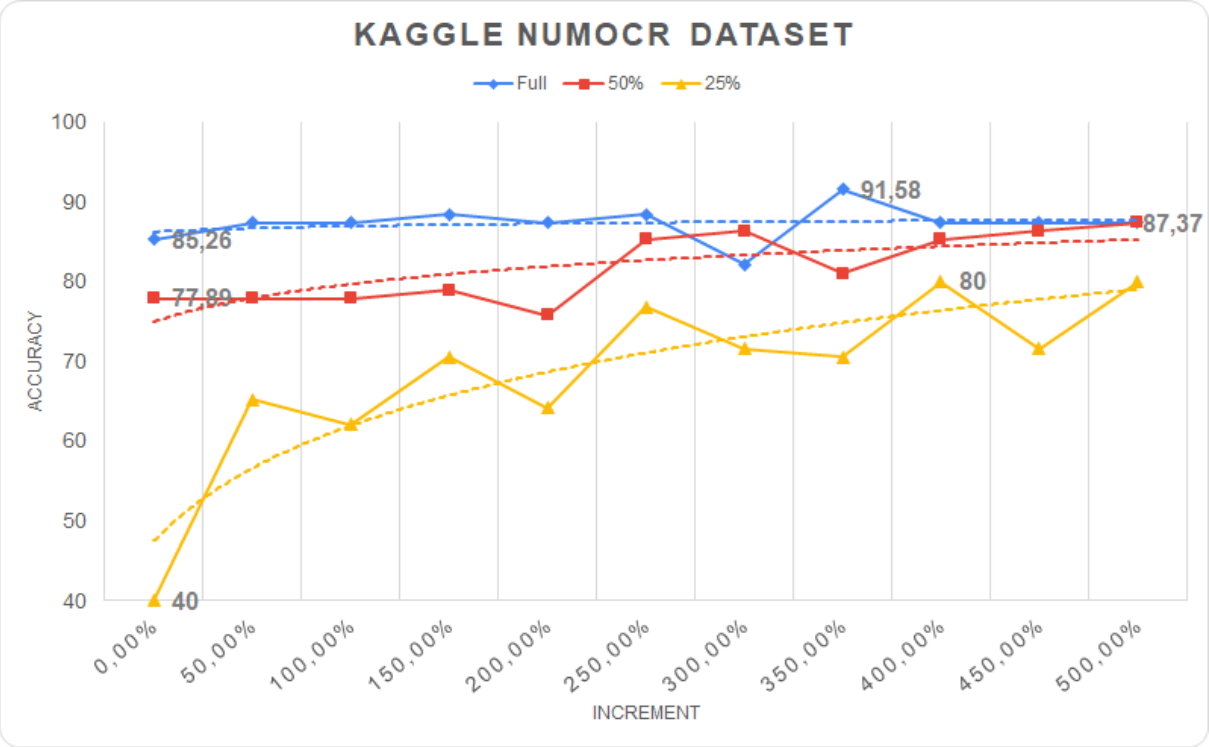


Figura 18 – Gráfico de Data Augmentation para o dataset Kaggle

(**99,62%**, -0,0011%), enquanto o Kaggle caiu para **86,32%**, mostrando sensibilidade a rotações amplas.

- **Shear**: cisalhamentos pequenos foram benéficos — SCUT **99,23%** (0,0363%) e Kaggle **90,53%** (0,7037%) — auxiliando no reconhecimento de dígitos inclinados.
- **Translação vertical**: impacto mínimo; SCUT **99,04%** (0,0331%) e Kaggle **89,47%**, sugerindo tolerância a deslocamentos leves.

6.6.1.2 Resumo dos Resultados

De modo geral, o SCUT atingiu acurácias consistentemente superiores às do Kaggle, possivelmente em razão da maior qualidade de imagem. Os ajustes de **brilho** e a **translação horizontal** foram as transformações mais eficazes em ambos os conjuntos. Em contrapartida, rotações excessivas mostraram-se prejudiciais, sobretudo no dataset Kaggle. Esses resultados reforçam que a efetividade do *data augmentation* depende do *dataset*: qualidade das imagens e acurácia de referência influenciam diretamente o impacto das transformações. Portanto, uma estratégia equilibrada de aumento de dados é fundamental para otimizar o desempenho sem incorrer em retornos decrescentes.

6.6.2 Comparação do Volume de Data Augmentation

A combinação de estratégias de aumento ajustadas de forma ótima pode superar, com larga vantagem, a aplicação isolada de técnicas individuais. Isso ressalta a importância

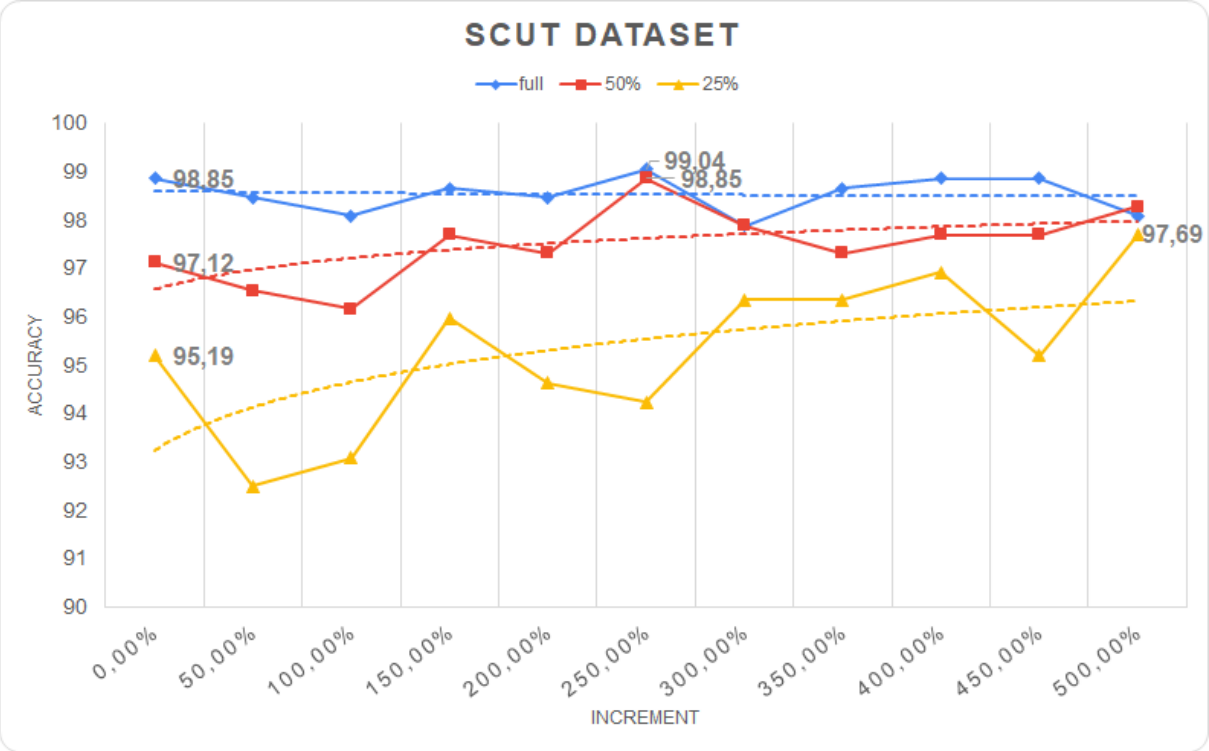


Figura 19 – Gráfico de Data Augmentation para o dataset SCUT

de uma abordagem integrada e cuidadosa ao *data augmentation* em tarefas de visão computacional. Existe, na biblioteca Keras, uma função que executa várias transformações com parâmetros aleatórios; entretanto, ela não contempla todas as técnicas estudadas neste trabalho e tampouco permite priorizar transformações mais eficazes. Por essa razão, utilizamos um código estendido sobre o Keras, no qual se pode controlar explicitamente a probabilidade de cada técnica.

Além disso, avaliou-se o uso de data augmentation *online* durante o treinamento, por meio das funções `ImageDataGenerator` do Keras que aplicam transformações aleatórias em tempo real. Nos experimentos preliminares, essa abordagem resultou em acurácias inferiores às obtidas com a geração controlada de dados: a combinação aleatória de ruídos, rotações e outras distorções gerou imagens deformadas que comprometiam a correta identificação dos dígitos, tornando-a ineficiente para o foco deste trabalho. Assim, optamos por gerar conjuntos aumentados de forma *offline*, escolhendo e parametrizando apenas as técnicas que efetivamente melhoraram a acurácia, conforme apresentado nas subseções anteriores.

Os testes foram conduzidos aplicando as técnicas que apresentaram melhor desempenho na seção anterior, com probabilidades de aplicação proporcionais aos coeficientes angulares positivos observados. No dataset Kaggle, utilizaram-se *blur*, brilho, elastic, translação horizontal, ruído, cisalhamento e translação vertical. No SCUT, selecionaram-se brilho, elastic, cisalhamento e translação vertical. As técnicas escolhidas estão destacadas nas Tabelas 12 e 13. Nesta subseção analisamos o efeito de incrementar o volume de dados

sintéticos nesses dois conjuntos; os resultados estão nas Figuras 18 e 19.

O procedimento avalia o impacto de volumes crescentes de *augmentação*, partindo do conjunto completo, de 50% e de 25% dos dados originais, e acrescentando-lhes até 500% de imagens geradas.

Dataset Kaggle: Conforme a Figura 18, o aumento de volume tem impacto variado, dependente do tamanho inicial. Com 100 % do dataset, a acurácia base era 85,26 %; os acréscimos gradativos elevaram-na até **91,58 %** quando o volume atingiu 400 %. A partir daí, a curva estabiliza em torno de 87,37 %, sugerindo saturação.

Para o subconjunto inicial de 50 %, observa-se tendência semelhante: a acurácia cresce de forma estável e atinge **80 %** quando o volume chega a 250 %. A regressão logarítmica (linha vermelha tracejada) indica tendência ascendente, porém menos acentuada que no conjunto completo.

Com apenas 25 % dos dados originais, o efeito é mais volátil. A acurácia parte de 40 % e sobe até cerca de 80 % em 400 % de aumento, mas com maior variabilidade. A tendência logarítmica (linha amarela) mostra ganho mais lento, revelando que, embora o *augmentation* ajude mesmo com poucos dados, os benefícios são menos consistentes.

Dataset SCUT: A Figura 19 mostra comportamento análogo. O SCUT parte de acurácia base de 98,85 %, o que explica os ganhos menores. Com 100 % do conjunto, a acurácia atinge **99,04 %** em 200 % de aumento e depois se estabiliza, apresentando platô (linha azul tracejada).

Para o subconjunto de 50 %, a acurácia inicial é 97,12 % e cresce até **98,85 %** em 300 % de aumento, seguindo tendência logarítmica (linha vermelha) mais suave. O subconjunto de 25 % inicia em 95,19 % e alcança **97,69 %** em 500 %, com crescimento lento (linha amarela).

Em resumo, tanto Kaggle quanto SCUT mostram que o aumento de dados melhora a acurácia até certo ponto; depois disso, os ganhos tornam-se marginais. O efeito de retornos decrescentes é mais evidente no SCUT, cujo patamar inicial já é alto.

6.6.2.1 Resumo dos Resultados

Os experimentos confirmam que o *data augmentation* pode elevar significativamente o desempenho, sobretudo quando o conjunto original possui tamanho razoável. Entretanto, existe um volume a partir do qual os ganhos se estabilizam ou até decrescem, caracterizando saturação. O Kaggle, por ter acurácia inicial mais baixa, beneficia-se de modo mais visível; o SCUT, com base elevada, apresenta incrementos modestos. Assim, embora o *data augmentation* seja ferramenta poderosa, sua eficácia diminui conforme o volume sintético cresce, especialmente em bases que já partem de alto desempenho.

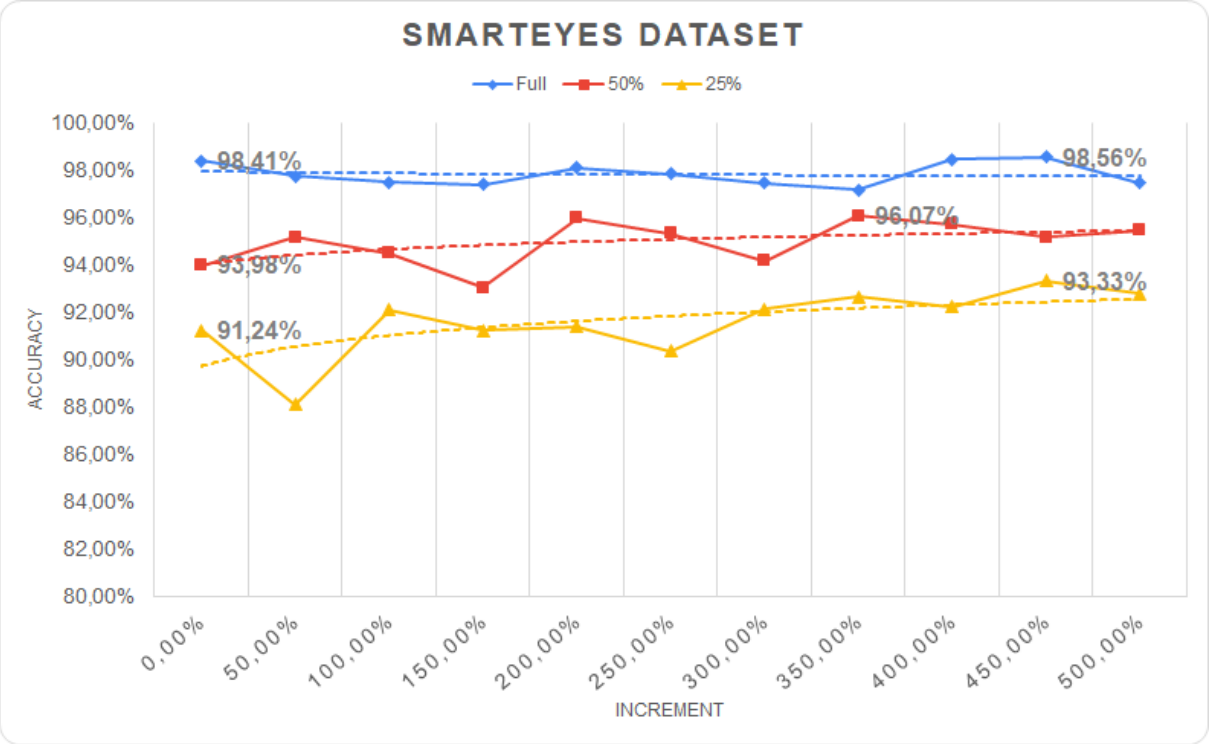


Figura 20 – Gráfico de Data Augmentation para o dataset privado SmartEyes

6.6.3 Dataset Privado SmartEyes

Os resultados obtidos trazem implicações diretas para implantações de IoT em cenários de cidades inteligentes. As técnicas de *data augmentation* consideradas ótimas neste estudo levam em conta não apenas o ganho de acurácia, mas também as limitações em memória, processamento e energia em dispositivos de borda. Esse equilíbrio entre desempenho e consumo de recursos é fundamental para que soluções IoT possam ser escaladas na infraestrutura urbana.

O presente trabalho também avaliou o impacto das técnicas de *data augmentation* na tarefa de reconhecimento de dígitos dentro do projeto *SmartEyes*. Replicamos o processo sobre nosso conjunto de dados proprietário (Figura 20), validando o mesmo comportamento observado nos datasets públicos. Os resultados demonstram que, com a escolha adequada de técnicas e parâmetros, é possível melhorar a acurácia mesmo quando se parte de um conjunto inicial reduzido. Vale ressaltar, contudo, que um volume mínimo de dados reais continua sendo decisivo para que o modelo final alcance desempenho robusto.

7 Escaneamento de Características de Imagens

Este capítulo cumpre um *papel exploratório*: demonstrar, por meio de um estudo preliminar sobre o *dataset* **SmartEyes**, que ainda há espaço para estratégias mais sofisticadas de melhoria dos modelos de reconhecimento de dígitos além das abordadas nos Capítulos 5 e 6. A ênfase recai sobre duas ideias:

(a) **Parametrização guiada por métricas**: varrer previamente o conjunto de imagens em busca de *desvios* de ruído, contraste, brilho e desfoque e outras características, empregando esses valores como *pontos de partida* para calibrar o *data augmentation* (DA);

(b) **Parametrização distinta por classe**: aplicar níveis de DA diferentes para cada dígito (0–9), uma vez que as métricas mostram dispersões heterogêneas entre classes.

7.1 Metodologia de Escaneamento

Foram implementados protótipos em `Python` que percorreram todas as 10 pastas do *SmartEyes* (uma por dígito) e calcularam, para cada imagem em tons de cinza de 64×96 px:

- **Ruído**: variância do componente de alta-frequência obtida por filtro passa-alta;
- **Contraste**: desvio-padrão do histograma normalizado;
- **Brilho**: média global de intensidade (0–255);
- **Desfoque**: variância do mapa de Laplaciano (σ^2_{Lap}).

Em seguida, computou-se a variância intra-pasta de cada métrica. A Tabela 14 consolida os valores encontrados.

7.2 Discussão dos Resultados

Observam-se dispersões expressivas entre dígitos. O dígito *1*, por exemplo, apresenta a maior variância de brilho, ao passo que o dígito *9* sofre o maior desfoque médio. Esses desvios sugerem que:

Tabela 14 – Variância de ruído, contraste, brilho e desfoque por classe no *SmartEyes*

Classe	**Ruído**	**Contraste**	**Brilho**	**Desfoque**
0	12 528.28	47.64	152.14	61.09
1	15 621.27	53.47	191.70	57.89
2	14 423.61	51.18	176.46	66.66
3	13 822.85	47.07	169.69	35.58
4	15 761.45	42.04	196.29	42.22
5	15 431.53	50.92	190.06	71.41
6	14 917.24	52.28	182.80	55.87
7	15 868.44	53.59	195.08	57.30
8	14 727.76	56.25	179.23	62.25
9	14 640.83	56.13	178.19	84.33
Média	**14 774.33**	**51.06**	**181.16**	**59.46**

- **DA dependente de classe** pode mitigar *underfitting* em classes "fáceis" (baixa variância) e *overfitting* em classes "difíceis" (alta variância);
- **Limiares iniciais** para injeção de ruído, ajuste de contraste ou blur podem ser escolhidos automaticamente a partir dos valores médios de cada métrica, reduzindo tentativas manuais.

7.3 Perspectivas de Continuidade

O presente escaneamento abre caminho para duas linhas de investigação já previstas nos *Trabalhos Futuros*:

(i) **AutoDA guiado por métricas**: incorporar os desvios calculados a algoritmos de *AutoDA* — com busca em espaço de políticas — de forma a restringir o intervalo de parâmetros a valores plausíveis para cada classe;

(ii) **Extensão a outras características**: repetir o escaneamento para rotação implícita, deslocamentos e deformações geométricas, fornecendo um mapa completo de variabilidade do conjunto.

Dessa forma, o capítulo reforça que o ganho de desempenho obtido com *data augmentation* não precisa ser fruto de tentativas empíricas; ele pode (e deve) ser orientado por estatísticas objetivas do conjunto de imagens sob estudo, respeitando as limitações de memória e energia dos dispositivos AIoT que compõem as infraestruturas de cidades inteligentes.

8 Conclusão

A presente dissertação investigou, de forma sistemática e com elevado rigor experimental, a viabilidade de executar modelos de *visão computacional* para reconhecimento de dígitos em *dispositivos AIoT* de baixo consumo, tomando como caso de uso a leitura automatizada em medidores de utilidades. Os objetivos delineados na Introdução — construir *datasets* específicos, comparar arquiteturas de *machine-learning* em microcontroladores ESP32-S3, explorar e parametrizar técnicas de *data augmentation* e propor um *framework* replicável — foram plenamente atingidos, conforme se resume a seguir.

8.1 Principais Contribuições

1. **Avaliação comparativa de modelos em hardware restrito**. Cinco arquiteturas (LeNet-5, TinyML, MobileNet, MiniVGG e SqueezeNet) foram comparadas após a conversão para TensorFlow Lite Micro. O modelo *LeNet-5* destacou-se com **98,95 % de acurácia**, tamanho de **0,30 MB** e **21 min** para inferir 10 000 amostras — a melhor relação precisão × custo computacional para *smart metering*.

2. **Metodologia padronizada de *data augmentation***. Propôs-se um algoritmo de varredura em três escalas (baixa, média e alta) para nove transformações fotométricas e geométricas, aliado a um controle do volume gerado (0 %–500 %). O procedimento permitiu **isolar o ganho marginal de cada técnica** e **identificar saturação de desempenho** acima de 400 % de aumento, sobretudo no *dataset* Kaggle NumOCR.

3. **Validação cruzada em múltiplos *datasets***. Os mesmos parâmetros foram testados em duas bases públicas (Kaggle NumOCR e SCUT-WMN) e em uma base proprietária (SmartEyes), demonstrando que *brilho controlado, translação horizontal* e *deformação elástica* formam o núcleo mínimo de aumento de dados para cenários com dígitos em mostradores analógicos.

4. ***Framework* reprodutível**. Todo o fluxo — do treinamento em estação de trabalho à inferência embarcada — foi encapsulado em *scripts* determinísticos, com *checkpoints* e fixação de *seeds*, constituindo um **guia metodológico** para pesquisadores e engenheiros de AIoT.

8.2 Discussão dos Resultados

Os experimentos confirmaram que **modelos convolucionais clássicos, aliados a quantização e podas estruturadas, permanecem competitivos** em 2025 para OCR embarcado. Arquiteturas profundas (p. ex. MiniVGG) atingem acurácia ligeiramente superior, mas o custo energético (≈ 28 h para 10 000 inferências) inviabiliza seu uso de campo. Modelos mais enxutos (SqueezeNet) oferecem menor latência, porém com perda de precisão de cerca de 1 p.p. — diferença inaceitável quando leituras incorretas impactam faturamento.

Quanto ao *data augmentation*, verificou-se que **incrementos progressivos beneficiam sobretudo conjuntos com baixa variabilidade intrínseca**. Bases já equilibradas (SCUT-WMN, 98,85 % de *baseline*) apresentam ganhos marginais e rápida saturação, corroborando a hipótese de *diminuição de retorno* após determinado volume sintético.

8.3 Limitações

- O estudo concentrou-se em *dígitos impressos do tipo odômetro*; não foram incluídos ponteiros analógicos nem mostradores LCD, restringindo a generalização imediata.
- A composição ótima das políticas de aumento exigiu *supervisão humana*; abordagens AutoAugment ou RandAugment não foram exploradas.
- A avaliação energética considerou apenas *tempo de inferência*; medições de corrente e impacto na vida útil da bateria permanecem como trabalho futuro.

8.4 Trabalhos Futuros

1. **AutoDA em borda**: investigar busca de políticas de *data augmentation* em espaço reduzido, considerando custo de CPU e memória.
2. **AutoDA por classe com varredura de características**: desenvolver algoritmos de *Auto Data Augmentation* capazes de *escanear previamente* o conjunto de imagens, estimar distribuição de ruído, contraste e orientação *por classe* (dígitos 0–9) e, a partir desse diagnóstico, gerar políticas de aumento diferenciadas e parametrizadas para cada classe, maximizando o ganho onde o modelo apresenta maior incerteza.
3. **Aprendizado incremental *on-device***: estudar *continual learning* para adaptação a novos algarismos ou degradação do visor sem necessidade de upload de imagens.
4. **Quantização mista e *pruning* sensível à energia**: aplicar técnicas de 4/8 bits híbridos e podas orientadas por perfil de consumo no ESP32-S3.

5. **Extensão multimodal**: integrar leitura de ponteiros via detecção de círculos e estimativa de ângulo, compartilhando camadas iniciais da CNN.

8.5 Síntese Final

Esta dissertação **demonstra, com evidências quantitativas e uma metodologia replicável, que é possível unir visão computacional, *data augmentation* e microcontroladores de baixo consumo para modernizar medidores legados**, mantendo autonomia de bateria e dispensando infraestrutura de nuvem. Ao prover uma *arquitetura de referência*, um *protocolo experimental* e *diretrizes concretas* para seleção de modelos e técnicas de aumento de dados, o trabalho contribui para consolidar práticas de *Edge-AI* em *smart metering* e, em sentido mais amplo, para a sustentabilidade das cidades inteligentes. Os resultados obtidos, aliados às recomendações propostas, posicionam-se para impactar tanto a comunidade acadêmica quanto a indústria, reduzindo barreiras técnicas e impulsionando a adoção de soluções AIoT de medição automática.

Referências

ADAM, Andrew G. Howard e Menglong Zhu e Bo Chen e Dmitry Kalenichenko e Weijun Wang e Tobias Weyand e Marco Andreetto e Hartwig. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. Citado 5 vezes nas páginas xi, xxv, xxvi, xxxii e xxxv.

AKBıYıK, Sadiye; ÖZYURT, Fatih; ÇELIK, Yakup. Data augmentation for deep learning based on fusion of deep convolutional generative adversarial networks. *IEEE Access*, v. 7, p. 141034–141042, 2019. Citado 2 vezes nas páginas xxvii e xxviii.

AL, Forrest N. Iandola1 et. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *ICLR2017*, 2017. Citado na página xxiii.

ASISH, Kottakota; TEJA, P Sarath; CHANDER, R Kishan; HEMA, Dr D. Neurowrite: Predictive handwritten digit classification using deep neural networks. *arXiv preprint arXiv:2311.01022*, 2023. Citado na página xliv.

AWAN, SS Ahmed e Z Mehmood e IA. A novel technique for handwritten digit recognition using deep learning. *Journal of Computer Applications*, 2023. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2023/2753941>. Citado 4 vezes nas páginas xvii, xxxii, xxxvi e xxxvii.

CUBUK, Ekin D; ZOPH, Barret; MANÉ, Dandelion; VASUDEVAN, Vijay; LE, Quoc V. Autoaugment: Learning augmentation strategies from data. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2019. p. 113–123. Citado na página lix.

CUBUK, Ekin D; ZOPH, Barret; SHLENS, Jonathon; LE, Quoc V. Randaugment: Practical automated data augmentation with a reduced search space. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2020. p. 702–703. Citado na página lix.

GAO, Yunze; ZHAO, Chaoyang; WANG, Jinqiao; LU, Hanqing. Automatic watermeter digit recognition on mobile devices. In: *Proceedings of the International Conference on Internet Multimedia Computing and Service*. [S.l.: s.n.], 2018. p. 87–95. Citado na página xxxv.

GULSHAD, R. et al. Built-in data augmentation. *Pattern Recognition Letters*, v. 138, p. 33–40, 2021. Citado 2 vezes nas páginas xxviii e xxxi.

GóMEZ, Lluis; RUSIñOL, Marçal; KARATZAS, Dimosthenis. Cutting sayre's knot: Reading scene text without segmentation. application to utility meters. In: *13th IAPR International Workshop on Document Analysis Systems (DAS)*. [S.l.: s.n.], 2018. p. 97–102. Citado na página xxxv.

HAFFNER, Yann Lecun e Leon Bottou e Yoshua Bengio e Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. Citado 5 vezes nas páginas xi, xxiii, xxiv, xxxiv e xliv.

HIDAYAT, Alam Ahmad; PURWANDARI, Kartika; CENGGORO, Tjeng Wawan; PARDAMEAN, Bens. A convolutional neural network-based ancient sundanese character classifier with data augmentation. *Procedia Computer Science*, v. 179, p. 195–201, 2021. Citado na página xxxvii.

HINTON, Alex Krizhevsky e Ilya Sutskever e Geoffrey E. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, v. 25, p. 1097–1105, 2012. Citado na página xxviii.

HOLZINGER, Marcus D. Bloice e Christof Stocker e Andreas. Augmentor: An image augmentation library for machine learning. *Journal of Open Source Software*, v. 4, n. 35, p. 1057, 2019. Citado 2 vezes nas páginas xxviii e xxxvii.

INC, Google. *TensorFlow*. Disponível em: <https://github.com/tensorflow/tensorflow/releases/tag/v2.13.0>. Citado 2 vezes nas páginas xliv e xlviii.

KEUTZER, Forrest N. Iandola e Song Han e Matthew W. Moskewicz e Khalid Ashraf e William J. Dally e Kurt. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv preprint arXiv:1602.07360*, 2016. Citado 6 vezes nas páginas xi, xxv, xxvi, xxvii, xxxii e xxxv.

KHOSHGOFTAAR, Connor Shorten e Taghi M. Image data augmentation for deep learning: A comprehensive review. *Journal of Big Data*, v. 6, n. 60, 2019. Citado 2 vezes nas páginas xxvii e xxxiii.

KUMAR, K Kranthi; BHARADWAJ, Rahul; CH, Sheshank; SUJANA, S. Effective deep learning approach based on vgg-mini architecture for iris recognition. *Annals of the Romanian Society for Cell Biology*, p. 4718–4726, 2021. Citado na página xxv.

LAROCA, Rayson; ARAUJO, Alessandra B.; ZANLORENSI, Luiz A.; ALMEIDA, Eduardo C. de; MENOTTI, David. Towards image-based automatic meter reading in unconstrained scenarios: A robust and efficient approach. *IEEE Access*, v. 9, p. 67569–67584, 2021. Citado na página xxxiv.

LAROCA, Rayson; BARROSO, Victor; DINIZ, Matheus A.; GONçALVES, Gabriel R.; SCHWARTZ, William R.; MENOTTI, David. Convolutional neural networks for automatic meter reading. *Journal of Electronic Imaging*, v. 28, n. 1, p. 013023, 2019. Citado na página xxxiv.

MAJUMDER, Sagnik; MALSBURG, Christoph von der; RICHHARIYA, Aashish; BHANOT, Surekha. Handwritten digit recognition by elastic matching. *arXiv preprint arXiv:1807.09324*, 2018. Citado na página xliv.

MOHAMMADI, Mehdi; AL-FUQAHA, Ala; SOROUR, Sameh; GUIZANI, Mohsen. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, v. 20, n. 4, p. 2923–2960, 2018. Citado na página xxxvi.

MUMUNI, Alhassan; MUMUNI, Fuseini. Data augmentation: A comprehensive survey of modern approaches. *Array*, Elsevier, v. 16, p. 100258, 2022. Citado na página lviii.

NANNI, Loris; PACI, Michelangelo; BRAHNAM, Sheryl; LUMINI, Alessandra. Comparison of different image data augmentation approaches. *Journal of Imaging*, v. 7, n. 254, 2021. Citado 2 vezes nas páginas xxviii e xxxiii.

NAVARRO, Pablo Doñate; JAISWAL, Abhishek. *NumOCR Dataset*. Disponível em: <https://www.kaggle.com/datasets/pablodoatenavarro/numerical-images-ocr>. Citado na página xviii.

NITSCHKE, L. Taylor e G. Improving deep learning with generic data augmentation. *IEEE Symposium Series on Computational Intelligence*, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8628742>. Citado 3 vezes nas páginas xxviii, xxxiii e xxxvii.

RAY, Partha Pratim. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, Elsevier, v. 34, p. 1595–1623, 2022. Disponível em: <https://doi.org/10.1016/j.jksuci.2021.11.019>. Citado 5 vezes nas páginas xi, xxiv, xxv, xxxii e xxxv.

SAINI, Madan Lal; TRIPATHI, Bharat; MIRZA, Mohammed Sohail. Evaluating the performance of deep learning models in handwritten digit recognition. In: *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*. [S.l.: s.n.], 2023. p. 116–121. Citado 2 vezes nas páginas xxiii e xliv.

SRIPANUSKUL, Nuntida; BUAYAI, Prawit; MAO, Xiaoyang. Generative data augmentation for automatic meter reading using cnns. *IEEE Access*, v. 10, p. 41234–41245, 2022. Citado na página xxxvii.

SS, Ahmed; Z, Mehmood; IA, Awan. Integration of deep learning into the iot: A survey of techniques and challenges for real-world applications. *Electronics*, v. 12, n. 12, p. 4925, 2023. Disponível em: <https://doi.org/10.3390/electronics12244925>. Citado 2 vezes nas páginas xxii e xxxvi.

SUN, Jian; FARD, Ali Pourramezan; MAHOOR, Mohammad H. XnODR and XnIDR: Two accurate and fast fully connected layers for convolutional neural networks. *Journal of Intelligent and Robotic Systems*, Springer Science and Business Media LLC, v. 109, n. 1, sep 2023. Disponível em: <https://doi.org/10.1007%2Fs10846-023-01952-w>. Citado na página xliv.

VEELING, Bastiaan S.; LINMANS, Jasper; WINKENS, Jim; COHEN, Taco; WELLING, Max. Rotation equivariant cnns for digital pathology. *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2018. Citado na página xxviii.

WONG, Sebastien C; GATT, Adam; STAMATESCU, Victor; MCDONNELL, Mark D. Understanding data augmentation for classification: When to warp? p. 1–6, 11 2016. Citado na página xliv.

XIE, Wei; GUO, Ying; WU, Zhe; GU, Yonghao; DONG, Shibo; XIE, Guokang; XU, Wei. Towards high-accuracy digital detection in meter images: The ass-yolo algorithm and dig21000 dataset. *Neurocomputing*, v. 647, p. 130606, 2025. Citado na página xxxv.

YANG, F.; JIN, L.; LAI, S.; GAO, X.; LI, Z. *Scut Water Meter Number DataSet*. 2019. Disponível em: <https://github.com/HCIILAB/Water-Meter-Number-DataSet>. Citado na página xviii.

YANG, Zihan; SINNOTT, Richard O; BAILEY, James; KE, Qiuhong. A survey of automated data augmentation algorithms for deep learning-based image classification

tasks. *Knowledge and Information Systems*, Springer, v. 65, p. 2805–2861, 2023. Citado na página lix.

ZHANG, Jiawen; TAO, Dacheng. Empowering things with intelligence: a survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet of Things Journal*, v. 8, n. 10, p. 7789–7817, 2020. Citado 3 vezes nas páginas xix, xxii e xxxvi.

ZHANG, R Wang e W Li e L. Blur image identification with ensemble convolution neural networks. *Signal Processing*, 2019. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0165168418303116>. Citado 2 vezes nas páginas xxviii e xxxi.

ZORZI, Michele Centenaro e Lorenzo Vangelista e Andrea Zanella e Michele. Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios. *IEEE Communications Magazine*, v. 54, n. 7, p. 60–67, 2016. Citado 2 vezes nas páginas xi e xxx.

ZOU, Lihua; XU, Leyan; LIANG, Yuan; CHEN, Li; YIN, Jia; XU, Ning. Robust water meter reading recognition method for complex scenes. *Procedia Computer Science*, v. 183, p. 46–52, 2021. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050921004944?via%3Dihub>. Citado 3 vezes nas páginas xi, xix e xxiii.

ZOU, Lihua; XU, Leyan; LIANG, Yuan; CHEN, Li; YIN, Jia; XU, Ning. Robust water meter reading recognition method for complex scenes. *Procedia Computer Science*, v. 183, p. 46–52, 2021. Citado na página xxxi.