Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Automação de iluminação de ambientes utilizando aprendizado de máquina embarcado

Larissa Santana de Freitas Andrade

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientadora
Prof.a Dr.a Carla Cavalcante Koike

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Automação de iluminação de ambientes utilizando aprendizado de máquina embarcado

Larissa Santana de Freitas Andrade

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof.a Dr.a Carla Cavalcante Koike (Orientadora)
CIC/UnB

Prof.a Dr.a Roberta Barbosa Oliveira
CIC/UnB

Prof. Dr. Marcus Vinicius Lamar
CIC/UnB

Prof. Dr. João Luiz Azevedo de Carvalho
Coordenador do Curso de Engenharia da Computação

Brasília, 27 de julho de 2023

Dedicatória

Eu dedico este trabalho a minha família, que sempre acreditou em mim.

Agradecimentos

Agradeço a todos os professores dos departamentos de Ciência da Computação e de Engenharia Elétrica, que me ensinaram os fundamentos necessários para a conclusão deste trabalho.

Em especial, agradeço a minha professora orientadora Carla Koike, que sempre me ofereceu suporte diante dos obstáculos enfretados.

Agradeço aos meus amigos de curso, que marcaram profundamente a minha trajetória pela UnB.

E, por fim, agradeço a minha família, que me deu todo o apoio necessário para a concretização desse momento.

Resumo

A área de *TinyML* constitui um campo emergente potencializado pelos conhecimentos estabelecidos com o estudo de *Internet of Things* (IoT) e *Machine Learning* (ML). Este trabalho detalha o desenvolvimento de uma solução *TinyML* enfatizando as necessidades e dificuldades inerentes ao desenvolvimento de aplicações embarcadas de aprendizado de máquina. Dessa maneira, é proposta uma arquitetura composta por sensores, controladores e luminárias de controle remoto para implementação de um projeto prático de automação, em que um espaço de testes é monitorado e tem sua iluminação manipulada por módulos de *software* alimentados de informações produzidas por modelos preditivos. A capacidade de aprendizado desse projeto é avaliada por meio de uma série experimentos associados ao retreino do algoritmo ML dado o surgimento de novos exemplos. No fim, a aplicação proposta expôs inteligência de controle, alterando o padrão de iluminação de acordo com a rotina demonstrada pelo usuário.

Palavras-chave: Internet of Things, TinyML, Hardware embarcado, Aprendizado de máquina, Iluminação

Abstract

The field of *TinyML* is an emerging research body enhanced by the accumulated knowledge built with the study of Internet of Things (IoT) and Machine Learning (ML). This work describes the development of a *TinyML* solution and emphasizes the different needs and difficulties inherent in developing of embedded machine learning applications. Therefore, an architecture composed of sensors, controllers, and remote control lighting is proposed as implementation of a practical automation project, where a test space is monitored and its lighting is manipulated by software modules fed with data produced by predictive models. The learning aspect of this application is evaluated through a series of experiments designed around the retraining step of the ML algorithm given the gathering of new information. In the end, the proposed application displays intelligent control, changing the lighting pattern according to the user routine.

Keywords: Internet of Things, TinyML, Embedded hardware, Machine Learning, Lighting

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AAEC *Activity-Appliance-Energy Consumption.*

ADC *Analog-to-digital converter.*

API *Application Programming Interfaces.*

DL *Deep Learning.*

DT *Decision Tree.*

GPC *Gaussian Process Classifier.*

GRU *Gated Recurrent Unit.*

HAR *Human Activity Recognition .*

IoT *Internet of Things.*

ISVM *Incremental Support Vector Machine.*

KNN *k-Nearest Neighbors.*

LDA *Linear Discriminant Analysis.*

LR *Logistic Regression.*

LSTM *Long short-term memory.*

ML *Machine Learning.*

MLP *Multilayer Perceptron.*

NB *Naive Bayes.*

PIR *Passive Infrared.*

RNA Rede Neural Artificial.

RNN *Recurrent Neural Network*.

SBC *Single Board Computer*.

SPEED *Sequence Prediction via Enhanced Episode Discover*.

SVM *Support Vector Machine*.

VFDT *Very Fast Decision Tree*.

Capítulo 1

Introdução

O campo de pesquisa de *Internet of Things* (IoT) investe na alta conectividade entre dispositivos como meio de fomentar soluções para problemáticas reais. De maneira concreta, essa premissa está sendo gradualmente comprovada com a expansão das aplicações IoT, que variam desde aparelhos domésticos inteligentes até o uso de ferramentas remotas para garantia do recebimento de cuidados médicos, como ocorreu no período da pandemia do COVID-19, em que o contato físico configurava uma situação de risco crítica [1].

Por outro lado, um dos clássicos desafios da área de IoT é a manutenção da privacidade de seus usuários. Considerando que projetos de IoT se comportam como uma rede de instrumentos por natureza, essas aplicações tornam-se suscetíveis a ataques cibernéticos, correndo o risco de exposição de dados sensíveis manipulados pelos seus respectivos sistemas. Nesse contexto, *Edge Computing* ou, em português, arquitetura de computação de borda, promove a aproximação dos serviços de processamento de dados a suas fontes de origem e surge como agente atenuador dessas deficiências [2].

Da mesma forma, o campo de aprendizado de máquina foi, recentemente, introduzido à pesquisa IoT com a inserção de modelos de predição a projetos de automação. Essa combinação se torna compreensível quando destacada a grande quantidade de dados disponíveis na maioria dos sistemas de automação. O projeto custeado pelo Ministério de Educação e Pesquisa Alemão, *OpenLicht* [3], é exemplo do poder dessa união ao propor um controle de iluminação extremamente dinâmico, capaz de inferir prefêrencias de residentes com auxílio de informações de sensores.

Assim, fruto da interdisciplinaridade, nasce o paradigma *TinyML*, o qual desfruta dos conhecimentos de IoT, *Machine Learning* e *Edge Computing* para o desenvolvimento de aplicações inteligentes e responsivas. Esse trabalho, então, inspira-se nesse novo panorama de projetos e propõe a montagem de uma aplicação de controle domiciliar que usufrua de recursos de aprendizado de máquina. Similar ao *OpenLicht*, o sistema projetado gerencia a iluminação de um ambiente enquanto aprende continuamente por meio de parâmetros

monitorados. Com este estudo, objetiva-se:

- Descrever o processo de elaboração e implementação de uma aplicação *TinyML*;
- Destacar os diferentes desafios inerentes a esse tipo de sistema; e
- Avaliar a dinamicidade do sistema proposto de acordo com o desempenho de seu aprendizado.

Este documento é divido em capítulos a serem detalhados em seguida. No Capítulo 2, são revisados conceitos teóricos fundamentais para o entendimento do projeto. No Capítulo 3, são verificados projetos da literatura envolvidos com a área de IoT e *Machine Learning*. No Capítulo 4, são explicitadas ferramentas de *hardware* e de *software* utilizadas para o desenvolvimento da aplicação. No Capítulo 5, são descritas as etapas para construção dos diferentes componentes do sistema. No Capítulo 6, são expostos os resultados produzidos. E, por fim, no Capítulo 7, resume-se o conteúdo explanado e apresenta-se possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta os conceitos teóricos fundamentais para o entendimento integral desse trabalho e as notações introduzidas no decorrer de seu texto são resumidas no Apêndice A.

2.1 Inteligência Artificial

McCarthy [4] compreende inteligência artificial como "a ciência e engenharia de construção de máquinas inteligentes, especialmente, a construção de programas de computador inteligentes", e afirma que "ela está relacionada a tarefa de utilizar computadores para entender a inteligência humana, embora não precise ser restringida por metódos biologicamente observáveis". Logo, de forma geral, o campo de inteligência artificial se preocupa não apenas com o entendimento de como se dá a formação de entidades inteligentes mas também em como aplicá-las de maneira efetiva e segura em diversos contextos [5].

Este trabalho usufrui de fundamentos de inteligência artificial a fim de melhorar o desempenho de uma aplicação de automação embarcada, explorando, assim, as vantagens da utilização de algoritmos inteligentes em comparação a regras previamente definidas.

2.2 Aprendizado de máquina

Aprendizado de máquina ou *Machine Learning* (ML) caracteriza-se como subcampo da pesquisa de inteligência artificial que estuda como programar computadores de maneira que aprendam com auxílio de dados [6]. Um fundamento de programas de ML é apresentar a capacidade de se desenvolver com experiências passadas, empregando um príncipio de inferência nomeado de indução, o qual, a partir de exemplos particulares, é capaz de produzir conclusões genéricas. Em síntese, algoritmos de ML resolvem problemas induzindo funções em cima de dados que refletem a situação analisada [7].

De maneira ainda mais prática, uma solução de ML pode ser descrita como um processo em duas etapas essenciais [8]:

1. Agrega-se um conjunto de dados; e
2. Elabora-se um modelo estatístico, com o uso de algoritmos, baseado no conjunto de dados adquirido.

Ao detalhar melhor os principais tipos de aprendizado em ML, encontra-se quatro categorias de aprendizado: o aprendizado supervisionado, o semi-supervisionado, o não supervisionado e o aprendizado por reforço. Neste trabalho é abordado, essencialmente, o aprendizado supervisionado. Por isso, a seguir são apresentadas explicações mais detalhadas dessa categoria, contrastando-a com o aprendizado não supervisionado.

2.2.1 Aprendizado supervisionado × não supervisionado

Em desafios de **aprendizado supervisionado**, um conjunto de dados é descrito por

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{N} , \tag{2.1}$$

onde o símbolo $\mathbf{x}_i$ representa o i-ésimo vetor de atributos de um conjunto de tamanho N e y_i, seu respectivo rótulo.

De maneira concreta, um vetor de atributos deve compreender características representativas do cenário a ser reproduzido pelo algoritmo ML. Da mesma forma, um rótulo configura a informação a ser predita pelo modelo, ou seja, a característica desconhecida a ser determinada a partir de outros atributos disponíveis. No geral, o rótulo refere-se a um elemento que faz parte de uma coleção finita de classes ou que representa um número real.

Considerando o que foi descrito e a Tabela 2.1 como exemplo de conjunto de dados, os valores de sensores da primeira linha da tabela podem ser interpretados como o vetor de atributos $\mathbf{x_1} = [1, 2]$, e a informação de luz, como o rótulo $y_1 = 1$, indicando, por exemplo, o estado aceso da lâmpada.

Tabela 2.1: Exemplo de conjunto de dados.

Sensor 1	**Sensor 2**	**Luz**
1	2	1
2	3	0
3	4	1

Por outro lado, no contexto de **aprendizado não supervisionado**, um conjunto de dados é descrito por

$$\{(\mathbf{x}_i)\}_{i=1}^{N} , \tag{2.2}$$

onde, novamente, $\mathbf{x}_i$ é um vetor de atributos de um conjunto de tamanho N, porém, dessa vez, não são denotados rótulos. Assim, é possível concluir que o aprendizado não supervisionado tem por característica o emprego de conjuntos de dados não rotulados para a resolução de problemas [8].

2.3 Classificadores

Uma das tarefas abordadas pelos algoritmos de aprendizado supervisionado é a classificação. Na Subseção 2.2.1, foi explicitada a possibilidade de um rótulo pertencer a um conjunto finito pré-estabelecido ou ao conjunto contínuo dos números reais. Quando o rótulo de uma base de dados pertence a um conjunto finito, entende-se que a problemática retratada refere-se a um desafio de classificação, e que o conjunto finito em questão descreve o conjunto de classes que poderão ser associadas a um novo exemplo. A seguir são apresentados alguns classificadores clássicos da área de aprendizado de máquina.

2.3.1 K-Vizinhos mais próximos

k-Nearest Neighbors (KNN), inicialmente introduzido por Fix e Hodges [9] e futuramente ampliado por Cover [10], é um algoritmo que classifica novos exemplos de um cenário comparando-os ao resto da coleção de dados disponível. Essa análise de semelhança se dá por funções de distância, sendo um exemplo clássico a distância Euclidiana, definida por

$$d(\mathbf{x}_i, \mathbf{x}_k) = \sqrt{\sum_{j=1}^{D} (x_i^{(j)} - x_k^{(j)})^2} , \tag{2.3}$$

onde $d(\mathbf{x}_i, \mathbf{x}_k)$ descreve a distância entre as amostras do conjunto de dados $\mathbf{x}_i$ e $\mathbf{x}_k$, ambas de dimensão D. Essa fórmula compara os atributos dos vetores $\mathbf{x}_i$ e $\mathbf{x}_k$ entre si, representados por $x_i^{(j)}$ e $x_k^{(j)}$, onde $j = 1, \cdots, D$.

No entanto, não é anormal verificar-se o emprego de outros tipos de equações de distância em diferentes implementações do KNN, como a de Chebychev ou a de Hamming. A partir dessa noção de similaridade o KNN classifica novos objetos, o algoritmo encontra os k exemplos mais próximos a esta amostra de acordo com a função de distância estabelecida e atribui a classe majoritária ao novo exemplo.

Uma característica importante de se notar a respeito do classificador KNN é a necessidade da manutenção de todo o conjunto de dados em memória para a predição de novas amostras [8].

2.3.2 Árvores de decisão

Algoritmos de *Decision Tree* (DT) [11] observam um problema de classificação como um problema de procura num espaço de soluções. Assim, a fim de solucionar desafios, uma árvore de decisão divide trabalhos maiores em trabalhos menores e repete essa técnica de maneira recursiva. Exemplos de algoritmos baseados em árvores de decisão são: ID3 [12], CART [13] e C4.5 [14].

Formalmente, uma DT compreende um gráfico acíclico direcionado, na qual cada nó que se ramifica estabelece uma condição ao examinar um atributo j em relação a um valor limite. Consequentemente, os nós terminais de uma DT instituem os vereditos acerca das classes que as amostras avaliadas farão parte. A Figura 2.1 explicita bem esse processo, apresentando o espaço de soluções designado pelos atributos $\mathbf{x}_1$ e $\mathbf{x}_2$ e os respectivos limiares a_1, a_2, a_3 e a_4 que dividem as regiões de decisão C_1, C_2, C_3, C_4 e C_5.

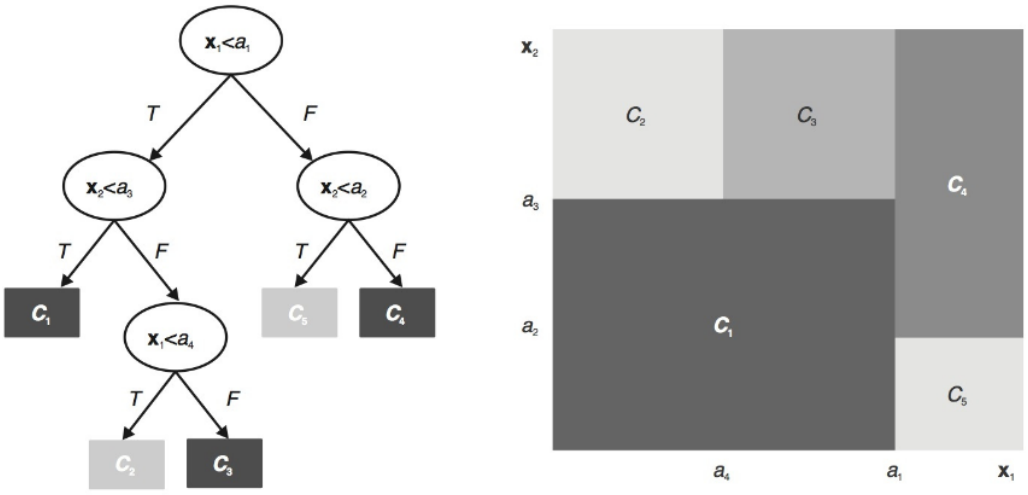


Figura 2.1: Visualização de uma arvóre de decisão. (Fonte: [7]).

2.3.3 Máquina de vetores de suporte

Um algoritmo de *Support Vector Machine* (SVM), proposto por Vapnik [15], visualiza cada um dos vetores de atributos de um conjunto de dados como um ponto localizado em um espaço com número de dimensões congruente ao tamanho D do vetor mencionado.

A classificação, então, ocorre ao traçar-se um hiperplano que divide o espaço introduzido entre classes positivas e negativas.

No contexto de SVMs lineares, isto é, SVMs que determinam fronteiras lineares para separação dos objetos de um conjunto, o hiperplano a ser obtido é descrito por

$$\mathbf{wx} - b = 0 \,, \tag{2.4}$$

em que $\mathbf{wx}$ retrata o produto escalar entre o vetor de pesos $\mathbf{w}$ e o vetor de atributos $\mathbf{x}$, e b expressa um número real. Da mesma forma, a predição de um vetor de entrada é dada por

$$y = \text{sign}(\mathbf{wx} - b) \,, \tag{2.5}$$

onde sign representa um operador matemático que produz as saídas $+1$ ou -1, e y, o rótulo resultante dessa predição, definido por classes associadas às saídas listadas.

Assim, o método SVM concebe um modelo de predição ao resolver um problema de otimização com restrições definidas. Mais especificamente, ele procura pelo classificador

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^*\mathbf{x} - b^*) \tag{2.6}$$

com os parâmetros ótimos $\mathbf{w}^*$ e b^* e sob as condições

$$\begin{cases} \mathbf{wx}_i - b \geq +1 & \text{se} \quad y_i = +1 \,, \text{ e} \\ \mathbf{wx}_i - b \leq -1 & \text{se} \quad y_i = -1 \,. \end{cases} \tag{2.7}$$

O problema de otimização supracitado refere-se a minimização da distância entre as amostras de classes opostas mais próximas no espaço trabalhado, ou seja, a minimização da margem do hiperplano a ser estipulado.

A Figura 2.2 ilustra o resultado da classificação para um vetor de atributos com apenas duas dimensões. Observa-se que, nessa situação, o hiperplano ótimo é descrito por uma reta.

2.3.4 Regressão logística

O classificador de regressão logística, do inglês *Logistic Regression* (LR), assim como o exemplo anterior, modela o seu valor de saída y como função linear dependente de $\mathbf{x}$. Entretanto, o algoritmo binário de LR adota o valor 0 para sua classe negativa e o valor 1 para sua classe positiva. Dessa maneira, buscando concordância com esse preceito, é aplicada a função sigmoide

$$g(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

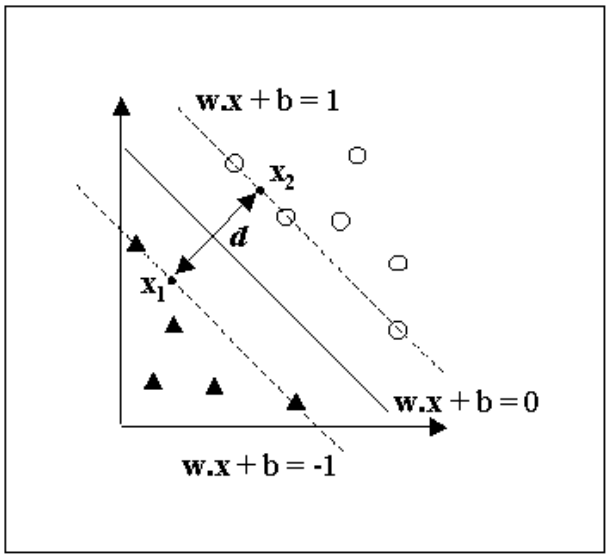


Figura 2.2: Hiperplano ótimo em duas dimensões (Fonte: [16]).

de contradomínio igual a $(0, 1)$ para obter-se o classificador

$$f(\mathbf{x}) := \frac{1}{1 + e^{-\mathbf{wx}+b}} \, , \tag{2.9}$$

onde, mais uma vez, $\mathbf{x}$ é um vetor de atributos, $\mathbf{w}$ é um vetor de pesos e b é um número real.

Nota-se, no entanto, que a função sigmoide é contínua e, portanto, abrange mais valores que apenas o 0 e o 1. Por isso, é utilizado um limiar que determina os intervalos correspondentes a cada uma das classes, por exemplo, valores menores que 0,5 são negativos e o restante, positivos [6].

2.4 Redes Neurais

Em 1943, os estudiosos McCulloch and Pitts, na tentativa de simular neurônios através de modelos matemáticos, publicaram os primeiros trabalhos do campo de pesquisa de redes neurais [17]. Ainda que essas primeiras redes propostas não apresentassem a habilidade de aprendizado, a sua investigação assinalou o grande poder computacional presente na combinação de unidades lógicas.

Similarmente, as redes neurais artificiais (RNAs) também são compostas por unidades atômicas de processamento que performam uma função matemática. Mais precisamente, esses neurônios calculam a soma ponderada entre o vetor de atributos recebido como entrada e os pesos referentes a cada uma de suas conexões. Por fim, o resultado desse cálculo é alimentado a uma função de ativação que estabelece o comportamento dessa unidade

matemática. Neste trabalho são abordadas a função de ativação sigmoide anteriormente introduzida (Equação 2.8) e a função de ativação ReLU, descrita por

$$g(x) = \begin{cases} x & \text{se} \quad x > 0, \text{ e} \\ 0 & \text{se} \quad x \leq 0. \end{cases} \tag{2.10}$$

Em contextos de aprendizado supervisionado, o aspecto inteligente de uma RNA é implementado por intermédio de um algoritmo de correção de erro, o qual gradualmente ajusta os pesos mencionados a fim de minimizar o erro entre os rótulos preditos e seus contrapontos reais.

2.4.1 Perceptron Multicamadas

O método *Multilayer Perceptron* (MLP) é caracterizado por uma RNA com múltiplas camadas de neurônios. Essas camadas são classificadas entre camadas intermediárias e de saída. Por natureza, as camadas intermediárias de uma MLP utilizam funções de ativação não lineares e são completamente conectadas, nas quais cada um dos nós de certa camada conecta-se a todos os outros nós pertencentes à camada anterior e a seguinte.

Em problemas de classificação, a camada de saída de uma MLP apresenta número de nós igual ao número de classes possíveis do seu conjunto de dados. Assim, um novo objeto é predito como pertencente a certa classe quando seu respectivo neurônio exibe o valor de saída mais alto. A Figura 2.3 ilustra uma rede MLP.

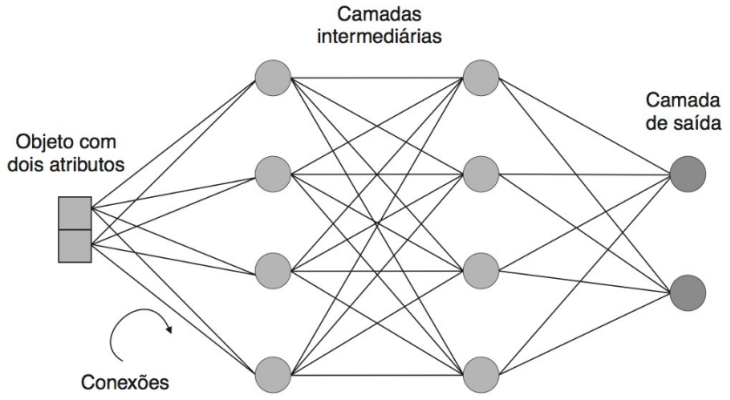


Figura 2.3: Ilustração de uma rede perceptron multicamadas. (Fonte: [7]).

2.5 Aprendizado incremental

O aprendizado incremental descreve um paradigma de aprendizado de máquina na qual um modelo amplia seu conhecimento ao se alimentar de dados recebidos de maneira contínua. Esse contexto diverge do cenário usual de aprendizado em *batch* que conta com a presença de todos os dados antes de se iniciar o treinamento. Em geral, o aprendizado incremental é implementado em sistemas com restrições de espaço de memória ou em ambientes em constante mudança [18]. Alguns exemplos de algoritmos clássicos adaptados para o aprendizado incremental são o ID4 [19], baseado em árvores de decisão, e o *Incremental Support Vector Machine* (ISVM) [20].

2.6 Normalização

A técnica de normalização mapeia uma faixa de valores assumida por certa característica para o intervalo real $[0, 1]$ utilizando a Equação 2.11. Nesse cálculo, $min^{(j)}$ refere-se ao menor valor designado ao atributo entre a coleção de dados e $max^{(j)}$, ao seu maior valor. Em geral, esse método é aplicado com objetivo de melhorar a performance do modelo ML durante seu treinamento [6].

$$\overline{x}^{(j)} = \frac{x^{(j)} - min^{(j)}}{max^{(j)} - min^{(j)}} \tag{2.11}$$

2.7 Métricas de avaliação de desempenho

Ainda mais importante do que implementação de um modelo de ML é a avaliação de seu desempenho. Por isso, com intuito de mensurar a performance de diferentes métodos e até, eventualmente, compará-los, são utilizadas diversas métricas de desempenho.

2.7.1 Matriz de confusão

A matriz de confusão é uma tabela que compara os resultados de um modelo de classificação aos rótulos esperados. Nesse tipo de visualização um dos seus eixos refere-se à classe predita pelo modelo e o outro, à classe real da amostra. Uma matriz de confusão pode ter inúmeras linhas e colunas de acordo com a quantidade de classes empregadas no problema.

Na Figura 2.4 observa-se uma matriz de confusão binária, exibindo apenas as classes positiva e negativa. Nessa imagem, também é possível verificar os seguintes elementos:

- Verdadeiros Positivos (VP): número de exemplos da classe positiva corretamente classificados como positivos;
- Verdadeiros Negativos (VN): número de exemplos da classe negativa corretamente classificados como negativos;
- Falsos Positivos (FP): número de exemplos da classe negativa incorretamente classificados como positivos; e
- Falsos Negativos (FN): número de exemplos da classe positiva incorretamente classificados como negativos.

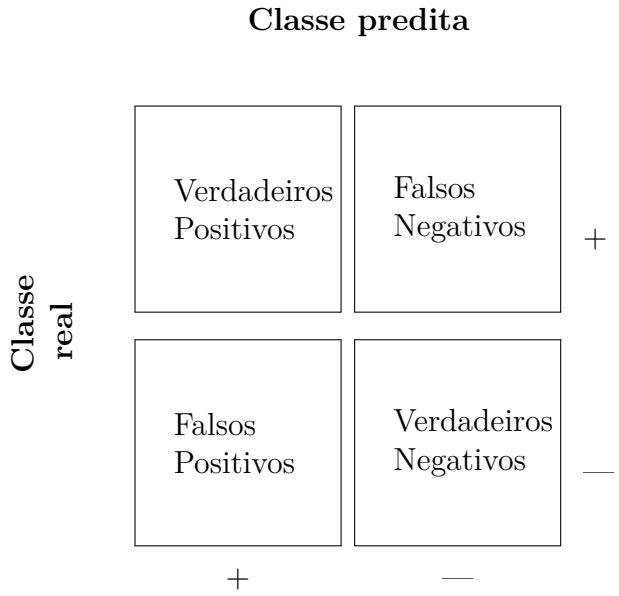


Figura 2.4: Matriz de confusão de duas classes.

2.7.2 Acurácia

A acurácia, demonstrada pela Equação 2.12, é calculada pela razão entre amostras com rótulos preditos corretamente e o total de amostras. Essa métrica é mais eficaz quando o erro de predição de todas as classes é igualmente relevante.

$$\text{Acurácia} := \frac{VP + VN}{VP + VN + FP + FN} \tag{2.12}$$

2.7.3 Precisão

A precisão (Equação 2.13) refere-se a razão entre as predições corretas da classe positiva e o número total de predições positivas. A sua análise faz sentido em cenários no qual a predição positiva errônea é considerada bastante custosa.

$$\text{Precisão} := \frac{VP}{VP + VN} \tag{2.13}$$

2.7.4 Revocação

A revocação (Equação 2.13), do inglês *recall*, determina a razão entre o número de exemplos preditos como positivos e o número total de amostras positivas. De maneira oposta, essa métrica é principalmente empregada quando entende-se como mais prejudicial a predição negativa incorreta [8].

$$\text{Revocação} := \frac{VP}{VP + FN} \tag{2.14}$$

2.7.5 F1-Score

O F1-score é determinado pela média harmônica entre as métricas de precisão e revocação. Pela Equação 2.15, verifica-se que o valor de F1-score é alto quando a precisão e a revocação também forem [6].

$$\text{F}_1 := 2 \times \frac{\text{precisão} \times \text{revocação}}{\text{precisão} + \text{revocação}} \tag{2.15}$$

2.8 Reconhecimento de Atividade Humana

Human Activity Recognition (HAR) abrange uma área de pesquisa que utiliza dados brutos provenientes de dispositivos e modelos de inteligência artificial para identificar e rotular diferentes atividades. O estudo de HAR é fomentado pela sua gama de aplicações reais, dominando a área de saúde e de cuidados com idosos [21]. No contexto de segurança e monitoramento, por exemplo, HAR pode ser implementado para identificação de atividades anômalas ou suspeitas. Além disso, também pode ser usado para reconhecimento de atividades de motoristas dentro de carros autônomos e pode ser aplicado em *smart homes* para a garantia do bem estar de ocupantes [22].

O fluxo comum de uma aplicação de HAR, ilustrado pela Figura 2.5, conta com

1. Captura de sinais;
2. Pré-processamento;

3. Treinamento do modelo; e

4. Predição de atividade.

A primeira etapa, a coleta de dados, é realizada por meio de dispositivos divididos entre visuais e sensores. Dispositivos visuais como câmeras são naturalmente considerados danosos à privacidade do usuário e necessitam de maior poder de processamento. Os dispositivos sensores, por outro lado, são menos invasivos e apresentam uma grande variedade. Em projetos de HAR, são verificados sensores que mensuram aceleração, temperatura, pressão, velocidade angular, entre outras medidas [21, 23, 22].

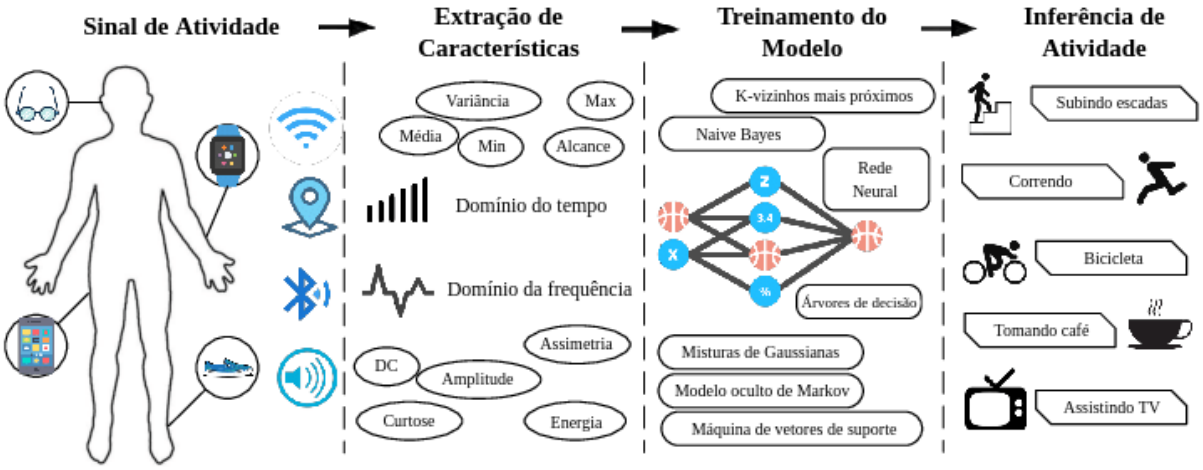


Figura 2.5: Fluxo de HAR (Imagem adaptada) (Fonte: [24]).

Em relação aos modelos típicos de problemas HAR, repara-se tanto o emprego de técnicas de ML clássicas como o uso de alternativas baseadas em redes neurais. Os algoritmos de DT, KNN e SVM são exemplos de modelos clássicos de ML comumente implementados como soluções de HAR. Do outro lado, com o uso de redes neurais, o catálogo de algoritmos se expande bastante, abrangendo tanto modelos convolucionais quanto recorrentes, sendo a sua escolha intrinsicamente conectada à natureza dos dados a serem consumidos [25].

Capítulo 3

Revisão de Literatura

Neste capítulo são revisadas as diferentes aparições da fusão de métodos de inteligência artificial a sistemas coletores de sinais na literatura. Nessa coleção de trabalhos é possível verificar divergências entre modelos, sensores, configurações e até mesmo objetivos finais, porém, de maneira geral, todos eles contribuem com o estudo da aplicação de princípios de IoT e inteligência artificial na resolução de uma problemática.

Em seu artigo, Bierzynski *et al.* [3] introduzem o projeto *OpenLicht*, o qual procura implantar um sistema inteligente de iluminação capaz de se adpatar às atividades e preferências do seu usuário. Um dos requisitos principais do *OpenLicht* é a centralização do processamento na borda da rede a fim de garantir maior privacidade para o residente. Dessa maneira, a configuração de *hardware* proposta conta com um *gateway* responsável pelo controle e processamento de dados. Essa controladora conecta-se com os nodos de sensores e luz através de inúmeras interfaces sem fio, como, por exemplo, *ZigBee*, *Z-Wave*, e *Bluetooth*. O sistema foi testado em um quarto real com diversos sensores e o *gateway* customizado.

No projeto, Bierzynski *et al.* [3] modelam a automação das luzes do ambiente como um problema da HAR (Seção 2.8), associando certa preferência de iluminação à atividade realizada pelo usuário. O fluxo inteligente inicia com o pré-processamento dos dados não tratados, os quais são, então, repassados para um *voting classifier* composto por múltiplas redes neurais e árvores de decisão. Assim, as predições produzidas e os sinais capturados pelos sensores são utilizados por modelos de *Deep Q-Learning* para definir a cor, luminosidade e temperatura das lâmpadas. Por fim, a arquitetura verifica constantemente a acurácia dos modelos citados com intuito de retreiná-los no cenário em que essa métrica decaia, uma possibilidade gerada por diferenças graduais nos estados dos sensores.

Também em relação à resolução de desafios de HAR, Kolkar *et tal.* [26] comparam diferentes técnicas de *Deep Learning* (DL) e propõe uma nova arquitetura do tipo *Gated Recurrent Unit* (GRU) para predição de atividades. Para isso, os autores recorrem a dois

conjuntos de dados, o UCI-HAR [27] e o WISDM [28], ambos confeccionados coletando-se valores de acelaração e velocidade angular capturados por celulares adjuntos ao corpo do usuário. Tanto o conjunto UCI-HAR quanto o conjunto WISDM trabalham com as mesmas 6 classes de atividades. A arquitetura proposta pelos autores, ilustrada pela Figura 3.1, utiliza camadas GRU para resolver o problema de dissipação de gradiente, o qual é frequentemente observado em redes neurais recorrentes, do inglês, *Recurrent Neural Networks* (RNNs) . Essa nova rede demonstrou performance superior em relação a outras arquiteturas RNN em ambos os conjuntos de dados, com acurácia média de 96,83% para o conjunto UCI-HAR e de 98% para o conjunto WISDM.

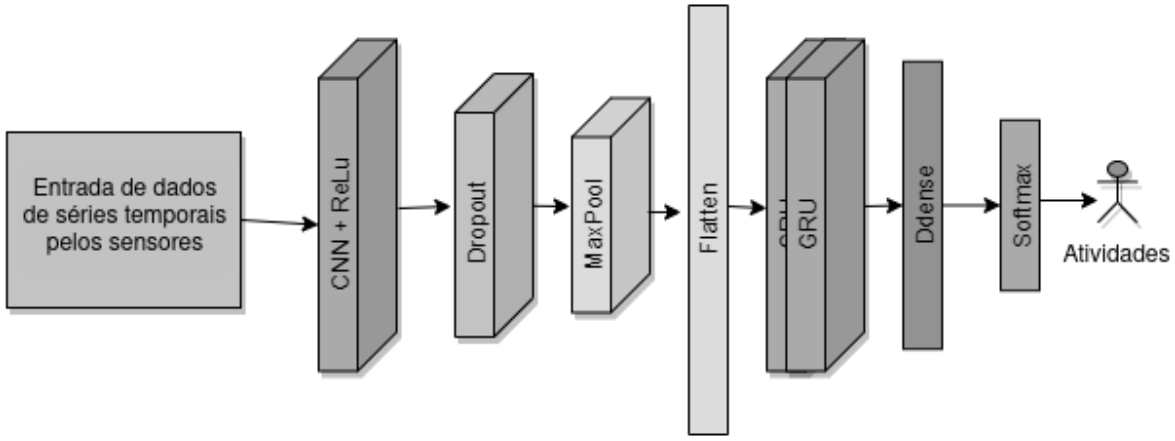


Figura 3.1: Arquitetura da rede neural GRU (Imagem adaptada) (Fonte: [28]).

Lima *et al.* [29] apresentam um novo método denominado *Activity-Appliance-Energy Consumption* (AAEC), que busca reduzir consumos de energia domiciliares associando atividades de um residente ao uso de certos eletrodomésticos. O método AAEC tem como estágio inicial uma problemática de HAR. A escolha do algoritmo ideal para essa fase foi avaliada com um conjunto de dados do MIT [30], construído a partir do monitoramento de sensores customizados instalados em objetos de interação diária. Esse conjunto apresenta a informação de ativação desses objetos, isto é, se eles são usados ou não, e suas atividades correspondentes. Com 99,96% de acurácia média, o modelo *Random Tree* foi escolhido para essa etapa do método AAEC.

Desapegando-se de técnicas de *Deep Learning*, Suryadevara *et al.* [31] avaliam um sistema de acompanhamento de idosos baseados em funções de bem-estar. Fisicamente, esse sistema contempla múltiplos dispositivos fixados a aparelhos de um ambiente. Esses dispositivos foram desenhados conectando sensores mensuradores de contato, corrente

elétrica e pressão a um módulo *ZigBee* de comunicação sem fio. Efetivamente, essas unidades capturam o estado dos seus respectivos utensílios, determinando se estão ativos ou não. Além disso, foi instalado um módulo especial capaz de mensurar valores de temperatura, umidade e luminosidade. Assim, todas essas medidas são centralizadas em um *ZigBee* coordenador, também responsável pelo envio desses dados para processamento em um computador.

A fim de determinar o bem-estar do residente observado, Suryadevara *et al.* [31] guardam o tempo de uso dos objetos e a atividade que foi identificada nesse período. A partir desses dados, os autores exibem duas funções matématicas que verificam se o padrão de uso dos aparelhos difere do que é experimentalmente considerado normal. A primeira função confere a inatividade dos aparelhos e a segunda função, o excesso de atividade.

Similarmente, Marufuzzaman *et al.* [32] dedicam-se à previsão de atividades em *smart homes*. Para isso, os autores modificam o algoritmo de *Sequence Prediction via Enhanced Episode Discover* (SPEED) [33], que introduz a noção de episódios, definidos como o intervalo de ínicio ao fim de execução de uma tarefa, e utiliza-os para composição de uma *Decision Tree*. Na sua versão modificada, os autores aproveitam dos dados de horário e de localização de ativação de um dispositivo com objetivo de aumentar a acurácia e a velocidade do modelo. Efetivamente, os resultados foram positivos, demonstrando valor de acurácia de 90%, uma grande melhora quando comparado aos 30% de acurácia do método original. As alterações também aperfeiçoaram o tempo de execução do algoritmo, medindo 94,5 segundos com o novo modelo e 28703,4 segundos com o SPEED clássico para predição de 689 amostras.

Os resultados descritos por Marufuzzaman *et al.* [32] foram calculados empregando a base de dados MavLab [34] do projeto MavHome [35]. Esse conjunto de dados compreende os estados de inúmeros dispositivos espalhados pelo ambiente MavHome, a data e hora dos eventos de captura e o cômodo em que se encontra o aparelho. Os autores testaram o algoritmo em um protótipo de *hardware* utilizando LEDs, chaves e um **Raspberry Pi 2B**. Nesse experimento, uma chave é inicialmente configurada em modo de entrada e os estados dos LEDs são armazenados, em seguida, esse modo é trocado para saída e o **Raspberry Pi 2B** toma controle dos estados dos LEDS.

Ampliando a pesquisa sobre iluminação inteligente, Dinata *et al.* [36] introduzem melhorias para o classificador *Very Fast Decision Tree* (VFDT), o qual, por sua vez, destaca-se no campo de sistemas dinâmicos por apresentar vantagens em contextos de aprendizado de máquina *online*. Dinata *et al.* [36] usam a técnica de estimativa de densidade kernel ao invés de assumir uma densidade gaussiana no algoritmo de VFDT usual e refina sua correção de Laplace com intuito de incrementar sua acurácia. Os

autores utilizam a base de dados KYOTO do projeto CASAS [37] com apenas informações de horário, de sensores de movimento, temperatura e luminosidade, e de interruptores de luz, totalizando 44 atributos e 1 rótulo. Em síntese, o VFDT reformado supera o resultado do modelo padrão tanto nos experimentos de aprendizado *online* quanto *offline*, exibindo acurácia média com validação cruzada de 10 *folds* de 98,74% *offline* e 99,03% *online*.

Pala *et al.* [38] também aludem ao controle de luzes em uma habitação projetando uma aplicação capaz de simular a presença de residentes com a finalidade de evitar furtos. A arquitetura estabelecida dispõe de um microcontrolador **Arduino Mega** e de relês para a coleta das condições dos diversos pontos de luz distribuídos pela casa. Com esse sistema, informações a respeito do nível de iluminação em cada cômodo foram registradas durante o período noturno, gerando uma base de dados com 5500 amostras, 9 atributos e 1 rótulo.

Os dados reunidos por Pala *et al.* [38] são processados em um computador por diversos algoritmos clássicos de aprendizado de máquina: *Naive Bayes* (NB), *Linear Discriminant Analysis* (LDA), MLP, KNN, SVM e DT. No fim, os resultados de acurácia média utilizando validação cruzada de 10 *folds* foram: 96,69% para MLP, 94,98% para SVM, 91,23% para NB, 87,54% para LDA, 74,18% para DT e 64,29% para KNN.

Ainda no contexto de *smart homes*, Gven *et al.* [39] projetam e implementa um sistema de automação que usa aprendizado de máquina para performar tarefas de forma independente. A configuração proposta pelos autores dispõe de um **Raspberry Pi 3 Model B+** para execução dos algoritmos e de um **Arduino Nano** para fácil comunicação com os sensores, os quais variam entre mensuradores de luminosidade, de temperatura, de umidade (**DHT11**), de movimento, de gás inflamável, de chuva e de chama. A partir desse sistema, uma base de dados foi concebida com os valores coletados dos sensores, com a data e o horário da captura, e com as informações a respeito de módulos atuadores, como lâmpadas, ventiladores e servo motores. Assim, foi construído um conjunto de dados com 18 atributos e 17531 amostras.

Gven *et al.* [39] utilizam essa base de dados para testar 15 algoritmos de aprendizado de máquina diferentes. No final, as métricas superiores, calculadas com validação cruzada de 10 *folds*, pertenceram ao método *Gaussian Process Classifier* (GPC), que exibiu 97% de acurácia, 98% de *recall*, 97% de *f1-score* e 97% de precisão.

Majeed *et al.* [40] apresentam um sistema similar ao supracitado e aborda desafios de segurança em relação à autentificação de dispositivos pela arquitetura. Nesse trabalho, Majeed *et al.* [40] exibem uma lista de componentes de *hardware* que contém, entre inúmeros itens, um **Raspberry Pi 2B**, um módulo de relês de 8 canais, um sensor de temperatura **DS18B20**, um sensor de luminosidade **LDR LM393** e um sensor de fumaça e gás inflamável **MQ2**. O computador Raspberry Pi implementa um servidor que processa e armazena mudanças de estado de dispositivos e atualiza remotamente

uma *Microsoft Azure Cloud Database*, o qual pode ser acessada fora da rede local. Os dispositivos mencionados, ao tentar conectar-se ao servidor, passam por uma etapa de autenticação por tecnologia *Blockchain*.

Para predição e controle inteligente desses dispositivos, Majeed *et al.* [40] utilizam uma base de dados composta pelos valores dos sensores enumerados anteriormente combinados à informação de estado de uma lâmpada. Um classificador SVM com 95,73% de acurácia foi selecionado para performar as tarefas de predição.

Salhi *et al.* [41] propõem uma aplicação que sinaliza riscos de incêndios e vazamentos de gás em residências. Sua arquitetura utiliza um **Arduino Uno R3** para agregação de dados de sensores enviados via protocolo *Zigbee* para um **Raspberry Pi 3 Model B**, o qual implementa uma interface gráfica e repassa os dados recebidos para uma camada superior de aplicação por meio do protocolo MQTT. Além disso, outro **Arduino Uno R3** é usado para simular respostas a riscos detectados. Esse protótipo, que conta com sensores de gás carbônico **MG-811**, de chama **LM35**, mais os conhecidos **DHT11** e **MQ2**, coletou 21.146 amostras rotuladas com 4 diferentes classes que reprentam o grau de perigo do ambiente. Esses dados alimentaram o experimento de múltiplos modelos de aprendizado, e o melhor resultado verificado foi da técnica de *Classification and Regression Trees* (CART), que exibiu acurácia média de 99,93% com validação cruzada de 10 *folds*.

Taiwo *et al.* [42] apresentam o iHOCS, um sistema inteligente de controle domiciliar e segurança. O iHOCS emprega o microcontrolador **ESP8266 Wi-Fi Development Board** como instrumento multiagente, o qual conecta dispositos da rede entre si, enquanto, ao mesmo tempo, coleta sinais de sensores e controla relês associados a aparelhos da habitação. Um dos sensores com uso particular no sistema iHOCS é a placa **ESP32-Cam**, responsável por capturar imagens quando detectado algum tipo de movimento. As imagens obtidas são analisadas por um classificador SVM treinado a partir de uma pequena base de dados com 20 imagens de humanos e animais, divididos entre duas classes de residentes e invasores. Ao final do experimento, o modelo SVM demonstrou acurácia de 80% com validação cruzada de 5 *folds*.

Afastando-se dos desafios de automação e controle, Casaccia *et al.* [43] focam na relação entre dados domóticos, isto é, dados mensurados por sensores, e conhecimentos subjetivos, como o bem-estar de um usuário. Para coleta desses dados objetivos, Casaccia *et al* [43] acoplam vários sensores de movimento e luminosidade junto a termostatos em casas reais de participantes do estudo. Enquanto isso, para coleta dos elementos subjetivos, os autores utilizam questionários a serem respondidos pelos residentes sobre noções gerais de saúde física, saúde mental e presença no ambiente. Os autores especificam o uso dos algoritmos de *Regression Tree* e *Regression Forest* para análise de correlação entre as informações domóticas e os níveis de bem-estar. Para isso, foram comparados

os dados coletados pelos questionários e as predições fornecidas pelos modelos. Após os testes realizados, o classificador *Regression Tree* exibiu 40% de erro médio absoluto, e o classificador *Regression Forest*, 32%.

Lentzas *et al* [44], por outro lado, divergem no emprego de sensores e se beneficia de dados de uso elétrico para o reconhecimento de ausência de residentes em um domicílio. Em seu trabalho, Lentzas *et al* [44] fazem uso das informações da base de dados UK-DALE [45] a respeito de 4 eletrodomésticos: televisão, chaleira elétrica, fogão e microondas. No entanto, como o conjunto UK-DALE não conta com atributos que indiquem a presença de habitantes, os autores performaram uma etapa de anotação manual que considerava uma rotina simulada. Os dados, então, foram testados por diferentes modelos de aprendizado de máquina, e o classificador MLP se sobressaiu no experimento de uma única execução, apresentando acurácia média de 98,2%.

Da mesma maneira, Solatidehkordi *et al* trabalham com informações de corrente e tensão elétricas de inúmeros dispositivos. Dessa vez, no entanto, o intuito é classificar o tipo do aparelho pelo seu padrão de consumo. Para isso, os autores usam uma arquitetura *Long short-term memory* (LSTM) treinada utilizando o conjunto de dados PLAID [46]. A rede é testada em um protótipo de *hardware* que consiste em um **Raspberry Pi** e diversos medidores inteligentes de energia. Nesse sistema, o **Raspberry Pi** coleta os dados dos medidores, executa o algoritmo LSTM e classifica cada um dos subcanais dos medidores. Os autores expõem as médias das métricas de precisão, *recall* e *f1-score* da rede LSTM como sendo 89,56%, 91,35% e 90.38%, respectivamente.

A Tabela 3.1 resume as referências retratadas nesse capítulo, destacando os algoritmos de ML examinados em cada trabalho. Na tabela, também é assinalado se a obra explicita algum tipo de sistema de *hardware* e, caso explicite, descreve a natureza do componente de aprendizado dentro da aplicação, dando ênfase no tipo de máquina em que o algoritmo de ML é executado.

Tabela 3.1: Lista de Referências

Referência	**Algoritmos de ML**	**Protótipo de *hardware*?**	**ML embarcado?**
[31]	Não utiliza algoritmos ML	Sim	N/A
[38]	KNN, DT, LDA, GNB, L-SVM, MLP	Sim	Não
[36]	NB, ANN, C4.5, VFDT, VFDT++	Não	N/A
[41]	LR, LDA, KNN, CART, NB, SVM	Sim	Não
[40]	SVM, KNN, RF, DF	Sim	Sim
[26]	GRU, CNN, LSTM, Hybrid CNN-LSTM, CNN-LSTM-Dense, Stacked-LSTM, Bidirectional-LSTM, Res-Bidirectional LSTM	Não	N/A
[29]	LR, NB, Bayes Network, Random Tree, Random forest, Best-First Decision Tree, J48, IBk, Kstar Beta Verion, Decision Table , SMO	Não	N/A
[42]	SVM, KNN, DT	Sim	Não
[44]	DT, C4.5, RF, NB, MLP, DNN	Não	N/A
[47]	LSTM	Sim	Sim
[43]	Regression Tree, Regression Forest	Sim	Não
[39]	LR, KNN, SVM, DT, XGBoost, MLP, RF, SGD, GPC, PAC, GNB, BNB, MNB, CNB	Sim	Sim
[32]	SPEED, SPEED modificado	Sim	Sim
[3]	Voting Classifier	Sim	Sim

Capítulo 4

Ferramentas

Neste capítulo são expostas as diferentes ferramentas de *software* e *hardware* que integraram esse trabalho. Enquanto seu uso concreto dentro do projeto é discutido apenas no Capítulo 5, esse capítulo busca introduzir artefatos fundamentais para a aplicação e comentar características ponderadas durante suas escolhas.

4.1 Hardware

A seguir são apresentados componentes de *hardware* empregados na construção do sistema de automação.

4.1.1 Computadores de Placa Única

O termo *Single Board Computer* (SBC) descreve computadores idealizados e construídos em uma única placa de circuito. Em geral, esses circuitos apresentam um processador de baixo custo, uma memória e diversas interfaces de entrada e saída. No catálogo de SBCs são encontrados tanto aparelhos que executam versões embarcadas de sistemas operacionais, como Linux e Windows, quanto peças mais simples, concebidas com microprocessadores programáveis. Além disso, usualmente, os SBCs não dispõem de *slots* de expansão capazes de aprimorar ou integrar componentes como é visto em outros modelos de computadores.

Sem grande rigor, pode ser dito que SBCs são dispositivos desenvolvidos e utilizados como ferramentas experimentais, perfeitas para provas de conceito. Afinal, esses computadores disponibilizam interfaces ideiais para captura de informações a partir de sensores, às vezes, até mesmo integrando-os à placa central.

A família Raspberry Pi é uma famosa linha de equipamentos composta de SBCs e acessórios de baixo custo. Em 2006, pesquisadores da Universidade de Cambridge inicia-

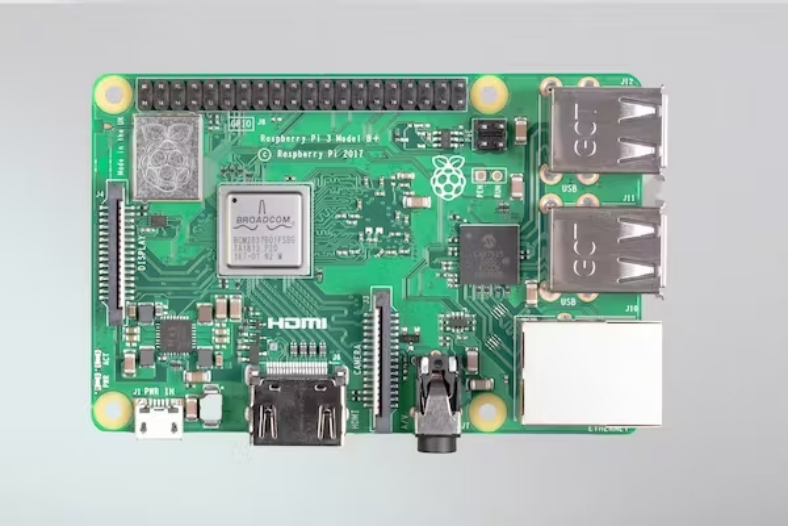


Figura 4.1: Raspberry Pi 3 Model B+ [1].

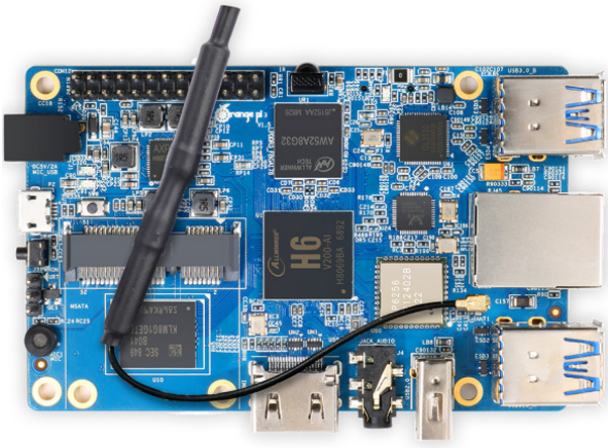


Figura 4.2: Orange Pi 3 [2].

ram os empenhos para seu desenvolvimento com intuito de criar um pequeno computador financeiramente acessível para crianças. A fundação Raspberry Pi disponibiliza um sistema operacional gratuito denominado Raspbian, baseado no sistema Linux Debian e otimizado para o *hardware* de sua placa [48]. A Figura 4.1 ilustra o modelo Raspberry Pi 3 Model B+.

Em vista do sucesso do Raspberry Pi, emergiram-se outras placas SBCs inspiradas no seu projeto, algumas caracterizando alternativas open-source, como o Orange Pi e o Banana Pi. O Orange Pi, ilustrado pela Figura 4.2, apresenta o benefício do suporte a

[1]Fonte: `https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/`

[2]Fonte: `http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-3.html`

diversos sistemas operacionais, como, por exemplo, Raspbian, Android, Ubuntu e Debian [48]. Enquanto o Banana Pi é conhecido por aprimorar as especificações dos seus modelos originadores, focando no aumento de sua performance [49].

4.1.2 Sensores

Sensores compreendem dispositivos pertencentes a sistemas de medição que são afetados por fenômenos físicos ou químicos. Quando um módulo sensor é capaz de converter uma grandeza de entrada em uma grandeza de saída com natureza diferente à inicial, entende-se que esse instrumento também integra um transdutor em seu circuito [50]. Em aplicações IoT, esses dispositivos configuram componentes cruciais para os sistemas propostos, agregando dados fundamentais para as regras de automação estabelecidas. Além disso, as diferentes categorias que permeiam o campo de IoT determinam uma enorme diversidade no emprego de sensores, o que é ilustrado pela Figura 4.3. Abaixo são discutidas algumas classes de sensores que aparecem nas arquiteturas de IoT [51].

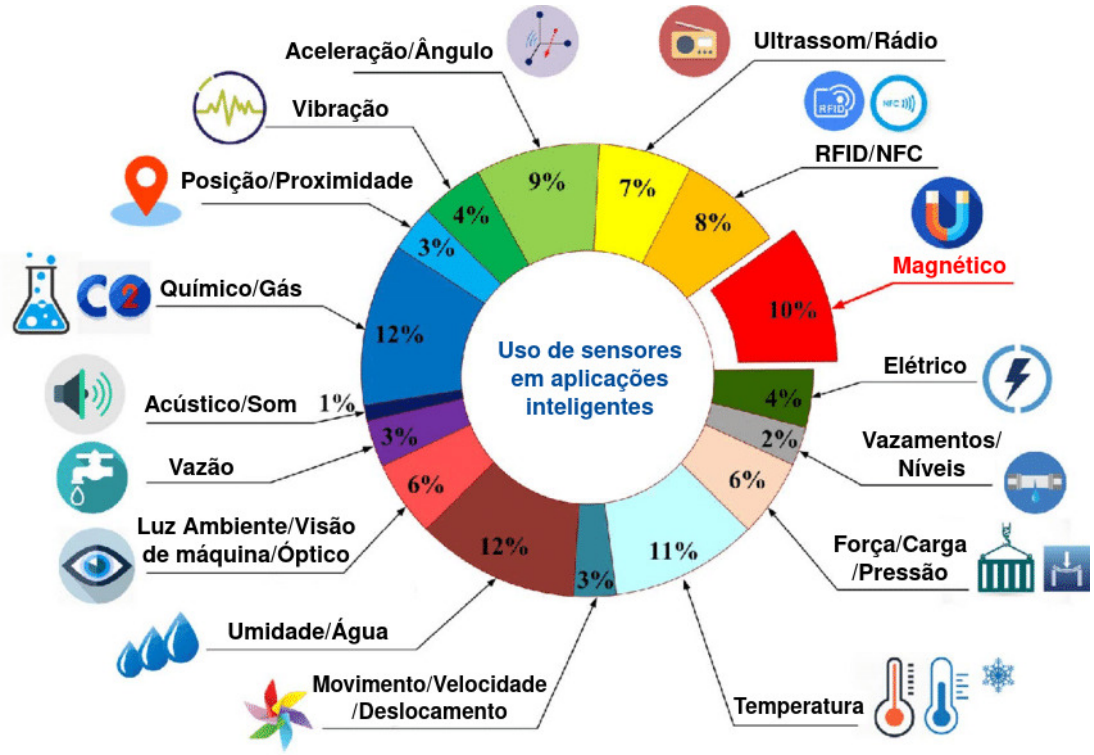


Figura 4.3: Uso de sensores no campo de IoT (Imagem adaptada) (Fonte: [52]).

Sensores de movimento

São sensores utilizados para captura de deslocamento em um ambiente. Neste trabalho, são explorados sensores de movimento *Passive Infrared* (PIR), que mensuram a radiação

infravermelha de corpos próximos ao seu campo de visão e produzem um sinal elétrico como reposta. De certa maneira, esses módulos também podem ser instalados como sensores de posição em um espaço monitorado.

Sensores de temperatura

São dispositivos que mensuram a energia térmica de certo corpo ou espaço. O sensor de temperatura empregado neste trabalho usa um termistor, um componente de resistência variável de acordo com as condições de temperatura, para aferição dessa grandeza.

Sensores de umidade

São sensores medidores da quantidade de água, normalmente no estado gasoso, em um ambiente. Neste projeto, o sensor de umidade escolhido utiliza um elemento capacitivo para medição, composto por duas placas de metal e um dielétrico poroso que varia de acordo com a presença de vapor d'água.

Sensores ópticos

Uma categoria de sensores que detectam ou medem níveis de energia eletromagnética. Neste trabalho, são empregados sensores ópticos baseados em fotoresistores, resistores com resistência variável em função da intensidade de luz, para detecção de luz ambiente.

Sensores de som

São sensores que detectam ou mensuram a intensidade de ondas sonoras. Neste projeto, é empregado um sensor sonoro que utiliza um microfone de eletreto para transformação das ondas acústicas em sinal elétrico.

4.1.3 Lâmpadas inteligentes

As lâmpadas inteligentes diferem das lâmpadas comuns incluindo microprocessadores ao seus circuitos elétricos. Dessa maneira, são criados dispositivos responsivos, que são capazes de comunicar com outros aparelhos em um ambiente. Geralmente, as lâmpadas inteligentes conectam-se a celulares por uma variedade de protocolos, dentre eles, o *Bluetooth*, o *Zigbee* e o *Wi-Fi*, e possibilitam a mudança de seus estados por meio de aplicativos móveis. Essas interfaces oferecem inúmeros serviços para o usuário, como a alteração de intensidade e temperatura da cor da lâmpada e a ativação ou desativação remota do dispositivo. A Figura 4.4 exemplica uma lâmpada que possibilita o controle via *Wi-Fi* e a Figura 4.5 ilustra o aplicativo de celular utilizado para seu manuseio.

Figura 4.4: Lâmpada Philips Hue White and Color [3]

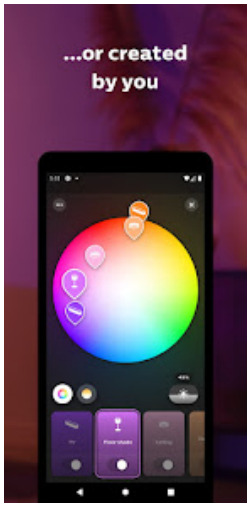


Figura 4.5: Philips Hue App [4]

4.2 Software

Nesta seção são introduzidas as ferramentas de *software* utilizadas no desenvolvimento das diferentes *Application Programming Interfaces* (APIs) implementadas pelo sistema para execução de rotinas. É importante destacar que esses programas operam em sistema embarcado e apresentam restrições de espaço e processamento.

4.2.1 Flask

Flask é um *framework* para desenvolvimento de aplicações *web* na linguagem Python notório pela pequena dimensão de seu código fonte. Diferente de outros *frameworks*, a ferramenta não possui suporte nativo para acesso de bancos de dados, autenticação de usuários ou outros tipos de tarefas de alto nível comuns em *softwares web*. Para usufruir desses serviços em um projeto *Flask*, é necessária a adição de extensões ao código principal.

Efetivamente, o *framework Flask* é uma alternativa que oferece flexibilidade ao desenvolvedor, tornando-o capaz de escolher individualmente os diferentes componentes da sua aplicação e facilmente anexá-los à estrutura já feita. Essa característica contrasta-se com outros *frameworks* de natureza monolítica, os quais optam por incorporar esses inúmeros serviços ao seu código principal [53].

Essa autonomia na escolha dos componentes integrantes da aplicação permite a manutenção do tamanho reduzido e da simplicidade do sistema, o quais configuraram fatores decisivos para a seleção do *Flask* neste trabalho.

[3] Fonte: `https://www.philips-hue.com/pt-br/p/hue-white-and-color-ambiance-pacote-de-1-e27/8718699705336`

[4] Fonte: `https://play.google.com/store/apps/details?id=com.philips.lighting.hue2`

Descrições a respeito de sua API e vários materiais de estudo podem ser encontrados no seu *site* oficial `https://flask.palletsprojects.com`.

Flask-RESTul

O Flask-RESTful é uma extensão do Flask que agrega suporte à implementação rápida e simples de APIs *RESTful* e foi utilizada neste trabalho a fim de assegurar a organização de suas interfaces. O *site* oficial `https://flask-restful.readthedocs.io` explica em detalhes como utilizá-la.

4.2.2 TinyDB

O TinyDB é um banco de dados orientado a documentos escrito em Python que apresenta a possibilidade de extensão através de *Middlewares*. Sua escolha é ideal para contextos com limitações de espaço e de natureza simples, e, por isso, foi empregado neste projeto. Em compensação, o seu uso não se encaixa em cenários exigentes em relação a performance ou cenários que dependem de recursos avançados.

Descrições a respeito de sua API e vários materiais de estudo podem ser encontrados no seu *site* oficial `https://tinydb.readthedocs.io`.

4.2.3 TensorFlow

O TensorFlow é uma biblioteca de código aberto que opera como interface para escrita de algoritmos de aprendizado de máquina [54]. Seu modelo de execução foi concebido de maneira que toda a lógica de estados e operações referentes a um algoritmo fosse representada por um grafo de fluxo de dados único. Dessa forma, no TensorFlow, cada vértice de um grafo de execução traduz-se em uma unidade de computação atômica, e cada nó, em seu respectivo valor de entrada ou de saída. [55].

A computação expressa pelo TensorFlow pode ser transcrita para uma variedade de sistemas com pouco ou nenhum retrabalho, e seu código foi idealizado para ser utilizado tanto em dispositivos móveis, como *tablets* e celulares, quanto sistemas distribuídos de larga escala com centenas de máquinas. Além disso, a biblioteca é capaz de expressar uma grande variedade de algoritmos e tem sido uma ferramenta crucial para a pesquisa e desenvolvimento de inúmeras áreas de estudo, como a visão computacional, a robótica, o processamento de linguagem natural, o reconhecimento de fala e muitas outras [54].

Neste trabalho, um dos aspectos determinantes para o uso do TensorFlow na implementação de algoritmos de aprendizado foi a existência de compilações oficiais do projeto para arquitetura ARM, a mesma arquitetura da SBC escolhida. Particularmente, foi utilizado o TensorFlow versão 2.0.0.

Descrições a respeito de sua API e vários materiais de estudo podem ser encontrados no seu *site* oficial `www.tensorflow.org`.

4.2.4 Pandas

Pandas é uma biblioteca de estrutura de dados em Python que tem por objetivo auxiliar o manuseio de diferentes conjuntos de dados e prover elementos fundamentais para a implementação de modelos estatísticos [56]. Ela é utilizada neste trabalho com intuito de facilitar o manejo dos inúmeros dados coletados e produzidos.

Descrições a respeito de sua API e vários materiais de estudo podem ser encontrados no seu *site* oficial `https://pandas.pydata.org/`.

Capítulo 5

Metodologia e Desenvolvimento

A pesquisa do Capítulo 3 foi crucial para a formação de uma perspectiva geral a respeito do cenário de automações IoT que exploram as vantagens dispostas pelo campo de aprendizado de máquina. Esse trabalho, fomentado pela ausência de projetos similares de acordo com o levantamento da Tabela 3.1, enfatiza o uso de *hardware* embarcado na execução dos algoritmos de ML e procura detalhar os respectivos obstáculos que nascem consequentes dessa decisão.

Dessa maneira, este capítulo descreve o projeto de iluminação inteligente citado na introdução do documento. Na Seção 5.1 são detalhadas as especificações de *hardware* determinadas de acordo com o escopo abrangido. Na Seção 5.2 e na Seção 5.3 são discutidos os métodos aplicados para desenvolvimento do módulo de ML do sistema. E, por fim, na Seção 5.4 são descritos os diversos componentes de *software* implementados para execução das tarefas propostas.

5.1 Protótipo de hardware

5.1.1 Definição de Escopo

O primeiro passo na estruturação do projeto foi a definição de seu escopo. Essa etapa foi fundamental para a escolha dos dispositivos utilizados e, consequentemente, dos tipos de algoritmos empregados pelo sistema. Dessa forma, enumerou-se requisitos guias para seleção dos *hardwares* a serem reunidos:

- Baixo custo; e
- Possibilidade de aprendizado embarcado.

Considerando isso, foi idealizada uma configuração simplificada, a qual compreende uma controladora, vários sensores e uma lâmpada inteligente. Fisicamente, todos os

sensores conectam-se via fios a um microcontrolador intermediador, responsável pela coleta dos múltiplos sinais. Diante disso, esse microcontrolador fornece esses dados para a controladora principal, a qual se encarrega do armazenamento e processamento dessas informações, assim como produz os resultados referentes ao modelo de ML. E, finalmente, a controladora, utilizando as predições do algoritmo de aprendizado de máquina, controla o estado da lâmpada inteligente, isto é, liga ou desliga a lâmpada.

Esse arranjo foi concebido para ser instalado em apenas um cômodo, de modo que o controle e a inspeção dos experimentos fossem facilitados. A Figura 5.1 ilustra a disposição real dos dispositivos no espaço de testes.

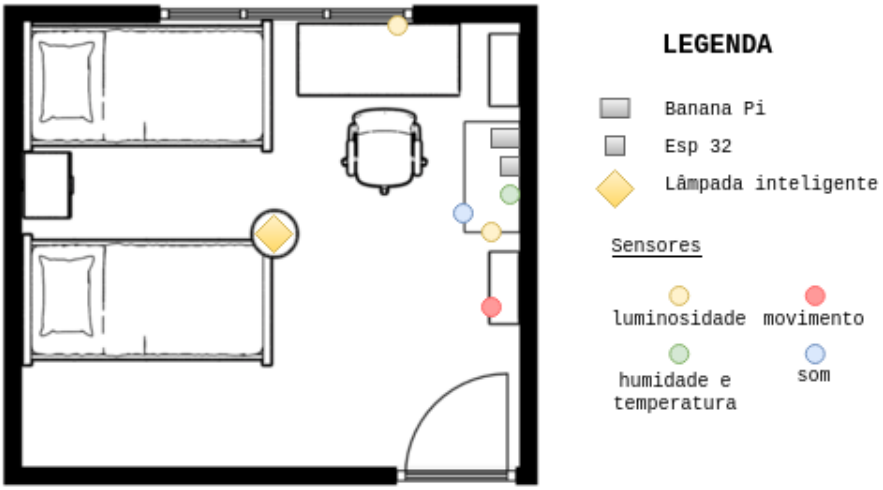


Figura 5.1: Ilustração dos componentes no espaço de teste.

5.1.2 Controladora

Os modelos de SBCs da família Raspberry Pi foram frequentemente mencionados na revisão de literatura estudada. Assim, a escolha da controladora do sistema foi embasada em opções que apresentassem recursos de processamento similares. No fim, foi verificado que linhas alternativas de SBCs possuem custo reduzido quando comparadas a família Raspberry Pi, e por essa razão foi escolhido o modelo Banana Pi BPI-M3.

Uma das dificuldades encontradas devido à escolha de uma SBC menos difundida foi a quantidade reduzida de documentos de suporte disponíveis. Nesse sentido, durante o processo de instalação de um sistema operacional na controladora, diversas imagens não funcionais foram testadas. No fim, optou-se pelo uso de sistemas operacionais do projeto *Armbian* [57], baseados em Linux e otimizados para chips do tipo ARM. Nesse período

de desenvolvimento, o modelo BPI-M3 utilizado não dispunha de suporte completo pela equipe *Armbian*, mas possuia uma lista de imagens possivelmente compatíveis listadas em `https://armbian.hosthatch.com/archive/bananapim3/archive/`. No fim, a imagem instalada foi a *Bionic Current Desktop* na sua versão 5.9.14.

A Tabela 5.1 detalha as especificações de *hardware* da placa **Banana Pi BPI-M3** empregada como controladora deste projeto.

5.1.3 Sensores

Como mencionado na Seção 5.1.1, a comunicação entre os sensores e a controladora foi intermediada por um microcontrolador, especificamente, o **ESP32 DEV-KIT V1**. Ao utilizar essa placa extra como ponte, foi possível usufruir dos benefícios da plataforma Arduino, que oferece uma gama de bibliotecas para a manipulação de componentes de monitoramento. Além disso, o microcontrolador escolhido implementa interfaces específicas não encontradas na SBC principal, como conversores de sinais analógicos para digitais (ADCs). A conexão dos diversos sensores ao microcontrolador é ilustrada pela Figura 5.2, e nas subseções seguintes os sensores selecionados para o projeto são detalhados.

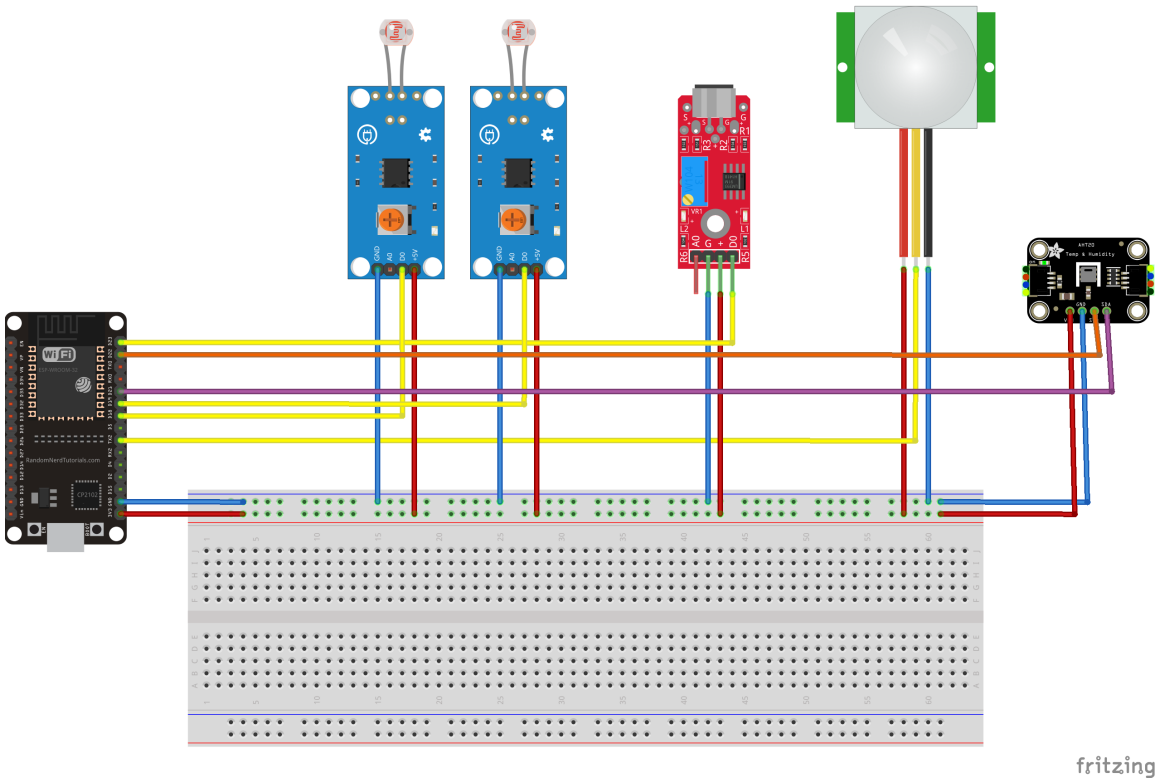


Figura 5.2: Protótipo com microcontrolador e sensores..

Tabela 5.1: Especificações da Controladora (Fonte: [58]).

	Banana Pi BPI-M3
CPU	Allwinner A83T ARM Cortex-A7 Octa-Core 1.8 GHz, 512KiB L1 cache e 1MiB L2 cache
GPU	PowerVR SGX544MP1 compatível com OpenGL ES 2.0 OpenCL 1x, $DX9_3$
Memória	2GiB LPDDR3 (compartilhada com GPU)
Disco	8GiB eMMC Flash On Board , slot Micro SD-Card e porta SATA 2.0 (interface USB para SATA)
Rede	10/100/1000 Mbit/s Ethernet (Realtek RTL8211E/D) + Wi-Fi 802.11b/g/n (AP6212) + Bluetooth BT4.0
Entrada de Vídeo	Conector de entrada CSI permite a conexão de um módulo de câmera
Saída de Vídeo	HDMI 1.4 e MIPI Display Serial Interface (DSI) para painel LCD bruto
Entrada de Áudio	Microfone na placa
Saída de Áudio	3.5mm jack e HDMI
Portas USB	USB 2.0 PORT (x2), USB OTG (x1)
Remoto	Receptor infravermelho na placa (x1)
GPIO	40 Pin Header : GPIO (x28) e Power (+5V, +3.3V and GND). Alguns dos pinos de I/O Pin podem ser usados para funções específicas, como UART, I2C, SPI ou PWM
Botões	Reset, Power e U-boot
LED	Power Status e 8P8C
Alimentação	5V/2A DC
Tamanho e Peso	92×60mm, 48g

Sensores de Luminosidade

Os módulos de detecção de luminosidade instalados nesse projeto são de uma categoria frequentemente assinalada em projetos IoT, que utilizam um comparador **LM393** e um terminal ajustável para detecção de um parâmetro qualquer. Nesse caso, a detecção de luz é implementada por meio de um resistor dependente de luz. Um circuito parecido também é verificado no módulo sensor de intensidade sonora a ser descrito posteriormente.

Desse modo, dois sensores de luminosidade foram situados no ambiente, um de frente a única janela do quarto e outro acima de uma luminária, com objetivo de supervisionar as fontes de luz do cômodo que não sejam a lâmpada inteligente.

Sensor de temperatura e umidade

Para monitoramento de informações ambientes do quarto, foi instalado um sensor de temperatura e umidade **AHT10** com precisão de ±0,3 °C e ±2 % de umidade relativa. A fundamentação para a captura desses sinais baseia-se na possibilidade da observação de preferências sazonais a respeito do conforto do usuário.

Anteriormente, foi testado um módulo analógico de sensor de temperatura **KY-013**, mas os valores lidos apresentavam erros relativamente grandes em comparação à faixa esperada. Entre as causas especuladas para o problema, foram listadas a falta de calibração do conversor ADC e a influência térmica de outros componentes posicionados próximos ao sensor. Assim, o módulo de alta precisão **AHT10** foi empregado com intuito de transpassar os obstáculos conferidos.

Sensor de movimento

Um módulo detector de movimento **MH-SR602** foi fixado próximo à porta do cômodo de testes. Esse sensor foi selecionado com a finalidade de verificar a presença de residentes no quarto. Especificamente, o módulo **MH-SR602** foi escolhido por atender os requisitos de alimentação determinados pelo microcontrolador utilizado no projeto, que opera com tensão de $3.3V$.

Esse sensor permite o ajuste do período de permanência do modo ativo mediante à variação de um valor resistivo em seu circuito, exemplificado pela Tabela 5.2. Na sua configuração final, foi aplicada uma resistência de $422k\Omega$.

Sensor sonoro

Por fim, um único sensor de intensidade sonora foi implantado no sistema, também com objetivo de auxiliar a detecção de presença de usuários no espaço. Nota-se, no entanto,

Tabela 5.2: Ajustes do módulo detector de movimento MH-SR602 (Fonte: [59]).

Resistência (Ω)	**Tempo (s)**
0 (padrão)	2,5
51k	6
91k	8
120k	10
180k	17
220k	32
270k	47,5
330k	62,5
360k	122
430k	243
510k	360
560k	480
680k	950
750k	1865
910k	2790
1M	3715

que o módulo manipula um simples microfone de eletreto sem usufruir de circuitos amplificadores auxiliares, apresentando, então, grandes limitações em relação às amplitudes sonoras verificadas.

5.1.4 Atuadores

O único atuador planejado na arquitetura descrita é a lâmpada principal do ambiente. Dessa forma, foi imperativo o desenvolvimento de um canal de comunicação atuador-controladora que possibilitasse a execução de ações modificadoras do estado da lâmpada. O uso de lâmpadas inteligentes, ou seja, de lâmpadas capazes de conectar-se a diversos dispositivos, foi a solução encontrada para essa premissa. Entretanto, ao revisar as diferentes alternativas de lâmpadas inteligentes, evidenciou-se a enorme influência de *firmwares* proprietários nesse setor de aparelhos. Esse fato, de maneira prática, indicava que a maioria das lâmpadas disponíveis inviabilizavam a comunicação direta, sem o uso de servidores proprietários terceiros, com o Banana Pi.

Dessa maneira, foi introduzida no sistema uma lâmpada programada com o *firmware* aberto Tasmota [60] que utiliza a rede *Wi-Fi* para sua comunicação. Essa especificação permitiu que a arquitetura completa operasse no âmbito da rede *Wi-Fi* local presente no espaço de testes.

5.1.5 Configuração final

Recapitulando a configuração inicialmente proposta, tem-se uma rede de sensores conectada a um dispositivos **ESP32**, que, por sua vez, envia via *Wi-Fi* os dados coletados pelos sensores para o computador **Banana Pi BPI-M3**, o qual, por fim, aproveita dessas informações para alterar aspectos da lâmpada Tasmota. A Figura 5.3 mostra uma visualização do processo explicado.

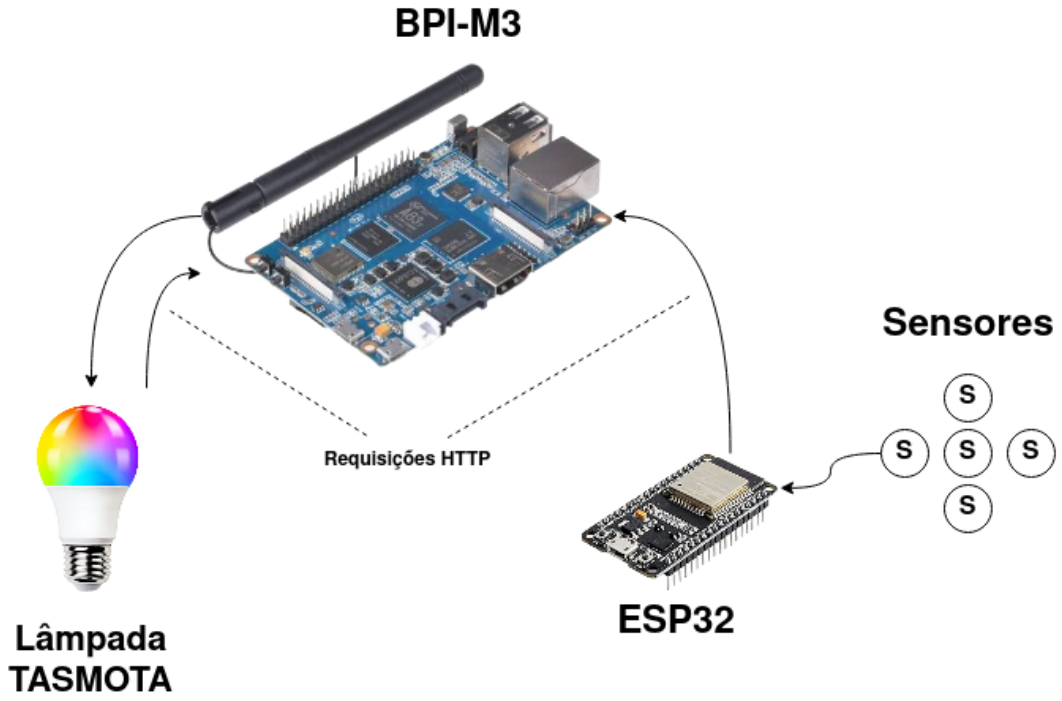


Figura 5.3: Esboço da configuração final.

5.2 Datasets

Para o desenvolvimento do modelo de ML a ser futuramente utilizado pelo sistema proposto, foi fundamental a construção de conjuntos de dados para o treinamento e a avaliação do algoritmo. Assim, a arquitetura de *hardware* descrita foi utilizada durante um intervalo de tempo de aproximadamente 8 meses para a coleta de informações de sensores

e do estado da lâmpada. Esse processo de captura foi realizado sequencialmente com uma taxa de coleta de 1 amostra a cada 10 minutos. Os arquivos das bases de dados resultantes são disponibilizados no repositório `https://github.com/Lary15/lightML-datasets`.

No repositório mencionado, verifica-se uma divisão de arquivos assinalada pelos prefixos *old* e *new*. Essa separação é consequência de um intervalo de manutenção realizado no sistema durante a fase de coleta, em que foi necessária a paralisação da captura de novos registros e a reinicialização do banco de dados interno da aplicação. Durante essa etapa de ajustes, também foi realizada uma troca de sensores sonoros na configuração de *hardware* utilizada. Assim, os registros coletados no período anterior a essa alteração, denominado período A, exibem valores do módulo detector de som característicos do uso de uma saída analógica, enquanto o período posterior á mudança, denominado período B, apresenta registros com valores de sensor sonoro típicos do uso de uma saída digital.

A Figura 5.4, ao visualizar um recorte dos dados coletados, ilustra as diferenças entre os dois períodos mencionados por meio dos valores de sensor sonoro registrados e destaca o intervalo de ajustes, marcado pela escassez de amostras. Dessa maneira, a Tabela 5.3 resume as informações presentes nos arquivos de conjuntos de dados disponibilizados de acordo com os períodos de coleta abrangidos por cada um.

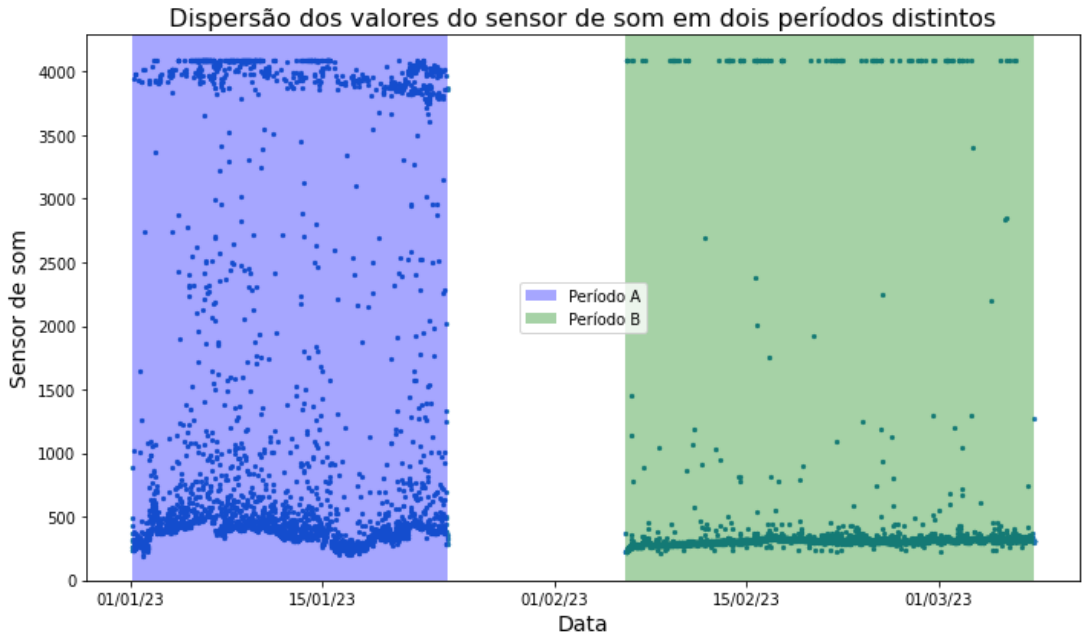


Figura 5.4: Recorte dos valores coletados pelo sensor sonoro em dois períodos distintos.

Os atributos trabalhados pelas bases de dados geradas são resumidos pela tabela Tabela 5.4. Além disso, as quantidades de amostras disponíveis em cada um dos *datasets* são expostas pela Tabela 5.5. Nesse trabalho, os conjuntos de treinamento, validação e

Tabela 5.3: Informações presentes nos conjuntos de dados por períodos de coleta.

Períodos de coleta	
A	A + B
old_train	new_train
old_test	new_test
old_val	new_val

teste foram separados de maneira aleatória e estratificada em razões de 70%, 20% e 10%, respectivamente, em relação aos dados englobados por períodos de coleta.

Tabela 5.4: Atributos das bases de dados.

Atributo	**Descrição**
pir	valor referente a sensor de movimento; 1 = movimento detectado
ldr_1	valor referente a sensor de luminosidade próximo à janela; 0 = luz detectada
ldr_2	valor referente a sensor de luminosidade próximo à abajur; 0 = luz detectada
temp	valor de temperatura ambiente em graus Celsius; varia de 0 a 40ºC
hum	valor de umidade relativa ambiente em porcentagem; varia de 0 a 100%
mic	valor referente a sensor de intensidade sonora; varia de 0 a 4095 de acordo com a resolução do módulo ADC utilizado para leitura
led_on	valor referente ao estado da lâmpada; 1 = luz acesa/ON
timestamp	estampa temporal no formato *Unix epoch*

Uma análise aprofundada da Tabela 5.5 também expõe o desbalanceamento existente entre as amostras de lâmpada acesa e os registros de lâmpada desligada. Esse cenário é condizente à rotina de iluminação residencial padrão, que, em geral, somente aciona suas luzes por volta do período noturno.

5.3 Modelo ML

Os trabalhos conferidos no capítulo de revisão de literatura demonstraram uma enorme diversidade no conjunto de algoritmos de aprendizado de máquina empregados em contextos de automação. Dessa maneira, durante o processo de decisão do método de aprendizado

Tabela 5.5: Números de amostras por classe em diferentes bases de dados.

Dataset	**Luz acesa**	**Luz apagada**	**Total**
old_train	3268	14324	17592
old_test	467	2047	2514
old_val	933	4093	4093
new_train	5256	20148	25404
new_test	751	2879	3630
new_val	1501	5757	7258

a ser utilizado pelo sistema proposto nesse trabalho, foram considerados os seguintes requisitos específicos ao escopo desse projeto:

- Processamento compatível com o *hardware* disponível;
- Possibilidade de aprendizado incremental.

A demanda em relação ao aprendizado incremental restringiu o uso de algoritmos clássicos, como o de KNN, DT e SVM, devido ao fato desses métodos não apresentarem suporte à esse tipo de aprendizado na sua conjuntura padrão. Assim, em seguida, foram analisadas soluções baseadas em redes neurais. Nesse contexto, experimentos iniciais realizados em um ambiente de teste para comparação de performances entre uma arquitetura CNN, uma arquitetura LSTM e uma arquitetura MLP favoreceram a rede MLP (Subseção 2.4.1) que exibiu valores de acurácia e precisão média ligeiramente superiores às outras alternativas. Portanto, nesse trabalho, foi empregada uma arquitetura MLP, compatível com o aprendizado incremental e o *hardware* utilizado.

As subseções seguintes detalham as etapas de pré-processamento realizadas para definição, treinamento e avaliação da rede; a composição da arquitetura utilizada; e o procedimento de retreino estabelecido nessa aplicação.

5.3.1 Pré-processamento

As etapas de pré-processamento realizadas foram:

1. Exclusão de amostras com indicações de erros de leitura, isto é, com valores de **temp**, **hum** e **mic** negativos;
2. Normalização dos atributos **temp**, **hum** e **mic** de acordo com os limites expostos na Tabela 5.4;

3. Modificação do atributo **timestamp** de *Unix epoch* para indicação da hora do dia;
4. Adição de atributo **weekday** para indicação do dia da semana;

Dessa maneira, a nova configuração de atributos estabelecida é resumida pela Tabela 5.6.

Tabela 5.6: Atributos após etapas de pré-processamento.

Atributo	Descrição
pir	valor referente a sensor de movimento; 1 = movimento detectado
ldr_1	valor referente a sensor de luminosidade próximo à janela; 0 = luz detectada
ldr_2	valor referente a sensor de luminosidade próximo à abajur; 0 = luz detectada
temp	valor normalizado de temperatura ambiente
hum	valor normalizado de umidade relativa ambiente
mic	valor normalizado referente a sensor de intensidade sonora
led_on	valor referente ao estado da lâmpada; 1 = luz acesa
timestamp	hora do dia
weekday	dia da semana

5.3.2 Arquitetura da rede

Para definição da arquitetura final da rede MLP a ser operada pela aplicação foi utilizado um algoritmo de otimização de hiperparâmetros denominado *Random Search* [61]. Esse algoritmo foi alimentado com os dados dos conjuntos de treinamento new_train e de validação new_val modificados pelas etapas de pré-processamento citadas. Em seguida, o algoritmo foi executado durante 20 tentativas com 200 épocas de treinamento cada, utilizando o erro como métrica de avaliação para determinar a quantidade de camadas e a quantidade de unidades por camadas ideais. A configuração resultante compreende:

- Camada de entrada: 8 unidades referentes aos atributos de entrada;
- Camada intermediária 1: 64 unidades densamente conectadas e ativadas pela função ReLU (Equação 2.10);
- Camada intermediária 2: 64 unidades densamente conectadas e ativadas pela função ReLU;

- Camada intermediária 3: 24 unidades densamente conectadas e ativadas pela função ReLU; e
- Camada de saída: 1 unidade referente ao rótulo **led_on** ativada pela função sigmoide (Equação 2.8).

As métricas de desempenho dessa MLP são expostas no Capítulo 6.

5.3.3 Retreino

O procedimento de retreino desse sistema é implementado por meio de novas chamadas ao método do modelo TensorFlow responsável pelo reajuste dos pesos da rede [1]. Dessa maneira, foi possível a adaptação da rede dado o surgimento de novos dados. Mais especificamente, o retreino foi realizado com o parâmetro de épocas igual a 5 a fim de evitar superajustes.

5.4 APIs

A organização das tarefas principais da controladora do sistema foi implementada por meio de diferentes subaplicações responsáveis por rotinas particulares. Essa separação de tarefas foi crucial para a atualização de partes do sistema sem a paralisação de serviços não relacionados. Dessa forma, foram desenvolvidas APIs utilizando as ferramentas Flask e TinyDB para estruturação de subsistemas simples, baseadas no padrão de projeto *Repository* [62] e que se comunicam via requisições HTTP. A seguir são listada as APIs utilizadas para execução de tarefas no Banana Pi.

5.4.1 lightML-api

API responsável pelo agrupamento e armazenamento de dados dos diversos sensores e da lâmpada Tasmota. Disponibiliza as seguintes requisições HTTP:

- GET /peripherals: listagem de amostras coletadas com valores de sensores e atuadores. Pode receber parâmetros *start* e *end* no formato *Unix epoch* para delimitação da pesquisa por datas.

Código fonte encontrado em `https://github.com/Lary15/lightML-api`.

[1] Método *fit* (Fonte: `https://www.tensorflow.org/api_docs/python/tf/keras/Model#fit`)

5.4.2 lightML-ml-api

API responsável pela execução de rotinas de retreino e predição do modelo de ML. Disponibiliza as seguintes requisições HTTP:

- POST /predict: prediz probabilidade da lâmpada estar acesa de acordo com as amostras repassadas;
- POST /retrain: aciona retreino do modelo utilizando dados repassados; e
- GET /reports: listagem de métricas de desempenho salvas por instância de reaprendizado.

Código fonte encontrado em `https://github.com/Lary15/lightML-ml-api`.

Capítulo 6

Resultados

Uma característica destacada por esse trabalho é a capacidade de aprendizado contínuo, explorada com intuito de superar métodos usuais de automação baseados em regras, e, assim, possibilitar configurações dinâmicas capazes de se moldarem a partir do cotidiano de seu(s) ocupante(s). Assim, tanto os experimentos sugeridos nesse capítulo quanto o estudo dos seus efeitos ponderam sobre a inteligência do sistema proposto, conferindo se o conhecimento adquirido pela aplicação realmente influenciou suas decisões futuras.

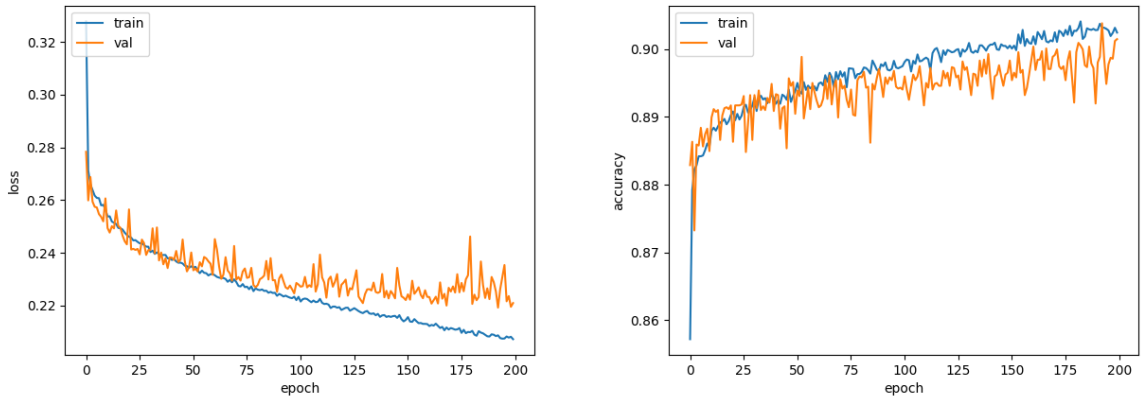
Este capítulo, então, abrange os experimentos realizados para avaliação da arquitetura descrita no capítulo anterior. Na Seção 6.1, são exibidas as métricas de desempenho observadas pelo modelo MLP utilizado na aplicação anterior a qualquer etapa de retreino. Na Seção 6.2, são detalhados os cenários de testes idealizados para avaliação da capacidade de reaprendizado do sistema. Na Seção 6.3, são expostas métricas de desempenhos coletadas durante as etapas de reaprendizado. E, por fim, na Seção 6.4 são discutidos os resultados obtidos.

6.1 Métricas precedentes aos experimentos

Para emprego do algoritmo no sistema de automação, a rede MLP descrita no capítulo anterior foi treinada com auxílio do conjunto de treinamento new_train e do conjunto de validação new_val, especificados na Seção 5.5. Ao validar o algoritmo com o conjunto de testes new_test, obteve-se 90% de acurácia, 76% de precisão, 78% de *recall* e 77% de F1-score. Esses resultados, conjuntamente à matriz de confusão exposta pela Tabela 6.1, demonstram a dificuldade do modelo em predizer corretamente a classe referente à lâmpada acesa, uma conjuntura possivelmente gerada pelo desbalanceamento de classes presente nos conjuntos de dados utilizados. Além disso, as curvas de acurácia e de perda do modelo são visualizadas pelas Figuras 6.1b e 6.1a, respectivamente.

Tabela 6.1: Matriz de confusão.

		Rótulos reais	
		Luz acesa	Luz apagada
Rótulos preditos	Luz acesa	2692	187
	Luz apagada	167	584



(a) Perda (b) Acurácia

Figura 6.1: Curvas de acurácia e perda da rede MLP utilizada pelo sistema

6.2 Experimentos

Para avaliação da habilidade de adapatação do sistema, idealizou-se um cenário em que foi inserida uma diferença comportamental em um dos sensores da arquitetura. Dessa maneira, é possível conferir as implicações dessa mudança nas predições do sistema e verificar se, dado o retreino, a aplicação é capaz de acomodá-las.

Na prática, essa alteração foi implementada em relação à leitura dos valores do sensor de intensidade sonora. Assim, durante o período de testes, a descrição do atributo do sensor de som foi atualizada para a soma de ultrapassagens de um limiar sonoro, determinando um constraste em relação ao significado do mesmo atributo no período de coleta de dados especificado na Seção 5.5. Nota-se que a alteração foi feita respeitando-se os limites já estabelecidos para esse atributo.

Dessa maneira, definiu-se duas etapas experimentais caracterizadas pela sua cadência de retreino. A primeira etapa foi programada com retreinos semanais, enquanto a segunda, com retreinos diários. Ressalta-se que a semana de testes 1 refere-se a um estágio anterior a primeira rotina de reaprendizado, e, dessa forma, é performada utilizando a configuração da MLP originária do treinamento inicial. A Tabela 6.2 resume os cenários definidos.

Tabela 6.2: Resumo das etapas experimentais e seus períodos equivalentes.

Semana de teste	**Período equivalente**
Treinamento semanal	
Semana 1	10/05/23 até 17/05/23
Semana 2	17/05/23 até 24/05/23
Semana 3	24/05/23 até 31/05/23
Semana 4	31/05/23 até 07/06/23
Semana 5	07/06/23 até 14/06/23
Treinamento diário	
Semana 6	14/06/23 até 21/06/23
Semana 7	21/06/23 até 28/06/23

6.2.1 Experimento com treinamento semanal

Com objetivo de ampliar a quantidade de exemplos disponíveis para os novos aprendizados do modelo ML, nesse experimento e no seguinte, a antiga taxa de captura de 1 amostra a cada 10 minutos foi substituída por 1 amostra a cada 3 minutos. Além disso, eventos de predição do estado da lâmpada foram programados em intervalos de 30 minutos.

Primeira semana

A primeira semana do experimento retrata o quão bem o sistema, ainda sem retreino, performou diante da mudança implementada no sensor. A Figura 6.2 permite a visualização da performance da aplicação nesse período. Nesse gráfico, a curva amarela explicita o estado da lâmpada em tempo real, e a curva azul, a probabilidade da lâmpada estar acesa de acordo com as predições do algoritmo de aprendizado.

De forma concreta, as predições produzidas influenciam diretamente o estado da lâmpada, já que valores iguais ou acima de 0,5 são interpretados como sinal da necessidade da luz ligada e ativam-na. Por isso, esse limiar foi destacada na imagem pela linha horizontal vermelha tracejada. Similarmente, linhas verticais pontilhadas foram usadas para delimitar a transição do período noturno, determinada pelo horário de 18h.

Neste trabalho, a detecção de erros de predição foi realizada analisando-se o estado da iluminação imediatamente após a execução de uma ação pelo sistema. Assim, entende-se que, caso a luz tome estado oposto ao predito nos 5 minutos seguintes à ação de controle automatizado, determinou-se um erro de predição, pois a lâmpada foi rapidamente

alterada pelo residente do cômodo. Esses erros também são assinalados na Figura 6.2, ilustrados pelos pontos circulares vermelhos.

No fim, essa semana demonstrou um desempenho razoável, seguindo um perfil de iluminação padrão apesar da alteração no atributo do sensor sonoro. Os dias 6 e 7, no entanto, exibiram padrões menos coerentes.

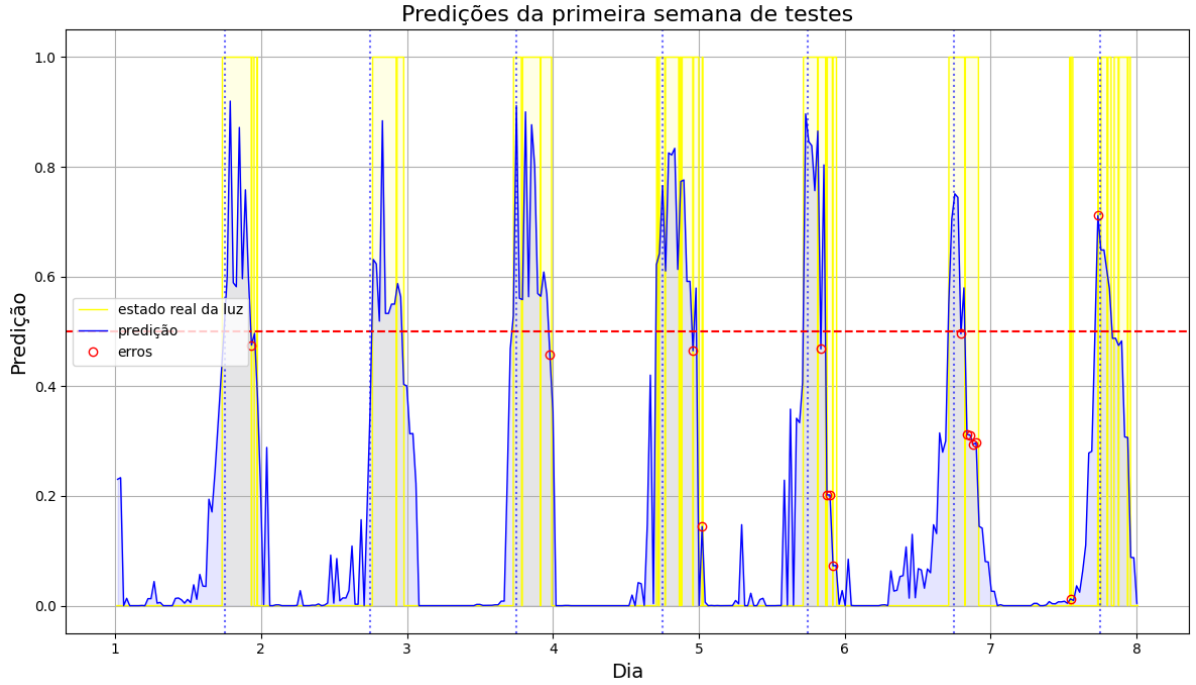


Figura 6.2: Visualização das predições durante a primeira semana de testes comparadas ao estado real da lâmpada no mesmo período.

Segunda semana

A segunda semana apresenta os resultados do sistema após seu primeiro retreino. Para visualizar os efeitos dessa atividade, a Figura 6.3 compara a performance do sistema com o estado simultâneo da iluminação no cômodo, enquanto a Figura 6.4 contrasta esse resultado aos dados coletados na semana anterior, consequentemente utilizados no retreino citado.

Nesse fase é possível verificar, principalmente ao observar a Figura 6.4, que o cotidiano das semanas 1 e 2 foi similar e que o sistema aparentou se adaptar bem a esse rotina, considerando que os valores de predição relativos à luz acesa incrementaram. É ressaltado que o número menor de erros destacados na Figura 6.3 não é integralmente fiel a realidade, devido a limitações da técnica de detecção de erros mencionada. Efetivamente, se um usuário não corrigir o sistema dentro do intervalo definido, não é possível a verificação

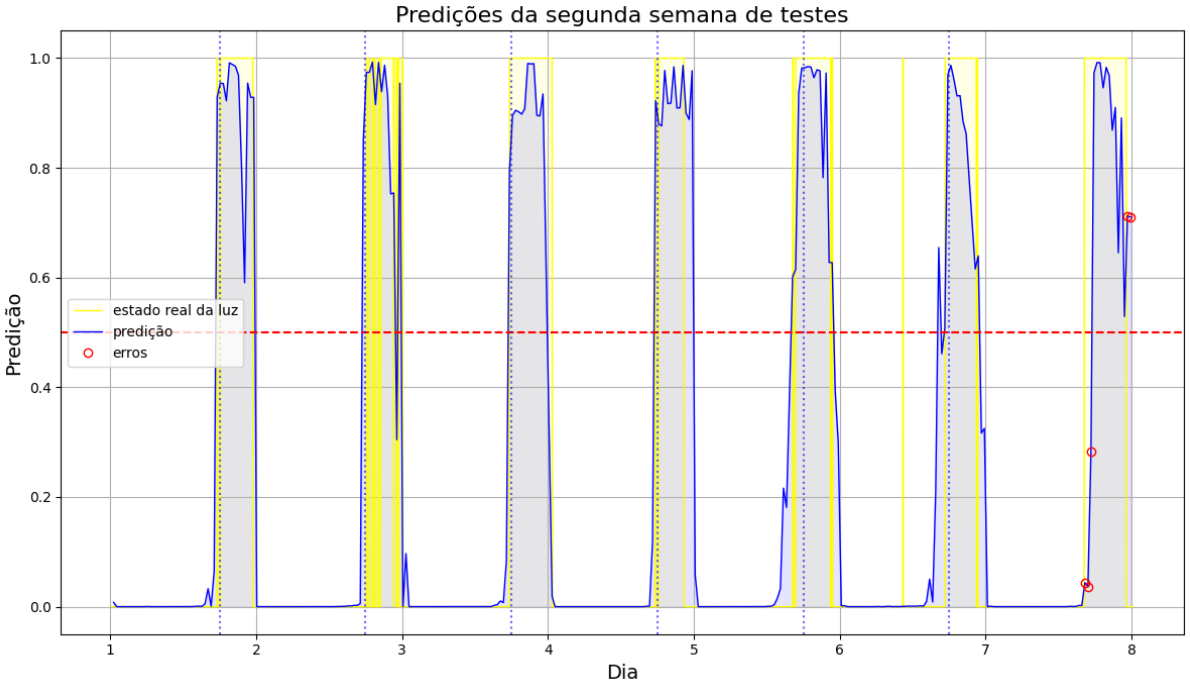


Figura 6.3: Visualização das predições durante a segunda semana de testes comparadas ao estado real da lâmpada no mesmo período.

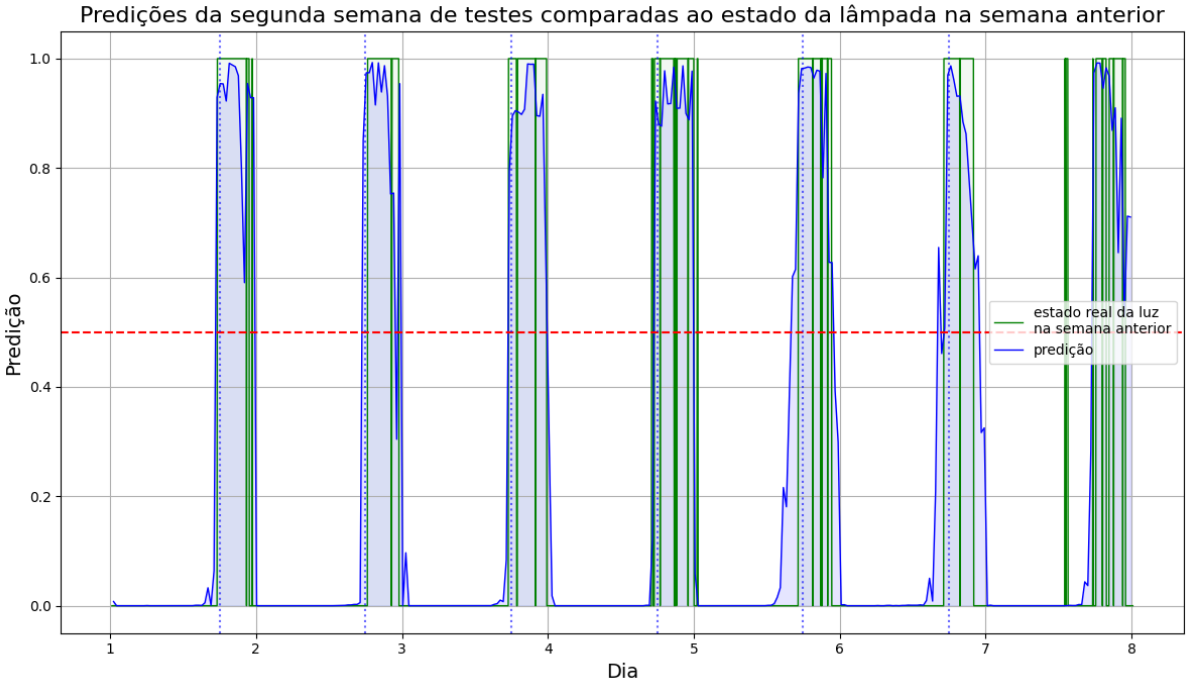


Figura 6.4: Visualização das predições durante a segunda semana de testes comparadas ao estado real da lâmpada na semana anterior.

automática de erros. O dia 2, apresentado no gráfico da Figura 6.3, ilustra essas dificuldades, ao demonstrar uma grande variedade no estado da lâmpada pós-predição mas não sinalizar nenhum erro.

Terceira semana

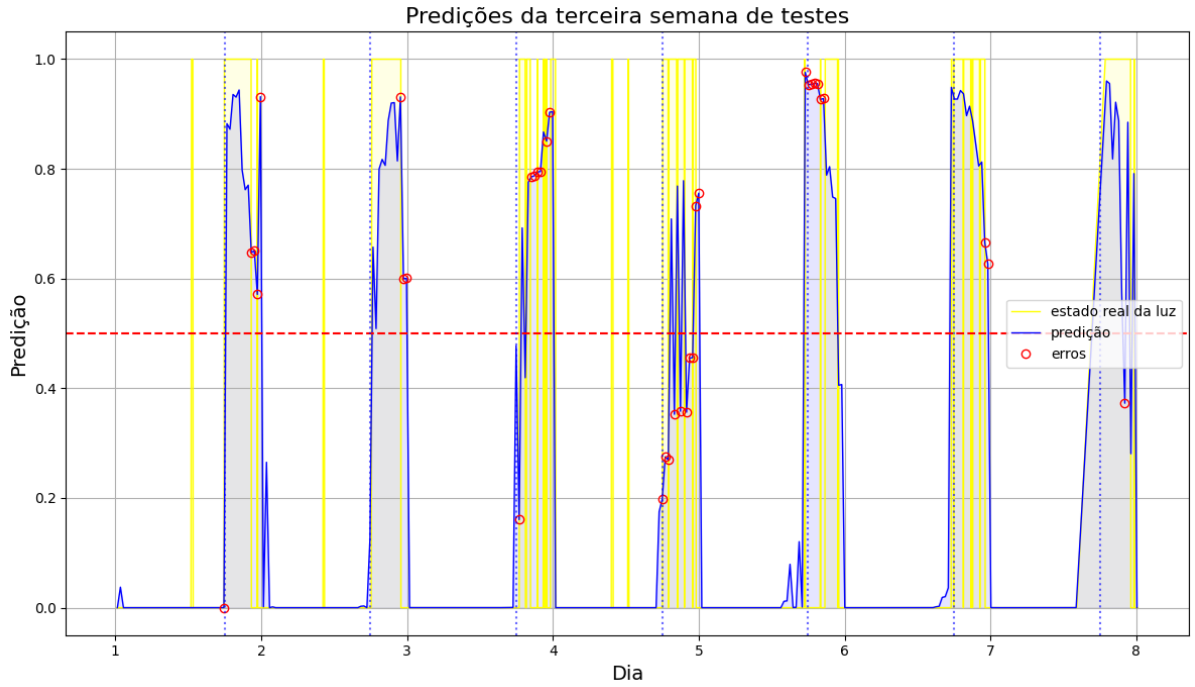


Figura 6.5: Visualização das predições durante a terceira semana de testes comparadas ao estado real da lâmpada no mesmo período.

A performance da terceira semana após a segunda etapa de retreino é visualizada pela Figura 6.5. Nessa imagem, é possível assinalar algumas novidades na rotina, nos dias 1, 2 e 4, por exemplo, a luz também foi acesa fora do período noturno.

Nesse mesmo gráfico, é observada uma queda considerável a respeito dos valores preditos para a luz acesa, e, enquanto esse declínio no dia 2 pode ser justificado pelos desligamentos da semana anterior, ilustrados na Figura 6.6, o mesmo não é verificado no dia 4, em que os valores de predição decaem consideravelmente apesar dos dados da semana anterior não refletirem um padrão de desligamento.

Um possível agente responsável por essas inconsistências é a perda de dados pelo sistema, normalmente provocada pela indisponibilidade da rede *Wi-Fi* no ambiente de teste. Nesse cenário, a comunicação entre diversos componentes do projeto e o registro de novas amostras são impossibilitados. As Figuras 6.9a, 6.9b, 6.9c e 6.9d demonstram a quantidade dessas ocorrências de perda por hora do dia. E os erros de coleta durante

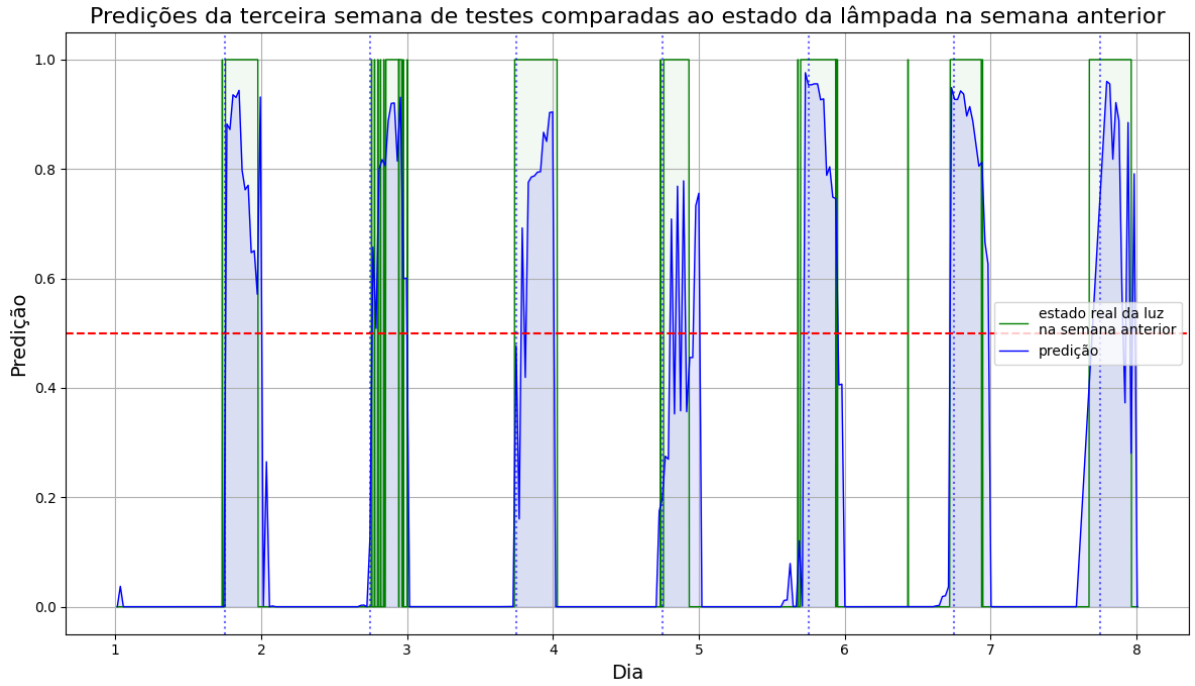


Figura 6.6: Visualização das predições durante a terceira semana de testes comparadas ao estado real da lâmpada na semana anterior.

a segunda semana de experimentos (Figura 6.9b), semana que alimentou o retreino dessa etapa, revelam-se mais frequentes quando comparados, por exemplo, à primeira semana (Figura 6.9a).

Quarta semana

A performance da quarta semana é expressa pela Figura 6.7. Nesse gráfico, verifica-se a continuidade do declínio dos valores de probabilidades calculados. Esse contexto é exemplificado, principalmente, pelos três primeiros dias da semana. Curiosamente, esse fenômeno não necessariamente instituiu um aumento na quantia de erros de predição. Isso se deu pelo fato dos valores produzidos, ainda que mais baixos, refletirem bem a rotina apresentada quando arredondados de acordo com o limiar estabelecido. Nota-se, por meio da Figura 6.9c, que a semana fornecedora dos dados consumidos por essa etapa também apresentou um número de erros de registros elevado quando comparado aos demais períodos.

Além disso, os esforços de adaptação do sistema para inclusão do novo hábito introduzidos na semana anterior são visualizados na Figura 6.8, ilustrados pelos picos de valores de predição fora do período noturno nos dias 2, 3 e 4.

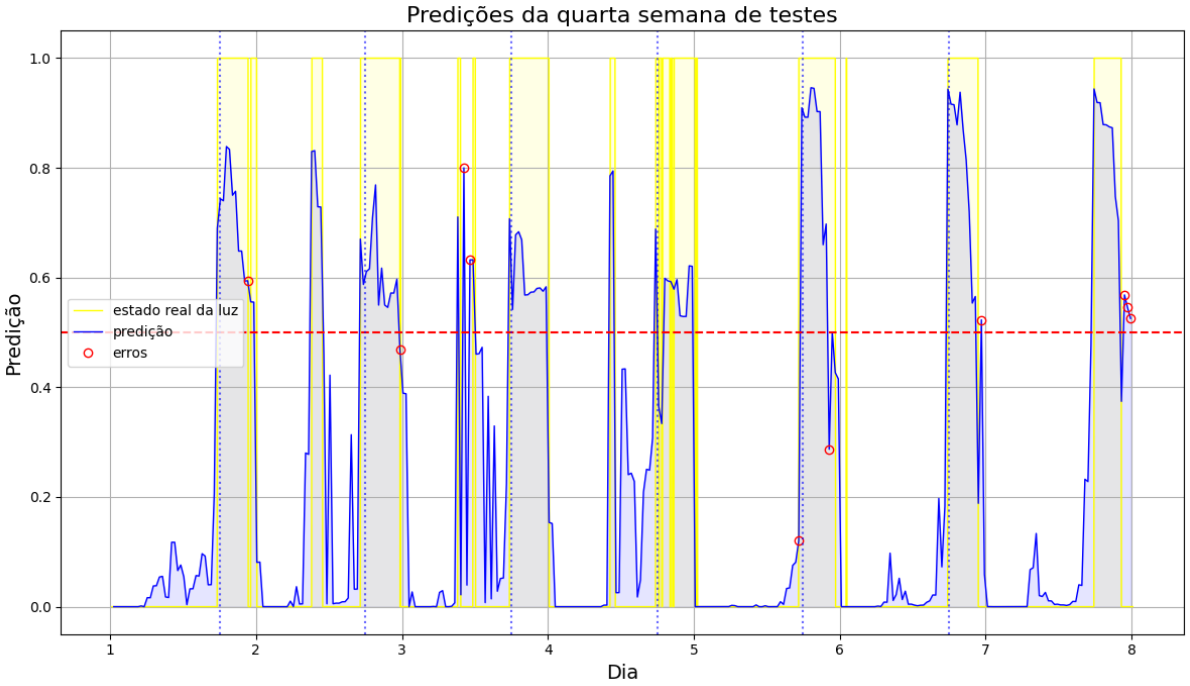


Figura 6.7: Visualização das predições durante a quarta semana de testes comparadas ao estado real da lâmpada no mesmo período.

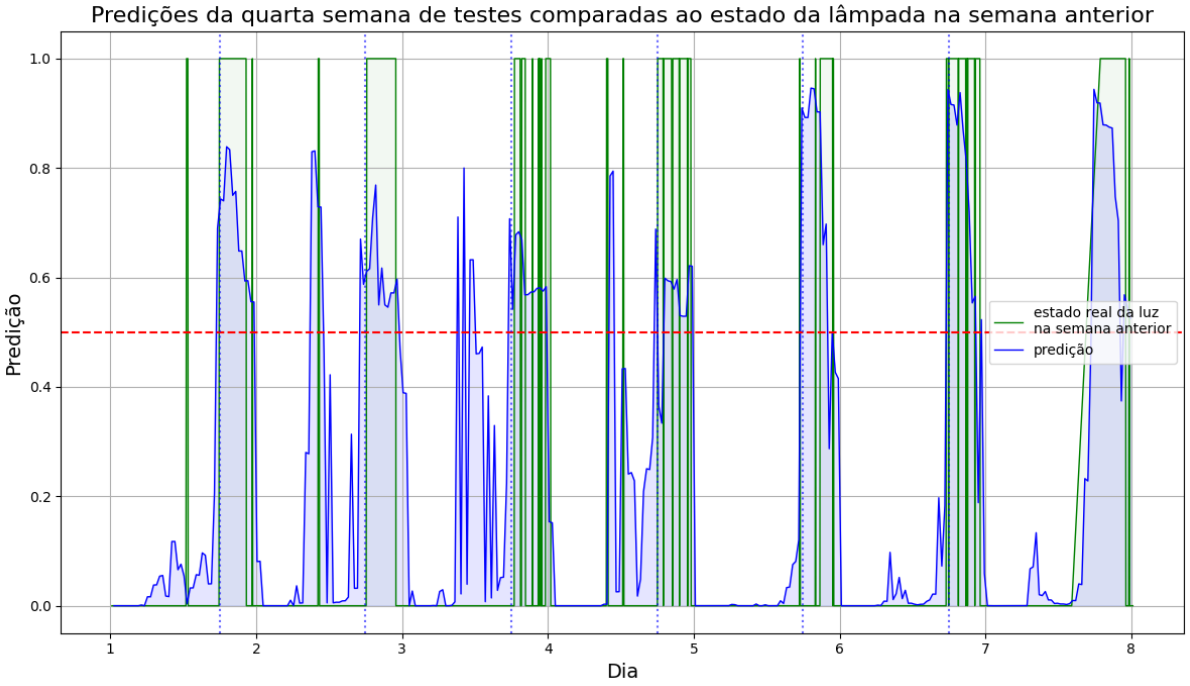
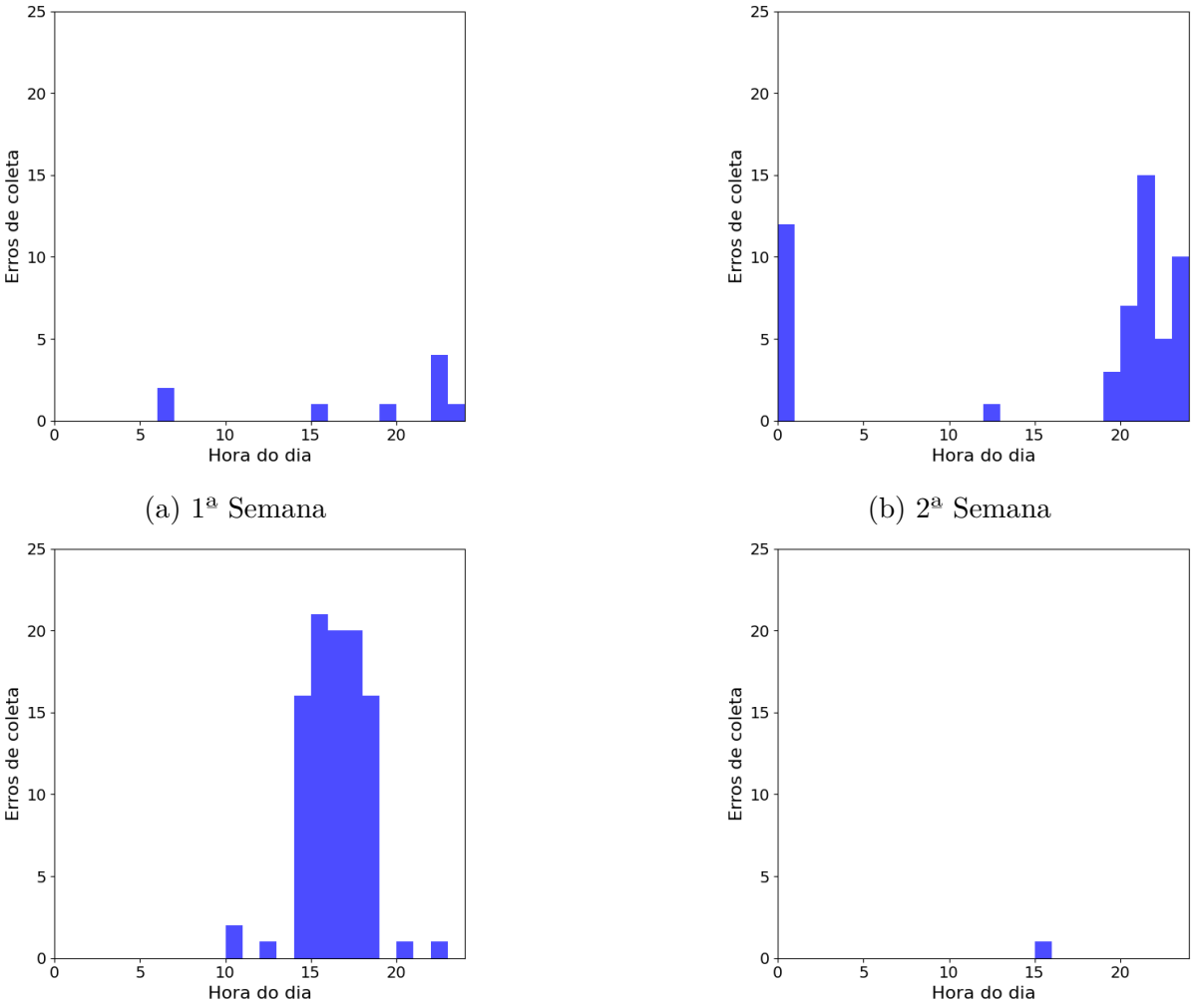


Figura 6.8: Visualização das predições durante a quarta semana de testes comparadas ao estado real da lâmpada na semana anterior.



(a) 1ª Semana

(b) 2ª Semana

(c) 3ª Semana

(d) 4ª Semana

Figura 6.9: Erros observados na coleta de dados por horas do dia em cada semana.

Quinta semana

A semana final dos experimentos de retreino semanal é visualizada pelas Figuras 6.10 e 6.11. Nesse período, verifica-se o incremento dos valores de probabilidade associados à luz acesa. Essa etapa é marcada pela extrapolação dos padrões reconhecidos anteriormente, como é visto, na Figura 6.10, pelos picos de acendimento da lâmpada. Essa generalização diária gerou alguns erros de predição.

6.2.2 Experimento com treinamento diário

Com objetivo de, possivelmente, observar o aumento do nível de responsividade do sistema, os experimentos da semana 6 (Figura 6.12) e 7 (Figura 6.13) foram remodelados

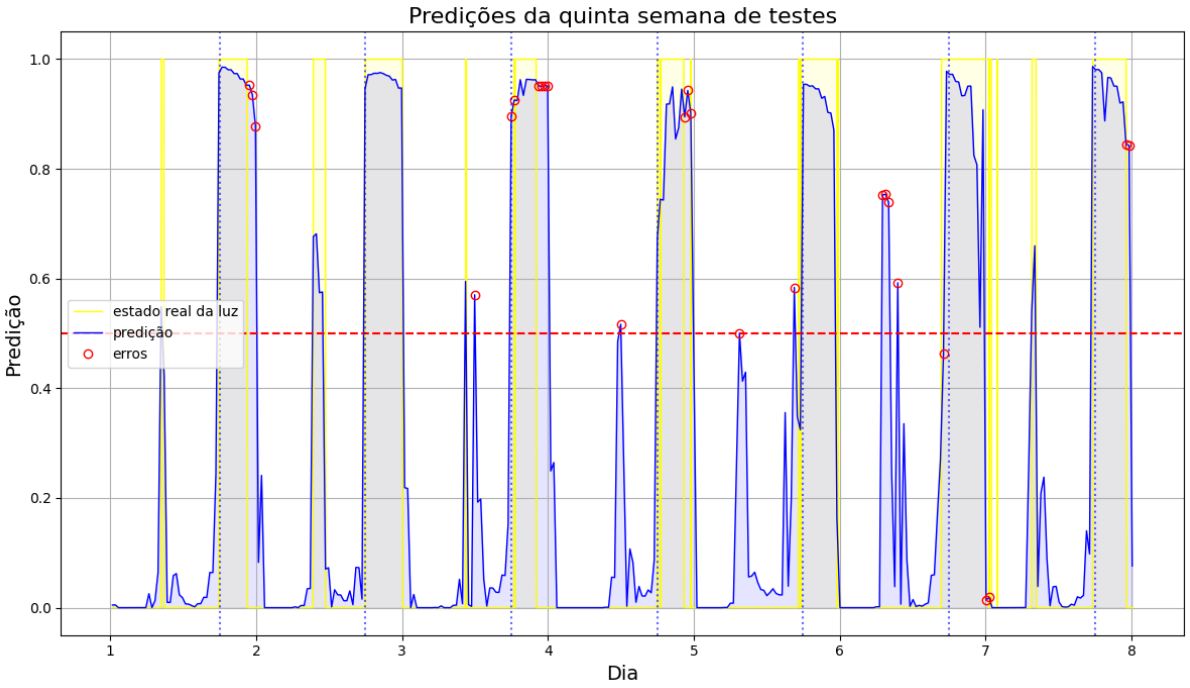


Figura 6.10: Visualização das predições durante a quinta semana de testes comparadas ao estado real da lâmpada no mesmo período.

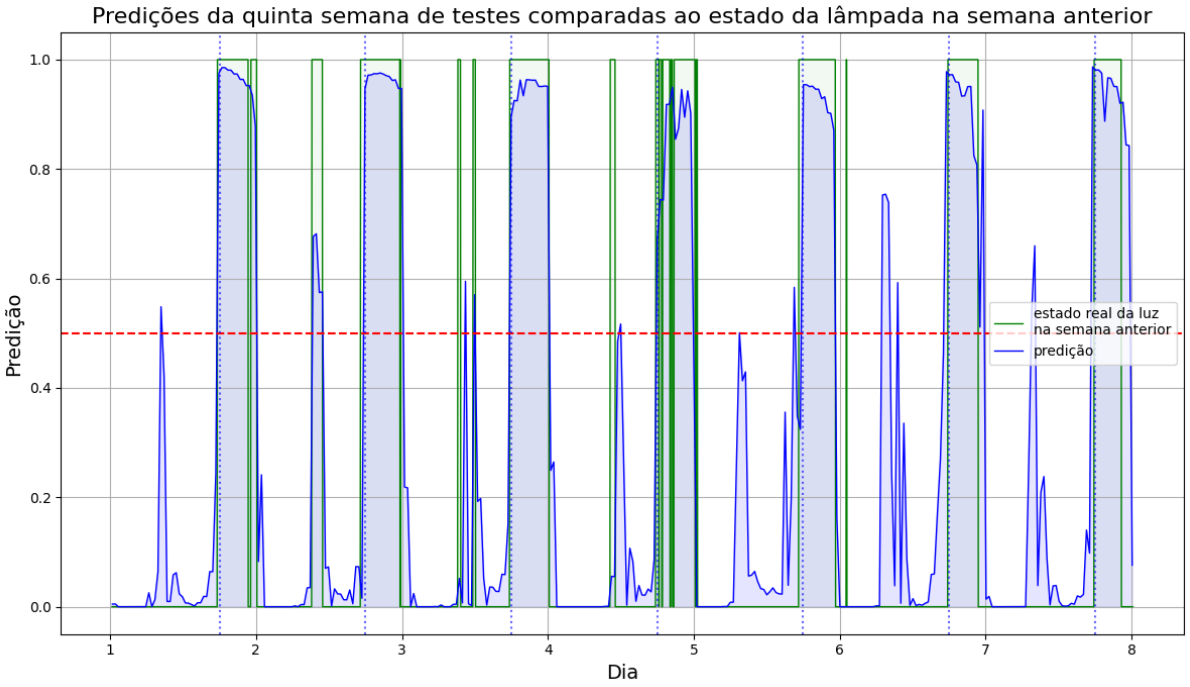


Figura 6.11: Visualização das predições durante a quinta semana de testes comparadas ao estado real da lâmpada na semana anterior.

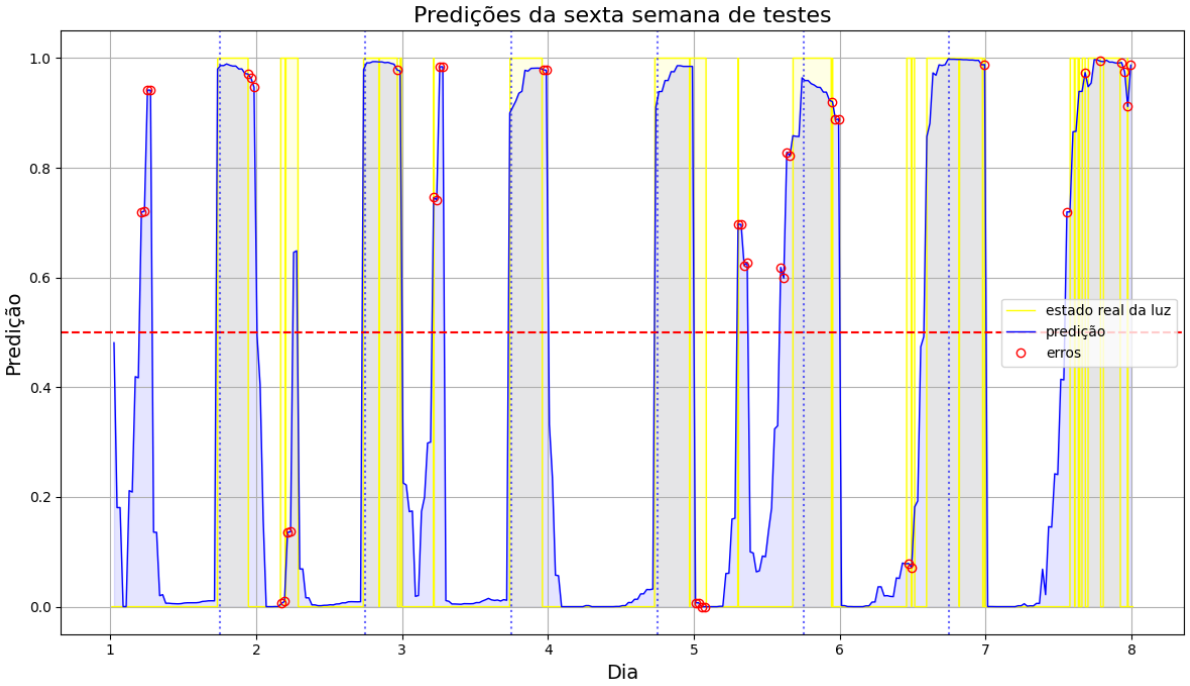


Figura 6.12: Resultados dos experimentos com retreino diário durante a sexta semana de testes.

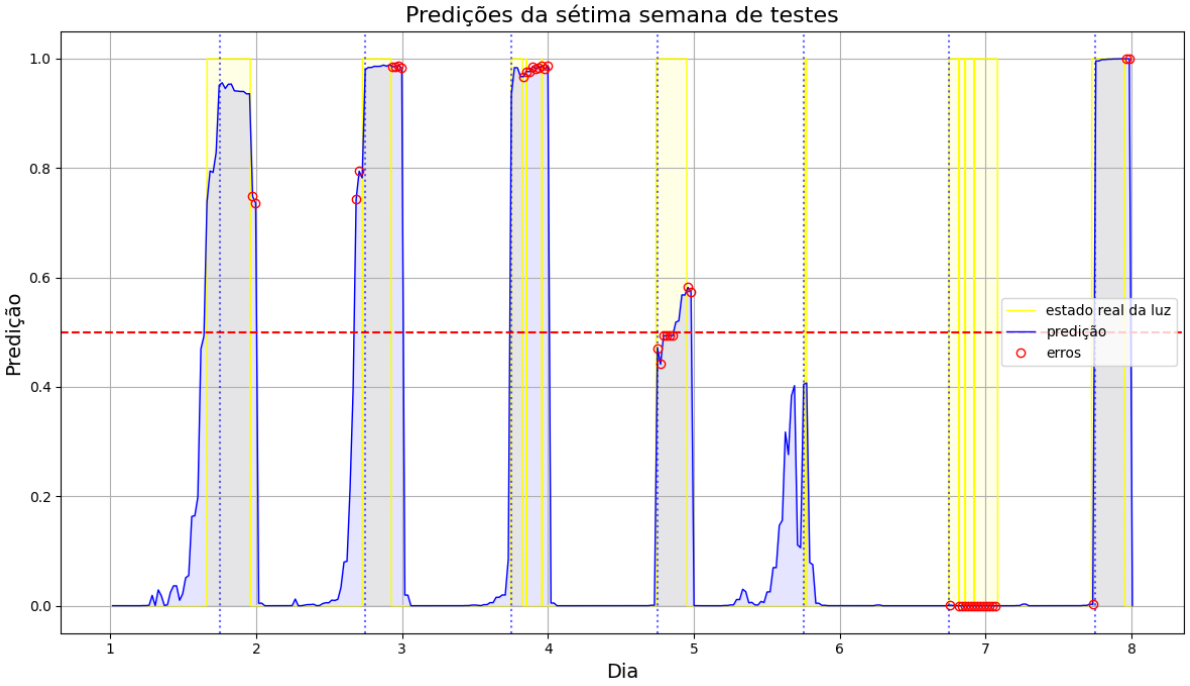


Figura 6.13: Resultados dos experimentos com retreino diário durante a sétima semana de testes.

para execução de retreinos diários, utilizando apenas os dados coletados no dia antecedente.

Nesse contexto, a performance da sexta semana, ilustrada pela Figura 6.12, demonstra o processo gradual de adaptação do algoritmo de aprendizado em relação à rotina apresentada, que já não exibe mais acionamentos da lâmpada fora do período noturno. Na sétima semana de experimentos, no entato, é verificado um comportamento inédito: no dia 6 não é feita nenhuma predição determinando a ativação da luz. Entende-se que os padrões de desligamento do dia 3 e 5 possam ter reforçado o perfil que, eventualmente, resultou no cenário do dia 6. Em contraponto, no dia 7, o sistema exibe um comportamento oposto ao do dia anterior, possivelmente fruto do reaprendizado com novas amostras favoráveis ao acionamento da luz.

6.3 Métricas por retreino

Além das análises qualitativas apresentadas, foram coletadas diversas métricas quantitativas durante o retreino do sistema. As Tabelas 6.3, 6.4 e 6.5 resumem as métricas de desempenho produzidas por cada procedimento de reaprendizado realizado no período experimental mencionado. Os valores expostos foram calculados realizando-se a média dos valores exibidos em cada uma das épocas de retreino. Nesse projeto, 10% do total de amostras recebidas para o início do reaprendizado eram separadas para validação.

Tabela 6.3: Métricas da Semana 6.

	acurácia		**precisão**		***recall***		**erros**		**acertos**	
Dia	trein.	val.	trein.	val.	trein.	val.	trein.	val.	trein.	val.
1	0,93	0,74	0,93	0,72	0,78	0,95	30	12	385	34
2	0,99	0,40	0,97	0,33	1,0	1,0	3	28	416	19
3	0,94	0,79	0,90	0,76	0,91	1,0	26	10	390	37
4	0,97	0,57	0,90	0,52	0,99	1,0	11	20	408	27
5	1,0	0,87	1,0	0,98	0,99	0,89	1	6	412	40
6	0,88	0,47	0,96	0,35	0,70	1,0	47	28	375	20
7	0,91	0,80	0,92	0,78	0,85	1,0	38	9	376	37

Os dados das Tabelas 6.3 e 6.4 expõem a dificuldade em generalização por parte do modelo quando destacada a discrepância entre suas métricas de treinamento e validação. Isso faz sentido quando considerado as grandes alterações comportamentais verificadas de

Tabela 6.4: Métricas da Semana 7.

	acurácia		**precisão**		***recall***		**erros**		**acertos**	
Dia	trein.	val.	trein.	val.	trein.	val.	trein.	val.	trein.	val.
1	0,91	0,51	0,81	0,13	0,92	0,80	36	23	380	24
2	0,98	0,92	0,98	0,90	0,96	0,95	7	4	407	42
3	0,98	0,21	0,97	0,10	0,94	1,0	7	37	407	10
4	0,92	0,98	0,75	0,0	0,54	0,0	31	1	383	45
5	0,98	0,49	0,99	0,48	0,92	1,0	6,8	24	413	23
6	0,97	1,0	0,01	0,0	0,07	0,0	12	0	399	46
7	0,98	0,85	0,94	0,98	0,93	0,87	9	7	400	39

Tabela 6.5: Métricas referentes às semanas do experimento de retreino semanal.

	acurácia		**precisão**		***recall***		**erros**		**acertos**	
Semana	trein.	val.	trein.	val.	trein.	val.	trein.	val.	trein.	val.
2	0,97	0,83	0,93	0,70	0,95	0,68	71	55	2816	265
3	0,96	0,85	0,87	1,0	0,91	0,64	124	47	2730	270
4	0,93	0,98	0,81	1,0	0,84	0,94	174	5	2603	304
5	0,95	0,94	0,92	0,80	0,90	1,0	137	21	2770	303

um dia ao outro. Nesse sentido, as métricas de validação respectivas ao retreino semanal verificadas na Tabela 6.5 demonstram um contexto menos propenso a superajustes quando comparadas ao retreino diário.

6.4 Análises dos resultados

Recapitulando o cenário experimental estabelecido inicialmente, alterou-se o significado do atributo do sensor sonoro para aferição da capacidade de aprendizado do sistema. Em relação a isso, os resultados da Semana 2, em que se realizou o primero retreino, demonstram um reajuste efetivo por parte da rede. Além disso, é possível concluir considerações positivas a respeito da habilidade de adaptação da arquitetura quando examinadas as pequenas variações na rotina aprendidas a partir da terceira semana de testes.

Em resumo, o sistema pareceu apresentar um comportamento ativo na delimitação de intervalos de iluminação de acordo com os perfis de luminosidade experenciados, porém,

não demonstrou expor a propriedade de construção de conexões mais inteligentes com os dados recebidos. Os resultados da Semana 7, na qual um dos dias não teve a lâmpada ligada em nenhum momento, demonstram isso quando considerado o fato de que não havia usuários presentes no cômodo de experimentação à noite nos dias antecedentes a esse fenômeno. Dessa forma, enquanto o sistema, na conjuntura atual, se empenharia para replicar esse padrão, um sistema mais inteligente se beneficiaria de outros dados, como a informação de presença, antes de analisar outros tipos de atributos.

Capítulo 7

Conclusão e Trabalhos Futuros

O paradigma *TinyML* agrupa fundamentos da pesquisa de IoT, de *Machine Learning* e de *Edge Computing* para composição de arquiteturas embarcadas dinâmicas, sustentadas pelos pilares de *hardware*, *software* e algoritmos de aprendizado. As aplicações do *TinyML* são variadas, miniaturizando problemas clássicos de aprendizado de máquina como a detecção facial ou o reconhecimento de fala. Projetos de reconhecimento de anomalias e detecção de invasões a partir de imagens capturadas por câmeras de monitoramento domiciliar são exemplos do nível de automação alcançado pela integração dessas diferentes áreas de estudo. No entanto, os limites dos *hardwares* embarcados atualmente disponíveis compreendem fatores bastante restritivos nesse campo de estudo em desenvolvimento.

Nesta monografia, foi apresentado uma proposta de projeto *TinyML* embasada no controle inteligente de iluminação de um cômodo e na prototipação da aplicação concebida. A configuração final do projeto foi capaz de aprender incrementalmente com novos dados alimentados ao sistema, porém os resultados expostos no capítulo anterior exibem um estágio de inteligência por parte da aplicação ainda prematura. De modo geral, as restrições de instalação de sensores devido a sua disposição física conectada por fios e a ambiguidade de algumas informações capturadas, causadas, por exemplo, pela presença de animais no espaço experimental, demonstraram-se aspectos taxativos à performance do algoritmo ML implementado.

Para trabalhos futuros, propõe-se a expansão do número e da categoria dos sensores disponibilizados para o sistema, cenário que pode ser concretizado pelo uso de módulos de sensores sem fio. Além disso, promove-se a introdução de outros tipos de dispositivos, como os sensores PET, específicos para a detecção de animais, ao conjunto de dados coletado. Essas melhorias constituiriam alicerces fundamentais para futuras evoluções relacionadas ao modelo de aprendizado de máquina utilizada pela aplicação. Ademais, nesse paradigma, a exploração de métodos de aprendizado de HAR também é assinalada como possibilidade de investimento futuro.

Referências

[1] Salih, Kazhan Othman Mohammed, Tarik A. Rashid, Dalibor Radovanovic e Nebojsa Bacanin: *A comprehensive survey on the internet of things with the industrial marketplace*. Sensors, 22(3), 2022, ISSN 1424-8220. `https://www.mdpi.com/1424-8220/22/3/730`. 1

[2] Ray, Partha Pratim: *A review on TinyML: State-of-the-art and prospects*. Journal of King Saud University - Computer and Information Sciences, 34(4):1595–1623, 2022, ISSN 1319-1578. `https://www.sciencedirect.com/science/article/pii/S1319157821003335`. 1

[3] Bierzynski, Kay, Florian Kalleder, Pavel Lutskov, Frank Rohde, David Morales Rodríguez, Juan Mena-Carrillo, René Schöne e Uwe Aßmann: *Openlicht - a self-learning lighting system based on openhab*. 2018. 1, 14, 20

[4] McCarthy, John: *What is artificial intelligence*. `http://www-formal.stanford.edu/jmc/whatisai.html`, 2007. 3

[5] Russell, Stuart e Peter Norvig: *Artificial intelligence*. Pearson, Upper Saddle River, NJ, 4ª edição, novembro 2020. 3

[6] Géron, Aurélien: *Hands-on machine learning with scikit-learn and tensorflow: Concepts*. O'Reilly Media, 2017. 3, 8, 10, 12

[7] Faceli, Katti, Ana Carolina Lorena, João Gama e André Carlos Ponce de Leon Ferreira de Carvalho: *Inteligência artificial: uma abordagem de aprendizado de máquina*. 2011. 3, 6, 9

[8] Burkov, Andriy: *The hundred-page machine learning book*, volume 1. Andriy Burkov Quebec City, QC, Canada, 2019. 4, 5, 6, 12

[9] Fix, Evelyn e JL Hodges: *Discriminatory analysis: nonparametric discrimination: consistency properties. report. 4*. T. USAF School of Aviation Medicine, 1951. 5

[10] Cover, T. e P. Hart: *Nearest neighbor pattern classification*. IEEE Transactions on Information Theory, 13(1):21–27, janeiro 1967. `https://doi.org/10.1109/tit.1967.1053964`. 5

[11] Winterfeldt, Detlof von e Ward Edwards: *Decision analysis and behavioral research*. 1986. `https://api.semanticscholar.org/CorpusID:141605781`. 6

[12] Quinlan, J Ross: *Discovering rules by induction from large collections of examples.* Expert systems in the micro electronics age, 1979. 6

[13] Breiman, Leo, Jerome Friedman, Richard Olshen e Charles Stone: *Classification and regression trees. wadsworth int.* Group, 37(15):237–251, 1984. 6

[14] Salzberg, Steven L: *C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993.* Machine Learning, 16(3):235–240, setembro 1994. 6

[15] Cortes, Corinna e Vladimir Vapnik: *Support-vector networks.* Machine Learning, 20(3):273–297, Sep 1995, ISSN 1573-0565. `https://doi.org/10.1007/BF00994018`. 6

[16] Lorena, Ana Carolina e André C P L F De Carvalho: *Uma introdução às support vector machines.* Revista De Informática Teórica E Aplicada, 14(2):43–67, dezembro 2007. 8

[17] McCulloch, Warren S e Walter Pitts: *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 5(4):115–133, dezembro 1943. 8

[18] Gepperth, Alexander e Barbara Hammer: *Incremental learning algorithms and applications.* Em *European symposium on artificial neural networks (ESANN)*, 2016. 10

[19] Schlimmer, Jeffrey e Doug Fisher: *A case study of incremental concept induction.* páginas 496–501, janeiro 1986. 10

[20] Diehl, Chris e Gert Cauwenberghs: *SVM incremental learning, adaptation and optimization.* Volume 4, páginas 2685 – 2690 vol.4, agosto 2003, ISBN 0-7803-7898-9. 10

[21] Gupta, Neha, Suneet K. Gupta, Rajesh K. Pathak, Vanita Jain, Parisa Rashidi e Jasjit S. Suri: *Human activity recognition in artificial intelligence framework: a narrative review.* Artificial Intelligence Review, 55(6):4755–4808, janeiro 2022. `https://doi.org/10.1007/s10462-021-10116-x`. 12, 13

[22] Dang, L. Minh, Kyungbok Min, Hanxiang Wang, Md. Jalil Piran, Cheol Hee Lee e Hyeonjoon Moon: *Sensor-based and vision-based human activity recognition: A comprehensive survey.* Pattern Recognition, 108:107561, dezembro 2020. `https://doi.org/10.1016/j.patcog.2020.107561`. 12, 13

[23] Bakar, U. A. B. U. A., Hemant Ghayvat, S. F. Hasanm e S. C. Mukhopadhyay: *Activity and anomaly detection in smart home: A survey.* Em *Smart Sensors, Measurement and Instrumentation*, páginas 191–220. Springer International Publishing, julho 2015. `https://doi.org/10.1007/978-3-319-21671-3_9`. 13

[24] Wang, Jindong, Yiqiang Chen, Shuji Hao, Xiaohui Peng e Lisha Hu: *Deep learning for sensor-based activity recognition: A survey.* Pattern Recognition Letters, 119, julho 2017. 13

[25] Jobanputra, Charmi, Jatna Bavishi e Nishant Doshi: *Human activity recognition: A survey.* Procedia Computer Science, 155:698–703, 2019. https://doi.org/10.1016/j.procs.2019.08.100. 13

[26] Kolkar, Ranjit e V. Geetha: *Human activity recognition in smart home using deep learning techniques.* Em *2021 13th International Conference on Information &amp Communication Technology and System (ICTS)*. IEEE, outubro 2021. https://doi.org/10.1109/icts52701.2021.9609044. 14, 20

[27] Anguita, Davide, Alessandro Ghio, Luca Oneto, Xavier Parra, Jorge Luis Reyes-Ortiz *et al.*: *A public domain dataset for human activity recognition using smartphones.* Em *Esann*, volume 3, página 3, 2013. 15

[28] Kwapisz, Jennifer R, Gary M Weiss e Samuel A Moore: *Activity recognition using cell phone accelerometers.* ACM SigKDD Explorations Newsletter, 12(2):74–82, 2011. 15

[29] Lima, Wesllen S., Eduardo Souto, Thiago Rocha, Richard W. Pazzi e Ferry Pramudianto: *User activity recognition for energy saving in smart home environment.* Em *2015 IEEE Symposium on Computers and Communication (ISCC)*. IEEE, julho 2015. https://doi.org/10.1109/iscc.2015.7405604. 15, 20

[30] Tapia, Emmanuel Munguia, Stephen S Intille e Kent Larson: *Activity recognition in the home using simple and ubiquitous sensors.* Em *Pervasive Computing: Second International Conference, PERVASIVE 2004, Linz/Vienna, Austria, April 21-23, 2004. Proceedings 2*, páginas 158–175. Springer, 2004. 15

[31] Suryadevara, Nagender Kumar e Subhas Chandra Mukhopadhyay: *Wireless sensor network based home monitoring system for wellness determination of elderly.* IEEE Sensors Journal, 12(6):1965–1972, junho 2012. https://doi.org/10.1109/jsen.2011.2182341. 15, 16, 20

[32] Marufuzzaman, Mohammad, Teresa Tumbraegel, Labonnah Farzana Rahman e Lariyah Mohd Sidek: *A machine learning approach to predict the activity of smart home inhabitant.* Journal of Ambient Intelligence and Smart Environments, 13(4):271–283, julho 2021. https://doi.org/10.3233/ais-210604. 16, 20

[33] Alam, M. R., M. B. I. Reaz e M. A. Mohd Ali: *SPEED: An inhabitant activity prediction algorithm for smart homes.* IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 42(4):985–990, julho 2012. https://doi.org/10.1109/tsmca.2011.2173568. 16

[34] Researchers MavHome: *MavHome::Research*, 2013. https://ailab.wsu.edu/mavhome/research.html, Acessado em 03/07/2023. 16

[35] Youngblood, G Michael, Diane J Cook e Lawrence B Holder: *The mavhome architecture.* Department of Computer Science and Engineering University of Texas at Arlington, Techinal Report, 33, 2004. 16

[36] Dinata, Ida Bagus Putu Peradnya e Bob Hardian: *Predicting smart home lighting behavior from sensors and user input using very fast decision tree with kernel density estimation and improved laplace correction*. Em *2014 International Conference on Advanced Computer Science and Information System*. IEEE, outubro 2014. https://doi.org/10.1109/icacsis.2014.7065885. 16, 20

[37] Cook, Diane J., Aaron S. Crandall, Brian L. Thomas e Narayanan C. Krishnan: *CASAS: A smart home in a box*. Computer, 46(7):62–69, julho 2013. https://doi.org/10.1109/mc.2012.328. 17

[38] PALA, Zeydin e Orhan ÖZKAN: *Artificial intelligence helps protect smart homes against thieves*. DÜMF Mühendislik Dergisi, maio 2020. https://doi.org/10.24012/dumf.700311. 17, 20

[39] Güven, C. T e M Aci: *Design and implementation of a self-learner smart home system using machine learning algorithms*. Information Technology and Control, 51(3):545–562, setembro 2022. https://doi.org/10.5755/j01.itc.51.3.31273. 17, 20

[40] Majeed, Rizwan, Nurul Azma Abdullah, Imran Ashraf, Yousaf Bin Zikria, Muhammad Faheem Mushtaq e Muhammad Umer: *An intelligent, secure, and smart home automation system*. Scientific Programming, 2020:1–14, outubro 2020. https://doi.org/10.1155/2020/4579291. 17, 18, 20

[41] Salhi, Lamine, Thomas Silverston, Taku Yamazaki e Takumi Miyoshi: *Early detection system for gas leakage and fire in smart home using machine learning*. Em *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, janeiro 2019. https://doi.org/10.1109/icce.2019.8661990. 18, 20

[42] Taiwo, Olutosin e Absalom E. Ezugwu: *Internet of things-based intelligent smart home control system*. Security and Communication Networks, 2021:1–17, setembro 2021. https://doi.org/10.1155/2021/9928254. 18, 20

[43] Casaccia, Sara, Luca Romeo, Andrea Calvaresi, Nicole Morresi, Andrea Monteriu, Emanuele Frontoni, Lorenzo Scalise e Gian Marco Revel: *Measurement of users' well-being through domotic sensors and machine learning algorithms*. IEEE Sensors Journal, 20(14):8029–8038, julho 2020. https://doi.org/10.1109/jsen.2020.2981209. 18, 20

[44] Lentzas, Athanasios e Dimitris Vrakas: *Machine learning approaches for non-intrusive home absence detection based on appliance electrical use*. Expert Systems with Applications, 210:118454, 2022. 19, 20

[45] Kelly, Jack e William Knottenbelt: *The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes*. Scientific Data, 2(1), março 2015. https://doi.org/10.1038/sdata.2015.7. 19

[46] Medico, Roberto, Leen De Baets, Jingkun Gao, Suman Giri, Emre Kara, Tom Dhaene, Chris Develder, Mario Bergés e Dirk Deschrijver: *A voltage and current measurement dataset for plug load appliance identification in households*. Scientific Data, 7(1), fevereiro 2020. https://doi.org/10.1038/s41597-020-0389-7. 19

[47] Solatidehkordi, Zahra, Jayroop Ramesh, A.R. Al-Ali, Ahmed Osman e Mostafa Shaaban: *An IoT deep learning-based home appliances management and classification system.* Energy Reports, 9:503–509, maio 2023. https://doi.org/10.1016/j.egyr.2023.01.071. 20

[48] Isikdag, Umit: *Internet of things: Single-board computers.* Em *Enhanced Building Information Models*, páginas 43–53. Springer International Publishing, 2015. https://doi.org/10.1007/978-3-319-21825-0_4. 22, 23

[49] Blair, Daniel: *Learning Banana Pi.* Packt Publishing, 2015. 23

[50] Balbinot, A e VJ Brusamarello: *Instrumentação e Fundamentos de Medidas*, volume 1. Rio de Janeiro: LTC/Grupo Gen, 2ª edição, 2012. 23

[51] Sehrawat, Deepti e Nasib Singh Gill: *Smart sensors: Analysis of different types of IoT sensors.* Em *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, abril 2019. https://doi.org/10.1109/icoei.2019.8862778. 23

[52] Xuyang, Liu, K.H. Lam, Prewitt K. Zhu, Chao Zheng, Xu Li, Yimeng Du, Liu Chunhua e Philip Pong: *Overview of spintronic sensors with internet of things for smart living.* IEEE Transactions on Magnetics, PP, julho 2019. 23

[53] Grinberg, Miguel: *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018. 25

[54] Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu e Xiaoqiang Zheng: *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. https://www.tensorflow.org/, Software available from tensorflow.org. 26

[55] Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden e Xiaoqiang Zhang: *Tensorflow: A system for large-scale machine learning.* maio 2016. 26

[56] McKinney Wes: *Data Structures for Statistical Computing in Python.* Em Walt Stéfan van der e Jarrod Millman (editores): *Proceedings of the 9th Python in Science Conference*, páginas 56 – 61, 2010. 27

[57] Armbian Team: *Armbian - Linux for ARM development boards*, 2023. https://www.armbian.com/, Accessado em 03/07/2023. 29

[58] Contribuidores Wikipedia: *Banana Pi — Wikipedia, The Free Encyclopedia*, 2023. `https://en.wikipedia.org/wiki/Banana_Pi`, Acessado em 03/07/2023. 31

[59] Contribuidores Arduino Forum: *MH-SR602 - high output signal time*, novembro 2020. `https://forum.arduino.cc/t/mh-sr602-high-output-signal-time/683630`, Acessado em 03/07/2023. 33

[60] Tasmota Team: *Tasmota - open source firmware for ESP devices*, 2023. `https://tasmota.github.io/docs/`, Acessado em 03/07/2023. 34

[61] Bergstra, James e Yoshua Bengio: *Random search for hyper-parameter optimization.* J. Mach. Learn. Res., 13:281–305, feb 2012, ISSN 1532-4435. 38

[62] Percival, Harry J W e Bob Gregory: *Architecture patterns with Python.* O'Reilly Media, Sebastopol, CA, março 2020. 39

Apêndice A

Notação

N	Número de amostras de um conjunto de dados
$\mathbf{x}$	Vetor de atributos
$\mathbf{w}$	Vetor de pesos
$f(\mathbf{x})$	Modelo preditivo aplicado a um vetor de atributos
$\mathbf{x}_i$	i-ésima amostra de um conjunto de dados
y_i	i-ésimo rótulo de um conjunto de dados
$x_i^{(j)}$	Valor do j-ésimo atributo da i-ésima amostra de um conjunto de dados
$d(\mathbf{x}_i, \mathbf{x}_k)$	Distância entre amostras $\mathbf{x}_i$ e $\mathbf{x}_k$