
UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MARIA JÚLIA DE OLIVEIRA VIEIRA

# ON-DEVICE MACHINE LEARNING FOR FORECASTING REFERENCE EVAPOTRANSPIRATION

Maceió

2024

MARIA JÚLIA DE OLIVEIRA VIEIRA

# ON-DEVICE MACHINE LEARNING FOR FORECASTING REFERENCE EVAPOTRANSPIRATION

Dissertation presented to the Postgraduate Program in Computer Science at the Federal University of Alagoas as a partial requirement for obtaining the title of Master in Computer Science.

Area of Concentration: Computer Systems Engineering.

Advisor: Prof.º Dr.º Erick de Andrade Barboza

Co-Advisor: Prof.º Dr.º David Bibiano Brito

Maceió-AL

2024

**Catalogação na Fonte**
**Universidade Federal de Alagoas**
**Biblioteca Central**
**Divisão de Tratamento Técnico**
Bibliotecário: Cláudio Albuquerque Reis – CRB-4 – 1753

V658o Vieira, Maria Júlia de Oliveira.
On-device machine learning for forecasting reference evapotranspiration / Maria Júlia de Oliveira Vieira. – 2024.
38 f. : il.

Orientador: Erick de Andrade Barboza.
Coorientador: David Bibiano Brito.

Dissertação (Mestrado em Engenharia de Sistemas Computacionais) – Universidade Federal de Alagoas. Instituto de Computação. Coordenação de Pós-Graduação em Informática. Maceió, 2024.

Bibliografia. f. 35-38.
Apêndice. f. 38.

1. Engenharia de sistemas de computação. 2. Sistemas embarcados (Computadores). 3. Aprendizado de máquina. 4. Eficiência energética. I. Título.

CDU: 004.41

**MINISTÉRIO DA EDUCAÇÃO**
UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
Av. Lourival Melo Mota, S/N, Tabuleiro do Martins, Maceió - AL, 57.072-970
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO (PROPEP)
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

## Folha de Aprovação

**MARIA JULIA DE OLIVEIRA VIEIRA**

# PREDIÇÃO DE EVAPOTRANSPIRAÇÃO DE REFERÊNCIA PARA PLANEJAMENTO DE IRRIGAÇÃO UTILIZANDO APRENDIZAGEM DE MÁQUINA EM SISTEMAS EMBARCADOS

# ON-DEVICE MACHINE LEARNING FOR FORECASTING REFERENCE EVAPOTRANSPIRATION

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 29 de outubro de 2024.

**Banca Examinadora:**

Documento assinado digitalmente
gov.br
ERICK DE ANDRADE BARBOZA
Data: 29/10/2024 11:43:42-0300
Verifique em https://validar.iti.gov.br

**Prof. Dr. ERICK DE ANDRADE BARBOZA**
UFAL – Instituto de Computação
**Orientador**

Documento assinado digitalmente
gov.br
TIAGO FIGUEIREDO VIEIRA
Data: 29/10/2024 14:40:48-0300
Verifique em https://validar.iti.gov.br

**Prof. Dr. TIAGO FIGUEIREDO VIEIRA**
UFAL – Instituto de Computação
**Examinador Interno**

Documento assinado digitalmente
gov.br
FERNANDO FRANCA DA CUNHA
Data: 29/10/2024 12:00:53-0300
Verifique em https://validar.iti.gov.br

**Prof. Dr. FERNANDO FRANÇA DA CUNHA**
UFV – Universidade Federal de Viçosa
**Examinador Externo**

# Acknowledgements

I am deeply thankful to my advisor, Prof.º Dr.º Erick Andrade Barboza, whose expertise, patience, and constructive criticism have guided me through every phase of this project. Beyond helping shape the direction of my research, your insightful mentorship has significantly enhanced my confidence in conducting independent research. Your belief in my abilities allowed me to push my limits and tackle challenges with greater determination, ultimately making me a more skilled and reflective professional. For that, I am sincerely grateful.

Special thanks to the faculty of PPGI at the Federal University of Alagoas for providing me with the resources and knowledge needed to complete this project. I am particularly grateful to Prof.º Dr.º Thiago Damasceno Cordeiro, whose encouragement and confidence in my abilities have been crucial to the completion of this work.

I would also like to thank my research group members, especially Emanuel Pereira. The collaboration we've shared has enriched my learning experience beyond measure.

To my family, thank you for your endless love and understanding. Especially my mother, Eliane Maria, you have been my pillar of strength throughout this long journey.

A heartfelt thanks to my friends, who have always been with me during my academic career and in life, for their continuous encouragement and understanding during the ups and downs of this journey. Special thanks to my friends, Bárbara Cotard and Laryssa Ribeiro. Without your unwavering support, this achievement would not have been possible. Your belief in me has kept me motivated.

# Dedication

*To all those who have inspired me to strive for excellence, this work is dedicated to you.*

# Resumo

O cenário climático-mundial atual demanda o uso dos recursos naturais de forma equilibrada, especialmente a água. A predição da evapotranspiração de referência ($ET_o$) pode auxiliar no gerenciamento dos recursos hídricos, sendo um parâmetro determinante na avaliação inicial da água para aplicações como programação de irrigação e estudos hidrológicos. A literatura mostra que modelos de Machine Learning (ML) têm sido amplamente aplicados na predição de $ET_o$ e têm mostrado resultados promissores, mesmo com o uso de parâmetros reduzidos. Trabalhos anteriores também apresentaram sistemas de irrigação automatizados que se conectam a serviços Web para prever os parâmetros necessários para fazer a programação de irrigação. No entanto, um sistema baseado em nuvem depende de uma infraestrutura de conexão à internet; tal infraestrutura nem sempre está disponível em áreas rurais. Além disso, um dispositivo que depende de uma conexão à internet geralmente consumirá mais energia. Portanto, esta pesquisa tem como objetivo principal desenvolver e avaliar um modelo TinyML projetado para prever $ET_o$, com o objetivo de estabelecer um modelo de machine learning energeticamente eficiente adequado para integração em um sistema embarcado. Avaliamos três modelos considerando temperatura do ar, umidade relativa e velocidade do vento como entradas. Os modelos foram incorporados em um microcontrolador ESP32 e comparados com suas versões em nuvem. Os resultados mostram que o modelo que estima $ET_o$ considerando apenas a temperatura do ar e a umidade é a melhor opção considerando o trade-off entre custo e precisão. Comparado com sua versão baseada em nuvem, este modelo incorporado consome 37,97% menos espaço de memória, usa aproximadamente 99,98% menos energia e roda 99,97% mais rápido.

**Palavras-chave** Eficiência energética, Sistemas embarcados, Eletrônica de baixa potência, TinyML, Aprendizagem de máquina.

# Abstract

The current climate-world scenario demands the use of natural resources in a balanced way, especially water. The prediction of reference evapotranspiration ($ET_o$) can assist in the management of water resources, being a determining parameter in the early assessment of water for applications such as irrigation scheduling and hydrological studies. The literature shows that Machine Learning (ML) models have been widely applied in the prediction of $ET_o$ and have shown promising results, even with the use of reduced parameters. Previous works also presented automated irrigation systems that connect to Web services to predict the parameters needed to make irrigation scheduling. However, a cloud-based system depends on an internet connection infrastructure; such infrastructure is not always available in rural areas. In addition, a device that relies on an internet connection will generally consume more power. Therefore, this research primarily aims to develop and assess a TinyML model designed to forecast $ET_o$, with the goal of establishing an energy-efficient machine learning model suitable for integration into an embedded system. We evaluated three models considering air temperature, relative humidity, and wind speed as inputs. The models were embedded in an ESP32 microcontroller and compared to its cloud versions. The results show that the model that estimates $ET_o$ considering only air temperature and humidity is the best option considering the trade-off between cost and precision. Compared to its cloud-based version, this embedded model consumes 37.97% less memory space, uses approximately 99.98% less energy and runs 99.97% faster.

**Keywords:** Power efficiency, Embedded systems, Low power eletronics, TinyML, Machine learning

# List of Figures


1 The figure shows the process of evaporation and transpiration occurring simultaneously in the system presented. Source: Author. . . . . 7

2 Reference Evapotranspiration ($ET_o$). Source: Author. . . . . 8

3 Machine Learning Applications. Souce: Authors. . . . . 10

4 General architecture of the artificial neural networks used in the study of (FERREIRA *et al.*, 2019). Source: (FERREIRA *et al.*, 2019) . . . . . 11

5 Composition of TinyML. Source (RAY, 2022) . . . . . 13

6 Taxonomy of the main TinyML applications. Source: (ABADADE *et al.*, 2023). . . . . 14

7 Meteorological stations in Brazil monitored by INMET. The map comprehends conventional stations, automatic stations and rainfall stations. Source: INMET. . . . . 21

8 Description of the components of the ESP32 board manufactured by Espressif (Source: (Espressif Systems (Shanghai) Co., 2024)). . . . . 26

9 Tools used to measure the power consumption of the ESP-32 board. Source: Author. . . . . 27

10 Print screen of the Nordic Power Profile Software used to measure the power consumption of the ESP-32 board for the model with only temperature as input. Source: Author. . . . . 28

11 Architectures utilized to assess the performance of the embedded models. On the left is the architecture of the embedded models, while on the right is the architecture of the Cloud Model. Source: Author. . . . . 29

12 Violin plot for the mean values comparing the performance of the models running on a PC and the embedded models. Source: Author. . . . . 30

# List of Tables


1 Mean values of Root Mean Square Error (RMSE), $R^2$ Score and Mean Absolute Percentage Error (MAPE) for the proposed ANN models and a reference ANN model proposed in (FERREIRA *et al.*, 2019). . . . . 38

2 Values of the memory occupied only by the model, memory occupied by the complete embedded program in the ESP32 storage, the time to run 1,000 inferences and the energy consumption to run 1,000 inference for the embedded models and the “Cloud Model". . . . . 38

3 Parameters used for training the ANN model using TensorFlow library and Optuna Optimization Framework. . . . . 38

# Glossary

**mm d$^{-1}$** Millimeters per day, the unit of measure for reference evaporation.

**Reference Evapotranspiration ($ET_o$)** The rate of evapotranspiration from a hypothetical grass surface under ideal conditions.

**TensorFlow** An open-source machine learning framework developed by Google for numerical computation and machine learning tasks.

**TensorFlow Lite** A lightweight version of TensorFlow designed for mobile and embedded devices.

**TinyML** A subfield of machine learning focused on deploying models on resource-constrained embedded devices.

# Summary

**1 Introduction** **1**

1.1 Motivation . . . . . 1

1.2 Research questions . . . . . 4

1.3 General and Specific Objectives . . . . . 4

1.4 Structure of the work . . . . . 4

**2 Fundamentals** **6**

2.1 Evapotranspiration . . . . . 6

2.1.1 Reference Evapotranspiration . . . . . 7

2.2 Machine Learning . . . . . 9

2.2.1 Supervised Learning . . . . . 10

2.2.2 Unsupervised Learning . . . . . 11

2.2.3 Evaluation metrics for regression models . . . . . 11

2.3 Embedded Systems: TinyML . . . . . 12

**3 Related works** **15**

3.1 Prediction of reference evapotranspiration values using Machine Learning techniques . . . . . 15

3.2 Automated irrigation systems based on IoT . . . . . 17

3.3 Machine learning models on embedded systems . . . . . 18

3.4 Relevance of the proposal . . . . . 19

**4 Materials and methods** **20**

4.1 Dataset and preprocessing . . . . . 20

4.2 Models creation . . . . . 22

4.3 Models training and test . . . . . 23

4.4 Model evaluation . . . . . 24

4.5 Embedding the models . . . . . 24

4.6 Evaluating the embedded models . . . . . 25

**5 Results and discussion** **27**

5.1 Model Comparison . . . . . 27
5.2 Performance of embedded models . . . . . 30
5.3 Discussion . . . . . 31

**6 Conclusion** **33**

**References** **35**

# 1 Introduction

This chapter describes the motivations for the development of the study to create an embedded machine learning model to estimate reference evapotranspiration.

## 1.1 Motivation

The climate situation in the current world scenario demands the use of natural resources in a balanced way. The water demand in Brazil, for example, has undergone several transformations throughout its history. The demand in the country has been continuously growing over the years, with an emphasis on supplying cities, the manufacturing industry, and irrigated agriculture. Between 2022 and 2040, an increase of approximately 30% in water withdrawals is estimated, representing an expansion in the use of 1 trillion and 290 billion liters of water on average per year (ANA, 2024). Agriculture, and especially irrigated agriculture, is the sector with the highest consumptive water use and water withdrawal [1].

Soil water balance is used to estimate the total available soil water and, indirectly, a crop's water use. The amount of available soil water varies during the season, adding eventual rainfall and / or irrigation water inputs, and finally subtracting the output related to evapotranspiration (ET) (COSTA *et al.*, 2018). ET contains two processes, evaporation from surfaces of soil and plants, and transpiration from crops to the atmosphere (CHEN *et al.*, 2020). It is critical to predict ET accurately to design irrigation plans and improve the water resource use efficiency (FERREIRA *et al.*, 2019).

The evapotranspiration rate from a reference surface, not short of water, is called the reference crop evapotranspiration or reference evapotranspiration and is denoted as $ET_o$ (ALLEN *et al.*, 1998). To calculate $ET_o$, the Food and Agriculture Organization of the United Nations (FAO) recommends the use of the Penman-Monteith method (FAO56). FAO56 also requires a large number of meteorological variables, such as the maximum and minimum air temperature, relative humidity, wind speed, and solar radiation. These variables are often difficult to obtain, as the necessary sensors are very expensive, making the calculation of $ET_o$ difficult (DIAS *et al.*, 2021). Improving the precision of the estimation of $ET_o$ plays a vital role in computing ET and therefore, increasing the irrigation efficiency and agricultural

[1] https://www.fao.org/aquastat/en/data-analysis/irrig-water-use

water reuse (CHEN *et al.*, 2020).

The lack of available meteorological data and infrastructure opened an opportunity for new fields of research. The literature shows that machine learning (ML) models have been widely applied to predict $ET_o$ and water demand (DIAS *et al.*, 2021; FERREIRA; CUNHA, 2020; SIDHU *et al.*, 2020). Some studies seek to simplify $ET_o$ estimation by using a smaller number of meteorological variables. In (FERREIRA *et al.*, 2019), the authors evaluated the effectiveness of alternative formulas compared to ML models in estimating daily values of $ET_o$ in Brazil with limited meteorological data. The machine learning models showed better performance compared to the alternative equations evaluated. This study also points out that the use of relative humidity in addition to temperature increased the generalization capacity of the models and provided performance gains for all evaluated models, representing an option to improve the estimation of $ET_o$ at a low additional cost.

The research conducted by (FAROOQUE *et al.*, 2022) examined three deep learning architectures, Long Short Term Memory (LSTM), 1D Convolutional Neural Networks (1D-CNN), and Convolutional LSTM (ConvLSTM), for $ET_o$ prediction daily. The relative importance method was used to identify the most relevant input variables for these models. For calibration and validation evaluation in annual daily $ET_o$ forecasts, the hybrid ConvLSTM model recorded errors lower than CNN and LSTM with the lowest RMSE for calibration and validation.

Moreover, the application of the predicted $ET_o$ alongside other meteorological data using ML techniques for irrigation decision and scheduling can also be observed in the literature (NAGAPPAN *et al.*, 2020; JIA *et al.*, 2023; CHEN *et al.*, 2021; LORITE *et al.*, 2015).

In addition, it has been shown that the advancement of automated irrigation systems, which are linked to Web services using the Internet of Things (IoT), can predict the parameters essential for making irrigation decisions (NAWANDAR; SATPUTE, 2019). (BU; WANG, 2019) presents a smart agriculture IoT system where deep reinforcement learning is integrated into the cloud layer to enable real-time intelligent decision-making, such as calculating the optimal water required for irrigation to enhance the crop growth environment. The research carried out by (GOAP *et al.*, 2018) presents an intelligent system based on open source technology to predict the irrigation requirements of a field using the detection of soil

parameters together with data from the Internet to represent weather forecast.

These past studies presume adequate network coverage and reliable internet access at the site where the system operates, ensuring communication between edge devices and either the *Cloud* or a local server. The work of (MONTELEONE *et al.*, 2019) investigates the variables that affect the intention to adopt Precision Agriculture for smart water management in the context of Agriculture 4.0 by developing a conceptual model and performing a systematic review of the literature on the theme. They concluded that the regional telecommunications infrastructure, especially the precariousness of Internet access, is one of the factors with the greatest impact on the process of adopting information technology (IT) in rural areas. Therefore, creating a solution that does not depend on communication infrastructure can expand the application of solutions using IoT and ML in a wider range of locations.

One way to use ML without depending on the network communication is to embed the ML model in the device placed in the field. Furthermore, embedding the ML model offers several benefits, including reduced energy consumption, reduced latency, efficient bandwidth utilization, improved data security, improved privacy, and cost savings. This enables IoT devices to function reliably without continuous dependence on cloud services while providing accurate ML solutions, making it an attractive option for IoT applications seeking more cost-effective results. The effort to embed ML models in resource constraint devices is called TinyML (DUTTA; BHARALI, 2021).

Recent works have used the TinyML concept to enable the use of ML models without relying on a network connection. (PEREIRA *et al.*, 2024), for example, presents an embedded system that uses a random forest model to classify the potability of water in a ESP32 microcontroller independently of an Internet connection. (HONG *et al.*, 2023) investigates the prediction of wind speed by applying an artificial neural network (ANN) model to an ESP32 achieving low-latency data analysis.

Running a machine learning model directly on an embedded device offers the benefit of eliminating reliance on a communication infrastructure, thereby simplifying implementation and reducing overall costs. We believe that this work can help to develop irrigation systems that are more energy efficient and more accessible to regions with limited technological resources, such as rural areas in developing nations.

## 1.2 Research questions

- **RQ1:** Is it possible to embed the regression model that estimates reference evapotranspiration values without losing performance?
- **RQ2:** What are the differences in terms of solution quality between the embedded machine learning models and the cloud model?
- **RQ3:** What are the main advantages and limitations of adopting the reference model of the literature?

## 1.3 General and Specific Objectives

The main objective of this study is to propose an energy-efficient ML model to predict $ET_o$ using minimal meteorological data that can be used in an embedded system. The specific contributions of this study are the following:

1. Propose an ML model that predicts $ET_o$ in an embedded scenario.
2. Perform and evaluation of the proposed model with the one implemented on a machine with higher computational resources.
3. Evaluate the proposed model energy consumption, memory footprint, and time to run inferences compared to a model running on a cloud-based scenario.

## 1.4 Structure of the work

This work is organized as follows. Section 2 introduces key concepts essential to understanding the research. It begins with a discussion on evapotranspiration, explaining its significance in environmental and agricultural contexts. The section then covers the basics of machine learning, detailing its various techniques and their relevance to predicting evapotranspiration. Lastly, the fundamentals of TinyML are explored, focusing on its application in resource-limited environments.

In section 3, previous studies related to the prediction of $ET_o$ using machine learning techniques are reviewed. It also covers the development of automated irrigation systems

based on IoT, as well as the implementation of machine learning models on embedded systems. This section highlights the relevance of the current proposal within the landscape of technological advancements in agriculture.

Section 4 outlines the dataset used for the research and the preprocessing techniques applied. It details the process of creating and training machine learning models, along with the evaluation methods used to assess their performance. Additionally, the section describes how the models were embedded into hardware and the process of evaluating their real-world performance.

In section 5, the model's performance is compared and analyzed, with a particular focus on the embedded model's efficiency and effectiveness. This section provides a comprehensive discussion on the findings, examining both the successes and limitations of the proposed solutions. Finally, section 6 summarizes the key findings of the research, reflects on its contributions to the field, and proposes potential directions for future work.

# 2 Fundamentals

This section covers the basics of evapotranspiration and its importance, alongside an introduction to machine learning techniques and embedded approaches such as TinyML.

## 2.1 Evapotranspiration

Evaporation is the process by which liquid water is converted to water vapor (vaporization) and removed from the evaporating surface (vapor removal). Water evaporates from a variety of surfaces, such as lakes, rivers, pavements, soils, and moist vegetation. Transpiration consists of the vaporization of liquid water contained in plant tissues and the removal of the vapor to the atmosphere. Transpiration, like direct evaporation, depends on energy input, vapor pressure gradient, and wind. Therefore, the terms radiation, air temperature, relative humidity, and wind speed must be considered when evaluating transpiration (ALLEN *et al.*, 1998). Evaporation and transpiration occur simultaneously and there is no easy way to distinguish between the two processes. In addition to the availability of water in the topsoil, evaporation from cultivated soil is mainly determined by the fraction of solar radiation reaching the soil surface.

Evapotranspiration (ET) is typically modeled using meteorological data and algorithms that describe the surface energy and aerodynamic characteristics of vegetation. ET is typically measured using systems that require the use of relatively complex physical principles and techniques. In many agricultural systems, plant density, height, vigor, and water availability are generally uniform, and the application of estimation algorithms and measurement of ET can be relatively straightforward, although it still presents substantial challenges (ALLEN *et al.*, 2011).

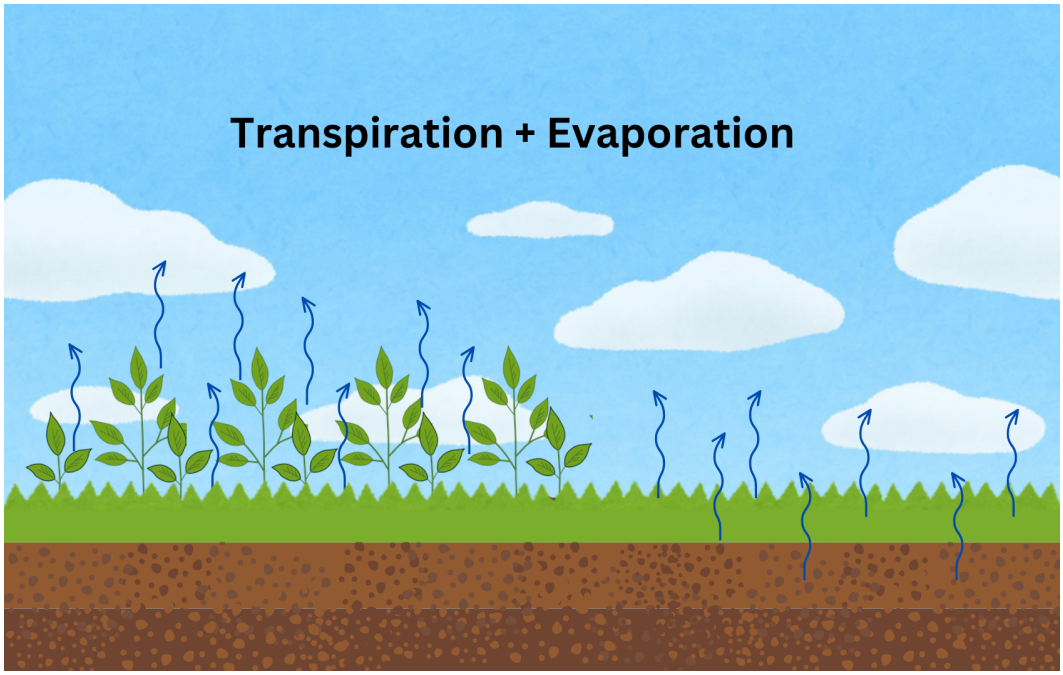


Figura 1: The figure shows the process of evaporation and transpiration occurring simultaneously in the system presented. Source: Author.

### 2.1.1 Reference Evapotranspiration

The evapotranspiration rate from a reference surface, not short of water, is called the reference crop evapotranspiration or reference evapotranspiration and is denoted as $ET_o$. The reference surface is a hypothetical grass reference crop with specific characteristics, as we can observe in Figure 2. ETo is a climatic parameter and can be computed from weather data, expresses the evaporating power of the atmosphere at a specific location and time of the year and does not consider the crop characteristics and soil factors (ALLEN *et al.*, 1998). $ET_o$ can be determined using lysimeters, structures designed to accurately measure precipitation, evaporation, and drainage events, and are typically used in the development and validation of other methods. Given the cost and complexity of lysimeters, their use is typically restricted to research (JIA *et al.*, 2023).

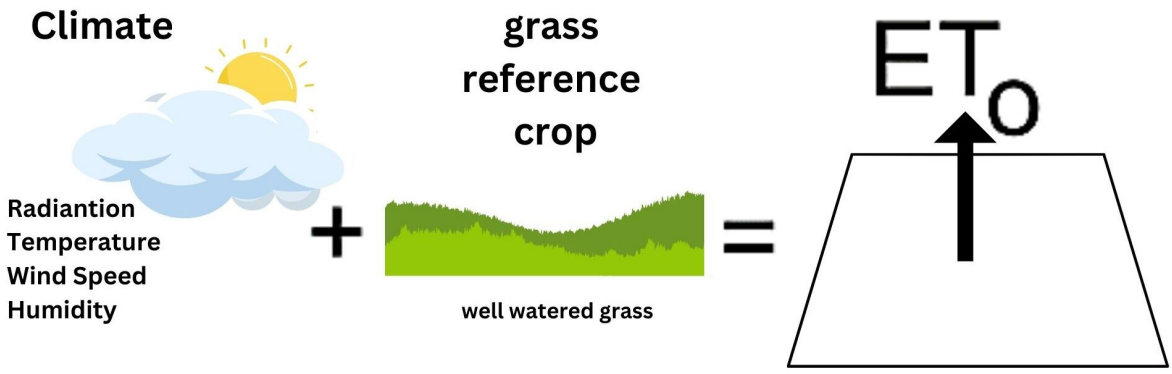


Figura 2: Reference Evapotranspiration ($ET_o$). Source: Author.

Estimating $ET_o$ through mathematical models is the most common and usual process; however, they present accuracy problems, mainly due to the lack of adjustment of their respective coefficients, reading errors, sensor accuracy and because they are developed for specific climatic and agronomic conditions (MORAIS *et al.*, 2015). Therefore, the Food and Agriculture Organization of the United Nations (FAO) and the International Commission on Irrigation and Drainage (ICID) in their bulletin 56, standardized the Penman-Monteith method, making it the standard model for estimating $ET_o$, from meteorological data for its application (ALLEN *et al.*, 1998). The FAO Penman-Monteith method is recommended as the only $ET_o$ method for determining reference evapotranspiration.

$$ET_o = \frac{0.408\Delta(R_n - G) + \gamma \frac{900}{T_{avg}+273} u_2 (e_s - e_a)}{\Delta + \gamma(1 + 0.34u_2)}, \quad (1)$$

$ET_o$ is the reference evapotranspiration ($mmd^{-1}$); $\Delta$ is the slope of the water vapor pressure curve (kPa $°C^{-1}$); $R_n$ is the net solar radiation ($MJm^{-2}d^{-1}$); G is the soil heat flux ($MJm^{-2}d^{-1}$), equal to 0 for daily estimates; $\gamma$ is the psychrometric constant (kPa $°C^{-1}$); $T_{avg}$ daily average air temperature (°C); $u_2$ wind speed measured at 2 m height ($ms{-}1$); $e_a$ the partial pressure of water vapor; $e_s$ water vapor saturation pressure; $(e_s - e_a)$ is the water vapor pressure deficit (kPa).

The scarcity of available water resources and climate change are the main factors affecting agricultural irrigation. To improve the productivity by water use, it is necessary to predict crop water needs in advance (JIA *et al.*, 2023). Where water resources are limited and seriously threatened by overexploitation, it is extremely important to estimate crop water

demand more accurately (MORAIS *et al.*, 2015).

In this way, good management and planning of available water resources can be achieved. One of the main water balance parameters used to determine the crop's water needs is ET.

## 2.2 Machine Learning

Machine Learning (ML) is one of the features of Artificial Intelligence (AI) and can be broadly defined as computational methods that use experience to improve performance, make accurate predictions, identify patterns, and make decisions. It can also be understood as a set of computational techniques that use experience to improve performance or make accurate predictions. In this context, experience refers to the information available to the machine, usually in the form of previously collected data.

Machine learning supports a very broad set of practical applications, which include text or document classification, natural language processing (NLP), speech processing applications, computer vision applications, among others (MOHRI *et al.*, 2018). We can classify ML models into three main categories: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

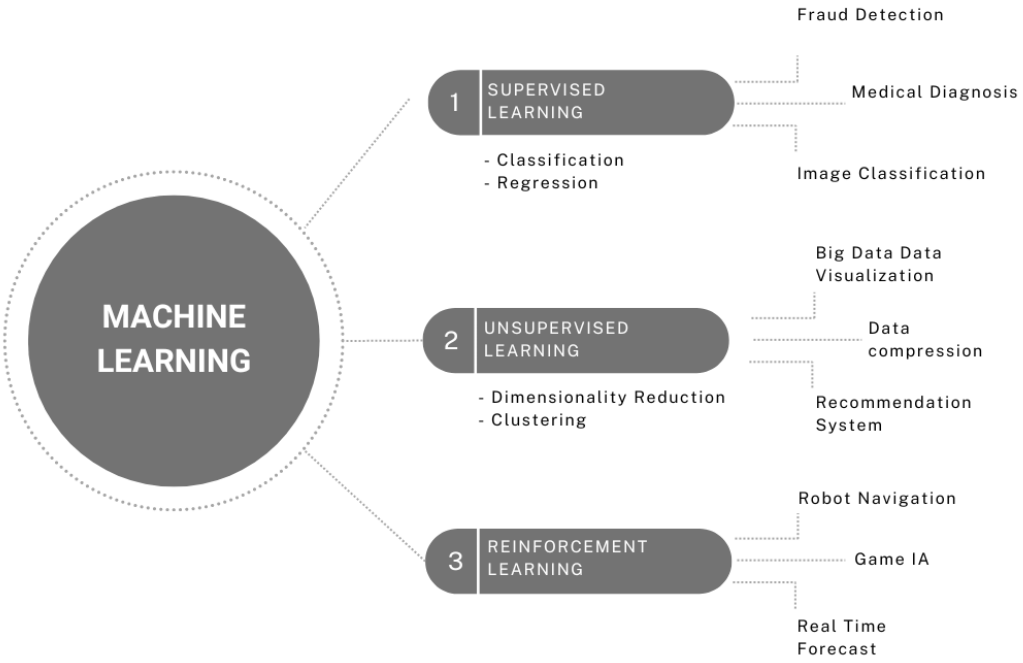


Figura 3: Machine Learning Applications. Souce: Authors.

### 2.2.1 Supervised Learning

Supervised Learning establishes a process that relates an output to an input based on labeled data, and continuously adjusts the predictive model until the model's predicted results achieve an expected accuracy. Examples of problems that fall under this category of learning are Classification and Regression. Examples of common supervised learning algorithms include Support Vector Machine (SVM), Artificial Neural Network (ANN), Decision Trees, Bayesian Classification, Least Squares Regression, Logistic Regression, etc.

An ANN is an information processing system that simulates the human brain's ability to find patterns and learn through trial and error, typically consisting of a series of processing elements called neurons, which are interconnected by synaptic weights. The most common architecture of an ANN consists of an input layer, where data is fed into the ANN, hidden layer(s) where the data is processed, and an output layer, where the results are produced (FERREIRA *et al.*, 2019). An example can be seen in Figure 4.

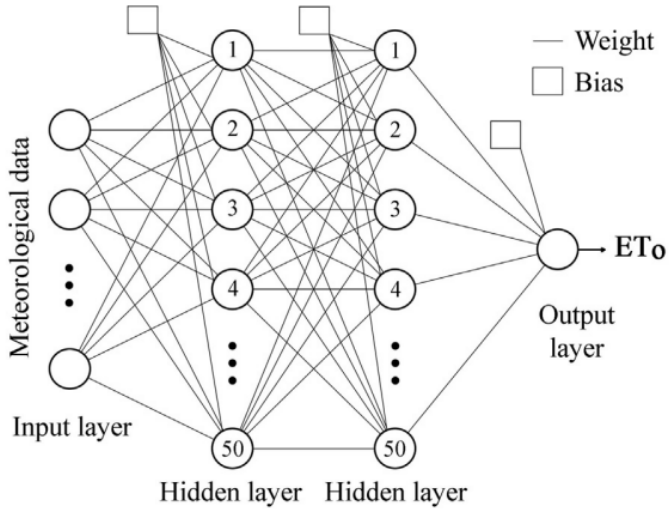


Figura 4: General architecture of the artificial neural networks used in the study of (FERREIRA *et al.*, 2019). Source: (FERREIRA *et al.*, 2019)

### 2.2.2 Unsupervised Learning

In the Unsupervised Learning, the input data is unlabeled, and algorithms must infer the intrinsic connections of the data, such as clustering and association rule learning, to discover new patterns in the data. The most common unsupervised learning task is Clustering. An algorithm for this type of learning includes K-Means, whose use is quite common in clustering tasks. The goal of the K-means algorithm is to divide n objects into k groups, with the result being that each object belongs to the closest group (FERREIRA *et al.*, 2019). Another example is principal component analysis (PCA), which can be used to reduce the dimensionality of data by finding the most important features that explain most of the variation in the data.

### 2.2.3 Evaluation metrics for regression models

Evaluation metrics or performance metrics, are crucial components of regression analysis and machine learning-based prediction models. They serve as tools to quantify the model's effectiveness and accuracy. The calculation of these metrics is crucial in the development of a model and is typically performed at the end of the training process, using a test dataset that the model has not encountered during training.

The Mean Squared Error (MSE) is a popular regression-related metric related to the

average squared error between the predicted and actual values. It takes positive or zero values and is given by

$$MSE = \frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N} \tag{2}$$

where $y_i$ is the predicted value, $\hat{y}_i$ is the observed value, and N is the number of observations.

One major disadvantage of MSE is that it is not robust to outliers. In case a sample has an associated error, with values larger than the one of other samples, the square of the error will be even larger. This, paired with the fact that MSE calculates the average of errors, makes MSE prone to outliers (PLEVRIS *et al.*, 2022).

The Root Mean Squared Error (RMSE) is a frequently used measure of the differences between values, predicted by a model, or an estimator and the values observed. It is the square root of MSE. Unlike MSE, RMSE provides an error measure in the same unit as the target variable.

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{3}$$

By expressing the error as a percentage, the Mean Absolute Percentage Error, one can have a better understanding of how off the predictions are in relative terms. It is given by:

$$MAPE = \frac{100\%}{N}\sum_{i=0}^{N-1}\frac{y_i - \hat{y}_i}{y_i} \tag{4}$$

The Coefficient of Determination ($R^2$) measures how closely predicted values match actual values, indicating the model's accuracy during training. In nonlinear models, such as ANN-based predictions, $R^2$ is calculated as the ratio of the variance explained by the model to the total variance. Unlike in linear models, $R^2$ can take negative values, which indicates that the model fits the data poorly, rather than being constrained between 0 and 100%.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{5}$$

## 2.3 Embedded Systems: TinyML

The rapid growth in miniaturization of low-power embedded devices and advancement in the optimization of ML algorithms have opened up a new prospect of the Internet

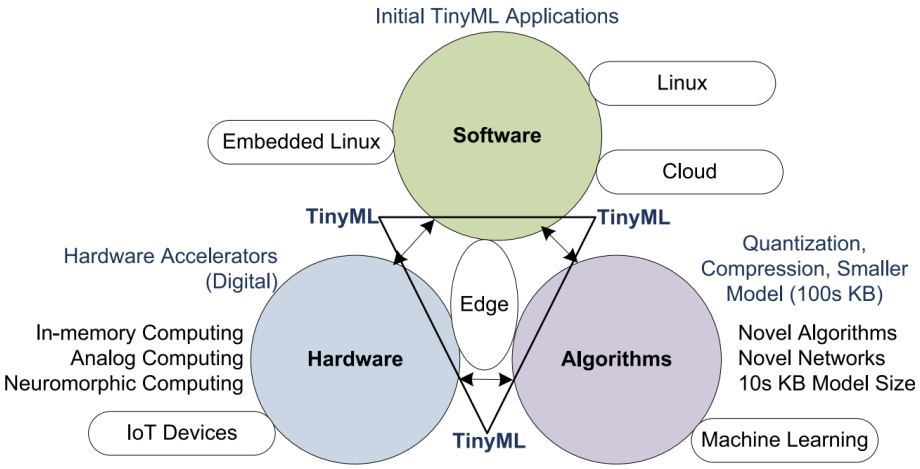


Figura 5: Composition of TinyML. Source (RAY, 2022)

of Things (IoT), tiny machine learning (TinyML), which calls for implementing the ML algorithm within the IoT device (DUTTA; BHARALI, 2021).

TinyML focuses on deploying compressed and optimized machine learning models on tiny, low-power devices such as battery-powered microcontrollers, and embedded systems (ABADADE *et al.*, 2023). TinyML framework in IoT is aimed to provide low latency, effective bandwidth utilization, strengthen data safety, enhance privacy, and reduce cost. Its ability to empower the IoT device to reliably function without consistent access to the cloud services while delivering accurate ML services makes it a promising option for IoT applications seeking cost-effective solutions (DUTTA; BHARALI, 2021).

TinyML system must accommodate the following requirements, (i) energy-harvesting edge devices for running learning models, (ii) enables battery-operated embedded edge devices, (iii) scalability to trillions of sensors enabled cheap embedded devices, and (iv) codes that can be stored within a few KB in the on-device RAM (RAY, 2022).

TinyML can be envisaged as the composition of three key elements (i) software, (ii) hardware, and (iii) algorithms. TinyML can be accommodated in Linux, embedded Linux, and cloud-based software where initial TinyML applications can be run. The hardware can comprise IoT devices with or without hardware accelerators. Such devices can be based on in-memory computing, analog computing, and neuromorphic computing for better learning experience (RAY, 2022).

(DUTTA; BHARALI, 2021) built a taxonomy of TinyML applications that have been

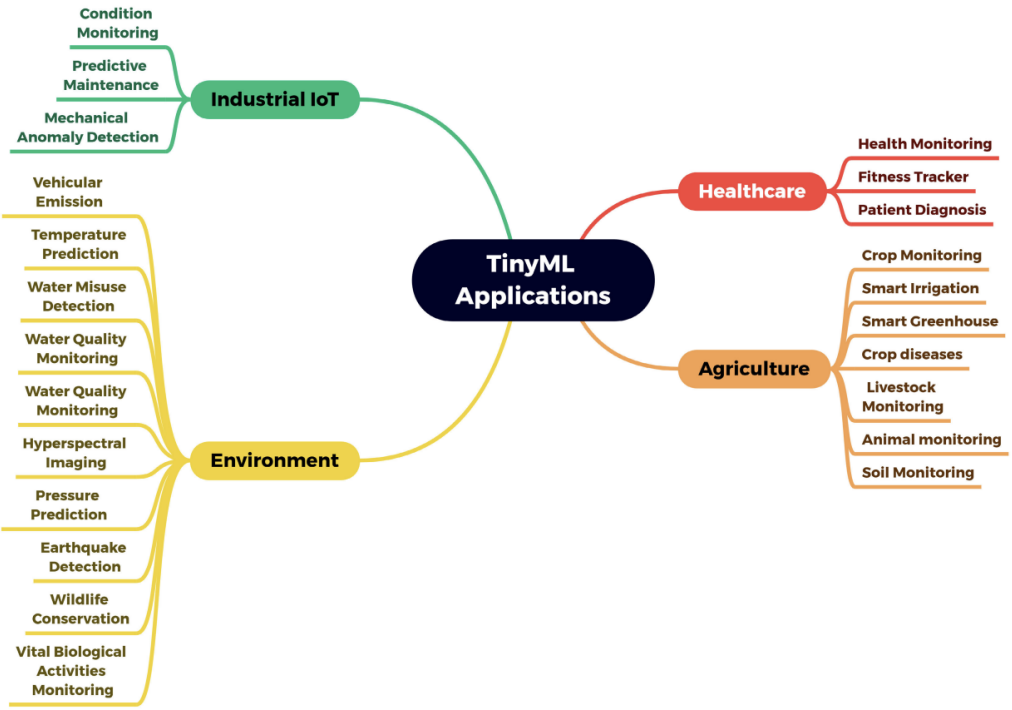


Figura 6: Taxonomy of the main TinyML applications. Source: (ABADADE *et al.*, 2023).

used to bring new solutions to various domains, Figure 6, such as healthcare, smart farming, environment, and anomaly detection.

TinyML can be applied to a variety of use cases, such as image recognition, speech recognition, sign language processing, phenomics, face detection, hand gesture recognition, few-shot keyword spotting, precision agriculture, body pose estimation, ecological monitoring, traffic management, environmental prediction, respiratory symptom detection, and autonomous vehicle monitoring, among others.

# 3 Related works

This section reviews existing literature and research related to the prediction of evapotranspiration using machine learning techniques. It examines the development of automated irrigation systems utilizing IoT, which aims to optimize water usage. Furthermore, it explores previous work applying machine learning models to embedded systems. The review emphasizes the gap in existing research and the relevance of the proposed solution to address these challenges.

## 3.1 Prediction of reference evapotranspiration values using Machine Learning techniques

The work by (FAROOQUE *et al.*, 2022) presents three deep learning models, including Long Short Term Memory (LSTM), 1D Convolutional Neural Networks (1D-CNN), and Convolutional LSTM (ConvLSTM), were tested to predict $ET_o$ on a daily time scale for a seven-day period. Daily air temperature, maximum, minimum, and mean values, solar radiation, relative humidity, and wind speed data were collected. The relative importance method was used to determine the most suitable input variables for the models. The deep learning models were evaluated with the Walk-Forward Analysis (WFA) cross-validation technique using statistical measures of Root Mean Square Error (RMSE) and Coefficient of Determination ($R^2$).

The modified FAO Penman–Monteith equation (FAO-56) equation was used as a reference method for comparison purposes. For calibration and validation assessment on annual daily $ET_o$ forecasts, the hybrid ConvLSTM model recorded lower errors than CNN and LSTM with the lowest daily calibration and validation RMSE of 0.64 and 0.62, 0.81 and 0.81, and 0.81 and 0.70 mm/day for the meteorological stations used. The robustness and accuracy of these forecast models can help farmers, water resource managers, and irrigation planners with improved and sustainable water management at the basin level, and for irrigation scheduling at the farm or field level.

The work (FERREIRA *et al.*, 2019) evaluated the performance of alternative equations compared to ML models to estimate daily $ET_o$ values throughout Brazil using daily data from meteorological stations of the National Institute of Meteorology (INMET), available in

the Meteorological Database for Education and Research (BDMEP) that measure temperature and relative humidity or only temperature. Two strategies, not yet used in Brazil, were used to develop the Artificial Neural Network (ANN) models, of the multilayer feed-forward type, and Support Vector Machine (SVM): (i) the definition of groups of meteorological stations with similar climatic characteristics, using the K-Means clustering algorithm, to develop specific models for each group; and (ii) the addition of previous meteorological data as input to the models. Six empirical equations commonly reported in the literature were used, as well as the FAO-56 to estimate data unavailable in the database. This approach was used to estimate $ET_o$ with only measured air temperature (PMT) and temperature and relative humidity (PMRH) data.

The ANN and SVM models showed superior performance to the alternative equations studied, even when calibrated. The evaluated strategies, clustering and data from previous days, provided considerable performance gains. For the temperature-based models, the best performance was obtained by the ANN developed with the clustering strategy and using data from two previous days as input. However, due to similar performance and greater generalization capacity, the ANN developed without clustering and using data from four previous days is recommended. For the models based on temperature and relative humidity, the ANN developed with data from four previous days was the best option.

Many types of artificial intelligence models have been applied to predict $ET_o$, however, there is still little literature on the application of hybrid models for parameter optimization of deep learning models (JIA *et al.*, 2023). The work of (JIA *et al.*, 2023) investigates the use of Long Short Term Memory (LSTM) to predict daily ETo. Two LSTM models of different topology were built and optimized using the hyperparameters of the particle swarm optimization (PSO) algorithm in the LSTM neural network. The accuracy of the two hybrid models was evaluated using four different data sets at the stations. The result showed that the optimized model has good accuracy for prediction, where the optimized hybrid models were also applied to different datasets, it can be found that the optimized hybrid model have better accuracy. In addition, the hybrid models developed in this study do not rely on external data, only needing data measured at a local meteorological station.

The (NAGAPPAN *et al.*, 2020) study, conducted at Veeranam Tank, India, determines the multivariate analysis of correlated variables involved in the estimation and modeling

of $ET_o$ from 1995 to 2016. A reduced-feature data model was constructed with the most significant variables of the model extracted by Principal Component Analysis (PCA). This work also explores the effectiveness of a Deep Learning Neural Network (DLNN) with the reduced feature model in predicting ETo compared to the conventional FAO-56 equation and Radial Basis Function Network (RBFNN) as a baseline machine learning method. The dimensionality of the input variables was reduced from six to three most significant variables in modeling $ET_o$. Among the machine learning methods, DLNN proved to be effective in predicting $ET_o$ with the reduced feature data model.

## 3.2 Automated irrigation systems based on IoT

The paper (GOAP *et al.*, 2018) presents an intelligent system based on open-source technology to predict the irrigation requirements of a field using the detection of soil parameters such as soil moisture, soil temperature, and environmental conditions along with the weather forecast data from the Internet.

An algorithm based on a combination of supervised and unsupervised machine learning techniques has been developed using Support Vector Regression (SVR) and K-Means clustering method for estimation of difference change in soil moisture due to weather conditions. The proposed algorithm is better compared to the SVR-based approach only. Due to higher accuracy and minimum mean squared error (MSE) function, the hybrid machine learning algorithm based on SVR and K-Means has been used in the irrigation planning module.

On the hardware side, a water pump is connected to a relay switch controlled by a Wi-Fi enabled node. The node is controlled by a web service through a real-time monitoring interface trigger. Using this interface, the water pump is remotely managed in manual and automatic modes. A WiFi module is used to send the data to the server. In the Wireless Sensor Network (WSN) scenario, the ZigBee network is used between the sensor node to the Gateway node and then a WiFi module or mobile data communication module sends the data from the Gateway node to the server.

Smart agriculture systems based on Internet of Things are the most promising to increase food production and reduce the consumption of resources like fresh water. In (BU; WANG, 2019) study, they presented a smart agriculture IoT system based on deep reinforce-

ment learning. They design a smart agriculture IoT system that is made of four layers from bottom to up including the agricultural data collection layer, edge computing layer, data transmission layer, and cloud computing layer. As a crucial component, deep reinforcement learning is deployed in the cloud layer for making immediate smart decisions. This work presents several representative deep reinforcement learning models that have the potential to be used in building smart agriculture systems, also expected to promote the development of smart agriculture and contribute to increasing food production.

(NAWANDAR; SATPUTE, 2019) work proposed targets to develop a low cost intelligent system for smart irrigation. It uses IoT to make devices used in the system to talk and connect on their own, with capabilities like: admin mode for user interaction, one-time setup for irrigation schedule estimation, neural based decision making for intelligent support and remote data monitoring. A sample crop test-bed was chosen to present the results of the proposed system, which includes an irrigation schedule, neural net decision-making, and remote data viewing. The neural network provides the required intelligence to the device that considers current sensor input and masks the irrigation schedule for efficient irrigation. The system uses MQTT and HTTP to keep the user informed about the current crop situation even from a distant location. The proposed system proves beneficial with its intelligence, low cost, and portability, making it suitable for greenhouses, farms, etc.

## 3.3 Machine learning models on embedded systems

Recent works have used the TinyML concept to enable the use of ML models without relying on a network connection. (PEREIRA *et al.*, 2024), for example, presents an embedded system that uses a random forest model to classify the potability of water in a ESP32 microcontroller independently of an Internet connection. This study proposes an energy-efficient TinyML model for classifying water potability, using only parameters available through electronic sensing. The study evaluated performance using metrics such as Accuracy, Precision, Recall, F1- Score, memory occupied by the model, execution time, and energy consumption, comparing models developed with Random Forest and Neural Networks algorithms. It also assessed the best combination of model and adaptation library for the embedded system. The initial Machine Learning model, using Random Forest, demonstrated good performance, reaching a Precision of 0.70, and compared to its cloud-based counterpart, it

can operate for years on a standard battery power source.

The study of (HONG *et al.*, 2023) provides the investigation of wind speed forecasting using an Artificial Neural Network (ANN) and Linear Regression Model with an ESP32 chip. A portable wind speed prediction system will aid in mitigating the risks associated with sudden gusts of wind by forecasting the maximum wind speed that may occur shortly. This research also demonstrates the application of TinyML in the field of Artificial Intelligence (AI) by applying a Multiple Linear Regression (MLR) and ANN models to small, low-powered ESP32 microcontrollers to analyze data with low latency. The system predicts wind speed in Setapak, Kuala Lumpur, based on the measured temperature and humidity using a DHT22 sensor and displays forecast results and sensor readings on LCD screens. To measure the accuracy of the MLR and ANN models, the $R^2$, MSE, and RMSE between predicted and actual results are evaluated. Results indicate that the ANN model outperforms the MLR model for predicting wind speed.

## 3.4 Relevance of the proposal

The work of (GOAP *et al.*, 2018; NAWANDAR; SATPUTE, 2019; BU; WANG, 2019) refers to the use of a cloud-based system for the inferences from the intelligence models, not on the devices themselves, implying the need for an internet connection for the proposed systems to function. The work of (MONTELEONE *et al.*, 2019) concluded that the regional telecommunications infrastructure, especially the precariousness of Internet access, is one of the factors with the greatest impact on the process of adopting IT in rural areas. Running a machine learning model directly on an embedded device could offer this solutions the benefit of eliminating reliance on a communication infrastructure, thereby simplifying implementation and reducing overall costs.

Until now, to our knowledge, no other scientific work has presented a study of prediction of $ET_o$ running a machine learning model directly in an embedded scenario. Therefore, the main objective of this study is to propose an energy-efficient ML model to predict $ET_o$ using minimal meteorological data that can be used in an embedded system. We believe that this work can help to develop studies related to irrigation systems that are more energy efficient and more accessible to regions with limited technological resources, such as rural areas in developing nations.

# 4 Materials and methods

This section describes the methodology adopted to carry out this work. This chapter is divided into six subsections, each representing a stage of the work. The subsection 4.1 outlines the set of tasks undertaken to create and preprocess the dataset used in this study. Subsection 4.2 refers to the set of tasks performed and tailoring the machine learning model to predict $ET_o$ values. Subsection 4.3 outlines the steps taken to train and test the neural network models and assess their performance. Subsection 4.4 outlines the procedures followed to evaluate the machine learning models. Subsection 4.5 implements the first ANN models developed on an embedded device, evaluates its performance, and highlights the TinyML libraries used in the development of embedded models. The subsection 4.6 presents the steps taken to evaluate the embedded models.

## 4.1 Dataset and preprocessing

We used data from the Brazilian National Institute of Meteorology (INMET) weather stations, accessible through the Meteorological Database for Teaching and Research (BDMEP). The dataset comprised daily records from 216 stations in all regions of Brazil over a 15-year period from 2001 to 2015, which is equivalent to a total of 873,903 observations.

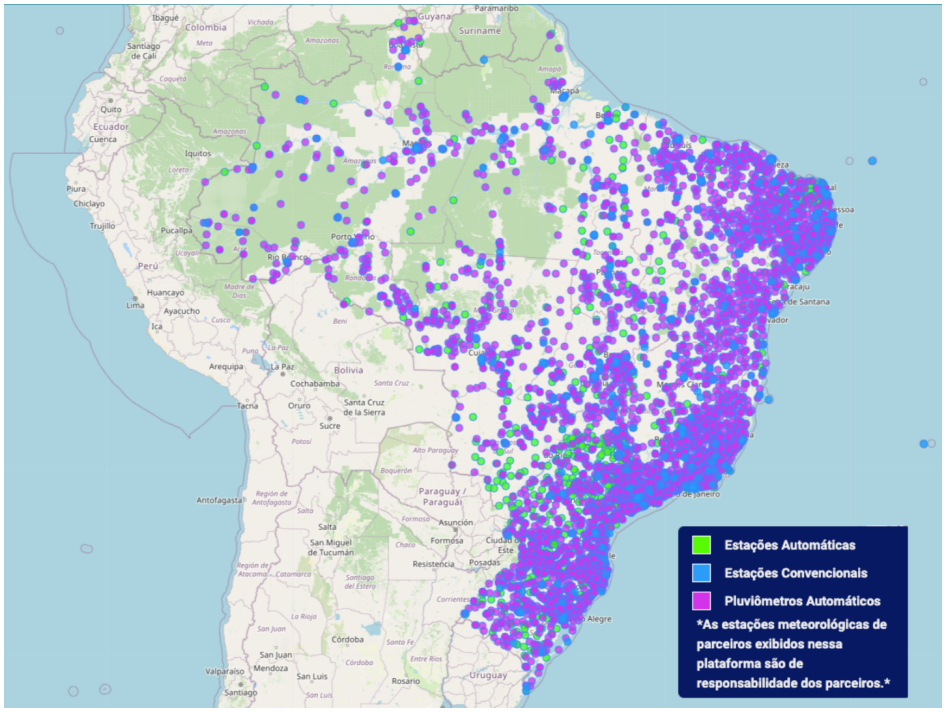


Figura 7: Meteorological stations in Brazil monitored by INMET. The map comprehends conventional stations, automatic stations and rainfall stations. Source: INMET.

During data pre-processing, we used the Python and Pandas library to handle data composed of maximum and minimum air temperature (°C), mean relative humidity (%), duration of sunshine (hours) and mean wind speed at 10 m ($ms^{-1}$). We converted the wind speed to 2 m height and used the duration of sunshine to calculate solar radiation following (ALLEN *et al.*, 1998).

Following the (FERREIRA *et al.*, 2019) methodology, we perform preprocessing, excluding days with missing data or with inconsistent values. Data were deemed inconsistent if the minimum temperature exceeded the maximum temperature, the duration of sunshine was negative or exceeded the photoperiod, the relative humidity was negative or above 100%, or the wind speed at a height of 10 meters had a negative value or exceeded 15 $ms^{-1}$.

To compute the reference values of $ET_o$, we used the FAO-56 6 Penman-Monteith equation (ALLEN *et al.*, 1998) described in equation 1. This approach has been widely used to calculate $ET_o$, due to its simplicity and acceptable accuracy. To help with calculations, we used a Python package to calculate the daily $ET_o$ values [2].

In the training process, we used data from 2001 to 2010 (10 years) from all meteo-

[2]https://github.com/Evapotranspiration/ETo

rological stations studied, adding up to 606,639 observations after data preprocessing. After being randomized, we divided the data into a training set (75%) and a validation set (25%). We used the data from the training set in the machine learning model, and we used the data from the validation set for hyperparameter optimization. We evaluated the performance of the ANN model with data from the test dataset, 267,263 observations from 2011 to 2015 (5 years).

We standardized the data using the StandardScaler from the Sci-kit Learn library, which removes the mean and scales to unit variance. Standardization of the data is a common requirement for many machine learning estimators prior to training the machine learning models, as they might behave badly and cause convergence problems. It makes different variables more comparable. Equation 6 depicts the normalization calculus. We used the data of the training set to reflect the real use of the models in which the mean ($\mu$) and the standard deviation ($\sigma$) were calculated and where $x_{ni}$ is the standardized value, $x_i$ is the observed value.

$$x_{ni} = \frac{x_i - \mu}{\sigma} \tag{6}$$

## 4.2 Models creation

We used the Artificial Neural Network (ANN), which is widely used for data regression and also yielded good results with limited meteorological data according to research by (FERREIRA; CUNHA, 2020; FAROOQUE *et al.*, 2022). We used the same ANN architecture presented in (FERREIRA *et al.*, 2019), where the ANN used consisted of an input layer, two hidden layers, and an output layer. The size of the input data varied according to the number of inputs.

Three ANN models were considered to evaluate the performance of the prediction according to the number of inputs. The first model (T) is a model that receives two inputs: minimum and maximum temperature. The second model (H) receives three inputs: minimum temperature, maximum temperature, and realtive humidity. The third model (W) receives four inputs: minimum temperature, maximum temperature, relative humidity, and wind speed.

In our methodology, we chose to use only the parameters that would come from

sensors that have easier access, focusing on reducing the costs of the final hardware. These parameters were specially chosen to simulate the input of sensor values in real cases to simulate an environment in which we have limited meteorological data availability from sensors. Other works used a similar approach and resorted to reducing the number of inputs. For example, in (SIDHU *et al.*, 2020) the authors performed correlation thresholds for feature selection, which helps in reducing the number of input parameters from the initial 26 to the final 11. In (FAROOQUE *et al.*, 2022) the authors used the relative importance method to determine the input variables best suited to the models, where six different variables including solar radiation, maximum temperature, mean temperature, minimum temperature, relative humidity, and wind speed were considered as inputs for the forecasting models.

## 4.3 Models training and test

The library used to train the model was TensorFlow. The parameters used in the model can be found in Table 3. The loss function used to evaluate the model during the training process was the equation 2, where we take the observed value, subtract the predicted value, square that difference for all observations, then sum all of those squared values and divide by the number of observations. It was minimized to improve the performance and show how good the model is in terms of predicting the expected outcome.

The Optuna framework [3] was used during the training process to identify the best hyperparameters for the model. It is an open-source hyperparameter optimization framework to automate hyperparameter search. It uses the terms study and trial, in which a study is an optimization based on an objective function and the goal of a study is to find the optimal set of hyperparameter values through multiple trials. We considered 30 trials and optimized the batch size, epochs, and learning rate. We perform optimization to minimize the root mean square error (RMSE) value after evaluating the model using the validation dataset as input. The hyperparameters found by this optimization process are shown in Table 3.

[3] https://optuna.org/

## 4.4 Model evaluation

Following the training phase, the model was evaluated using the previously separated test dataset, composed of 5 years of data, from 2011 to 2015. Based on (FERREIRA *et al.*, 2019), we evaluated the models for each weather station utilizing equations 3, 5, and the equation 4 was used as a loss function for this regression problem for its very intuitive interpretation in terms of relative error.

## 4.5 Embedding the models

After the validation of the ANN model, the TensorFlow Lite library (TENSORFLOW, 2024) was used to translate the model, converting a trained TensorFlow model implemented using Python to run on microcontrollers. The model was converted to a FlatBuffer (identified by the .tflite file extension), reducing the size of the model and modifying it to use TensorFlow Lite operations. As many microcontroller platforms do not have native filesystem support, a C source file was generated, containing the TensorFlow Lite model as a char array, and then the model was included as a C array and compiled into the program.

TensorFlow Lite for Microcontrollers library is an on-device machine learning toolset that helps developers run models on mobile, embedded, and IoT devices. It is written in C++ 11 and requires a 32-bit platform. It has been tested extensively with many processors based on the Arm Cortex M Series architecture and has been ported to other architectures, including ESP32. The framework is available as an Arduino library (DAVID *et al.*, 2021).

With the model ready and validated, it was prepared and deployed on an Espressif ESP32-S3-DevKitC-1 board, which is shown in Fig. 8. This board is suitable for running various machine learning models, requires a low voltage for power, has a 32-bit MCU with 240 MHz dual core, 4 MB of programmable memory, and 532 kB of RAM and has good connectivity options integrating Wi-Fi and Bluetooth. The C source file of the model was compiled on the board with the assistance of the open source software Integrated Development Environment (IDE) (Arduino, 2024).

## 4.6 Evaluating the embedded models

After deploying the model on the board, it was evaluated considering the same test dataset used to evaluate the model on the computer. The embedded model was evaluated in two different ways: (a) with the complete test dataset, streaming the 267,263 observations via serial, to calculate the model metrics (RMSE, $R^2$ Score, MAPE), and (b) with a random sample of a thousand observations, converted into floating-type data arrays, one for each input parameter, and integrated into the code compiled on the board to calculate the occupied memory space by the model and the energy consumption.

Given the restricted memory capacity of the selected microcontroller, embedding the entire test dataset for model evaluation was not feasible. To preserve the integrity of the tests using the identical evaluated dataset, we opted for (a) methodology. This involved establishing communication between the computer and the microcontroller to transfer data via the serial port. In both cases, each inference made by the model used one item from each array, continuing this process until the entire database was processed. This study used only simulated input parameters, there was no data from real sensors.

The time and energy consumption required for the model to run a thousand inferences from the test set were measured, as well as the memory space occupied by the model on the board. The information on the memory space occupied by the model was obtained using the Arduino IDE tool, which reports these data on its console after compiling the code. The execution time was measured with the help of the Arduino's $millis()$ function, which starts a millisecond timer during the code execution on the board. The timer was started immediately before making the first inference with the model and was stopped immediately after the last inference was executed. The outcome of the timer was displayed on the Arduino IDE serial output.

Following (PEREIRA *et al.*, 2024), the measurement of the energy consumption on the ESP-32 board was performed using a Power Profiler Kit II, manufactured by Nordic Semiconductor (Normic Semicondutor SA, 2024). The board allows accurate power consumption measurements for the entire range typically seen in low-power embedded applications Figure 9.

The software provides information on the amount of charge and the average current over a given time window in seconds. Five measurements were taken using the software

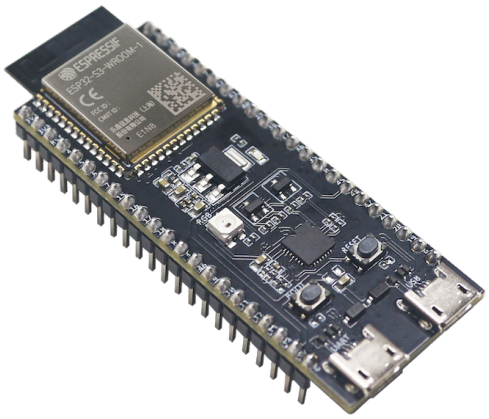

Figura 8: Description of the components of the ESP32 board manufactured by Espressif (Source: (Espressif Systems (Shanghai) Co., 2024)).

and the kit, making all inferences from the test dataset for each measurement. The first measurement was discarded because of interference spikes from the process of powering up the board. From the next 5 measurements, an average of the amount of charge used in each execution window of the model was calculated. By multiplying the amount of charge by the applied voltage on the board, which is 3.3 V, we obtain the energy in Joules (PEREIRA *et al.*, 2024). In Figure 10, it is possible to observe that all tests started with a five-second delay to avoid unstable consumption when powering the board.

The regression model was also deployed in the cloud using the Google Cloud platform [4]. The aim was to compare the model's performance when it is running in the cloud and accessed by the ESP-32 via API with the model's performance when it is running embedded in the ESP-32. The inferences were executed iteratively, employing the same strategy and utilizing the identical test database used for the embedded model. During the test, a delay of 1 second was imposed between each inference to prevent overload of the server. Figure 11 presents a diagram that illustrates the architecture used for both the embedded model and the cloud model.

[4]https://cloud.google.com/

Figura 9: Tools used to measure the power consumption of the ESP-32 board. Source: Author.

# 5 Results and discussion

This section presents the results of the research, beginning with a comparison of the performance of different models in terms of accuracy and efficiency. The focus then shifts to the performance of the models when embedded in hardware devices, evaluating their effectiveness in practical applications. A detailed discussion follows, where the results are analyzed, and insights are provided into the strengths and limitations of the proposed models. This section also explores potential improvements and implications for future work.

## 5.1 Model Comparison

Table 1 shows the mean values of RMSE and $R^2$ reported in (FERREIRA *et al.*, 2019) and calculated in the model evaluation phase for each model considered in this work (Models T, H and W). The table also shows the MAPE metric considered in the current study. Models T, H, and W showed a lower mean RMSE compared to (FERREIRA *et al.*, 2019). However, only Model W presented a higher mean value of $R^2$ score compared to (FERREIRA *et al.*, 2019).

Table 1 also shows, in the last three columns, the results of the adapted model running

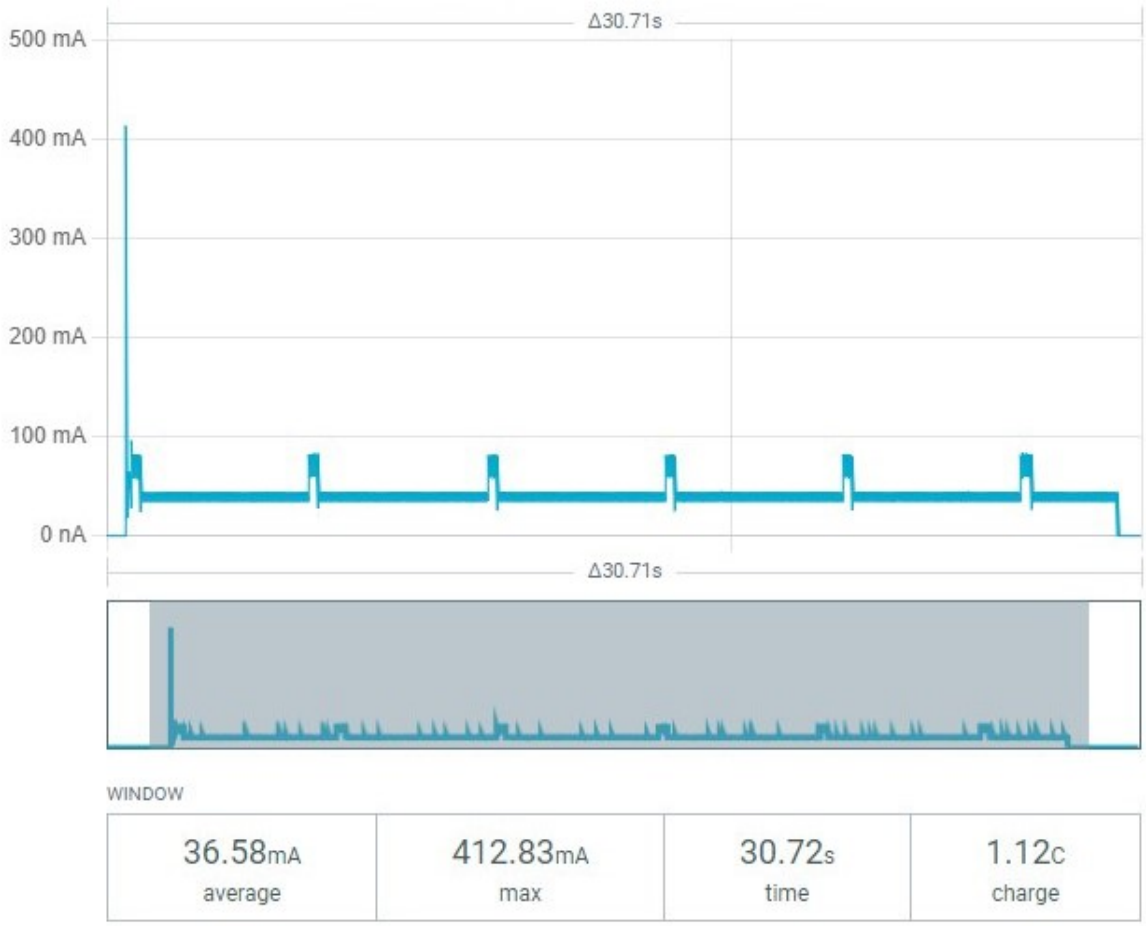


Figura 10: Print screen of the Nordic Power Profile Software used to measure the power consumption of the ESP-32 board for the model with only temperature as input. Source: Author.

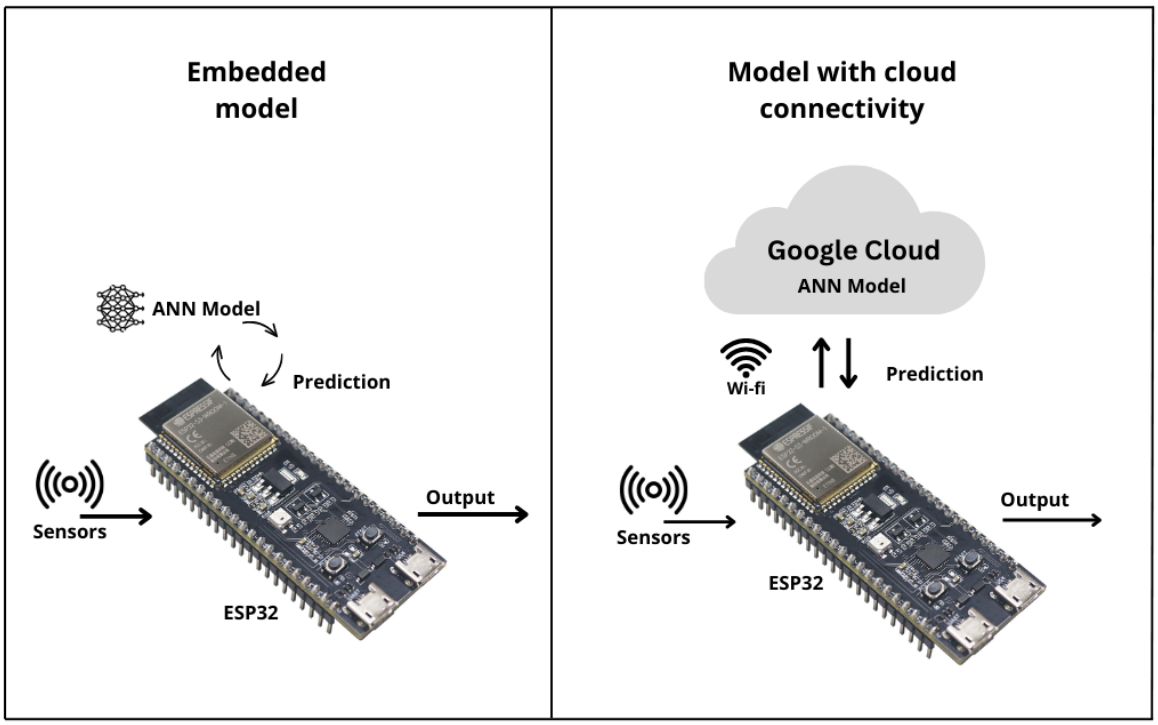


Figura 11: Architectures utilized to assess the performance of the embedded models. On the left is the architecture of the embedded models, while on the right is the architecture of the Cloud Model. Source: Author.

on the ESP-32 board, using the full test dataset and streaming the observations through the serial port (Embedded Models T, H and W). One can see that the embedded model that considers only the minimum and maximum temperature parameters, the Embedded Model T, showed a higher RMSE, 19% higher, compared to Model T (not embedded), while it presented the same mean values for the MAPE and $R^2$ metrics. The embedded model that added relative humidity values, the Embedded Model H, showed an increase of 0.16 $mmd^{-1}$ in the mean RMSE value and returned the same values for $R^2$ and MAPE compared to Model H. The model with four parameters that added wind speed values as input, the Embedded Model W, showed an increase in RMSE of 0.13 $mmd^{-1}$ compared to Model W, while the MAPE of Model W and Embedded Model W are equal.

Figure 12 shows the violin plot of the RMSE values. One can see that the error distribution of the embedded models closely matches that of the model running on the computer for the three models analyzed.

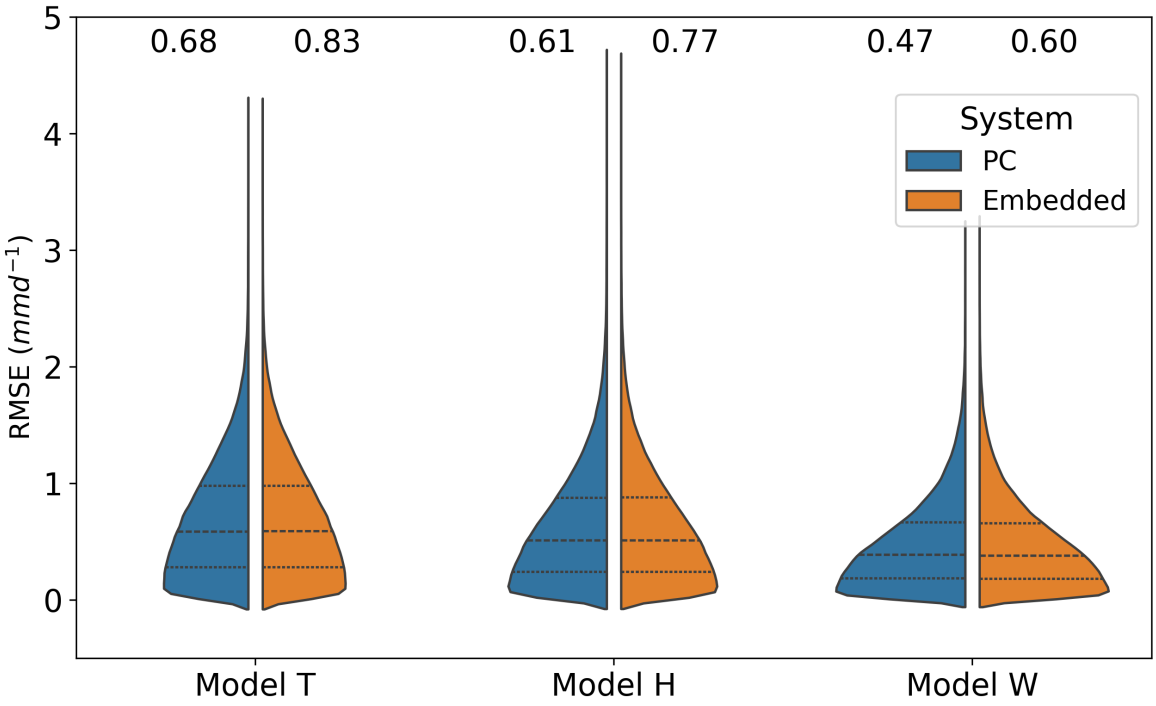


Figura 12: Violin plot for the mean values comparing the performance of the models running on a PC and the embedded models. Source: Author.

## 5.2 Performance of embedded models

Table 2 shows the memory consumption, time to run the inferences, and energy consumption measured in the performance evaluation of the embedded models. The Embedded Model T used 13,104 bytes of memory, while the Embedded Model H utilized 264 bytes more, and the Embedded Model W required an additional 464 bytes. One can see that in terms of memory space, the addition of new features as input did not significantly increase the size of the model.

For the time required to execute 1,000 inferences on the test dataset, the Embedded Model T required 267.5 ms, while the Embedded Model H took an additional 5.3 ms, and the Embedded Model W needed 10 ms more in comparison to the Embedded Model T. The embedded models presented a relatively close execution time, with each model not exceeding 280 milliseconds for all thousand inferences.

In terms of energy consumption, the embedded models also presented very similar values around 0.06 J. To calculate the energy consumption, we considered the charge value of 1,000 inferences returned by the Power profile software. We ran the tests without any

battery optimization method.

The model that was deployed in the cloud and used in the comparison was Model H, since it obtained a better result than Model T and uses data from sensors that are easy to access (air temperature and relative humidity) than Model W (wind speed). The Cloud Model had a total execution time of 30 minutes and 6 seconds and occupied 841,165 bytes of the ESP32 storage. Since we added a one second delay to not overload the server, after removing the total of one thousand seconds of delays, one can see that the Cloud Model spent 806 seconds to execute 1,000 inferences.

The Cloud Model had a consumption mean value of 264 J for the thousand inferences, three orders of magnitude higher than the embedded models. Empirical analysis of charge values showed that the measured charge per inference using the Cloud Model varies between 80 mC and 87 mC. To simplify the calculus of the energy in Joules, we considered that each inference of the Cloud consumes 80 mC. Therefore, our calculations are underestimating the Cloud Model energy consumption and the difference between the Cloud Model and the embedded models can be higher.

## 5.3 Discussion

The model reproduced from (FERREIRA *et al.*, 2019) and with optimized hyperparameters, Model T, showed better performance than the one reported in (FERREIRA *et al.*, 2019) considering RMSE. The model using three parameters as input, Model H, showed a slight improvement compared to Model T, with a lower RMSE and higher $R^2$. The model using four parameters as input, adding the wind speed, Model W, had the best performance, with the lowest mean RMSE and MAPE, and a higher mean $R^2$.

The embedded model that considers only the minimum and maximum temperature parameters (Embedded Model T) showed a higher RMSE compared to the one running on a computer (Model T), while it presented the same values for the MAPE and $R^2$ metrics. The embedded model that considers relative humidity values (Embedded Model H) also suffered a worsening when comparing the RMSE values of Model H. The model with four parameters (Embedded Model W) had the best performance among the embedded scenarios, although it presented a higher mean RMSE value compared to Model W.

The decrease in RMSE in the embedded models may be related to differences in

floating-point operations on the board or in the process of converting the adapted model from Python to C using the (DAVID *et al.*, 2021) library.

Both Model H and Embedded Model H had a more balanced performance among the other proposed models. Additionally, given that numerous sensors can measure both air temperature and relative humidity, it is more cost-effective and simpler to employ hardware that detects these two variables rather than hardware that also measures wind speed. Therefore, we believe that models that only use air temperature and relative humidity are the best choice.

Making local inferences using the Embedded model H consumed 37.97% less space than using the Cloud Model considering the entire program storage and the included embedded test dataset. We can also observe that the embedded model was more energy efficient, using approximately 99.98% less energy to process all inferences compared to the cloud model. Moreover, the comparison showed that embedding the model makes inferences 99.97% faster.

The findings revealed a pattern similar to that observed in (PEREIRA *et al.*, 2024), which served as a foundation for our work. Evaluating the balance between expense and limited data, the proposed embedded solution demonstrated acceptable performance, closely aligning with findings in the literature, particularly our primary reference (FERREIRA; CUNHA, 2020), as well as (FAROOQUE *et al.*, 2022), which noted an RMSE of 0.62 mm/day for one of the weather stations analyzed.

# 6 Conclusion

In this paper, we present an energy-efficient TinyML model for prediction $ET_o$ with minimal meteorological data, using simulated parameters as input for trained models and comparing its performance with a cloud-based scenario. The performance of the obtained embedded model, using the ANN algorithm with mean minimum and maximum air temperature values as input, was close to the reference in the literature considering metrics such as RMSE (0.83 $mmd^{-1}$), $R^2$ (0.56) and MAPE (17%). We also evaluate two new models by adding relative humidity and wind speed as input. The new models present an improvement in accuracy when comparing the model with fewer values, and the new model with more parameters presented 5% less for MAPE. The presented model outperformed the compared cloud-based model in metrics such as memory footprint, 37.97% less memory used, inference time, 99.97% faster, and energy consumption with 99.98% less energy consumed.

The findings from this study indicate that embedding the prediction of $ET_0$ using minimal meteorological data as input into a cost-effective device is feasible without substantially compromising performance and accuracy. Running a machine learning model directly on the embedded device streamlines its hardware implementation. It reduces the overall solution cost since it eliminates the need for a communication framework with an external source. Additionally, an optimized solution facilitates better energy consumption by enabling battery-operated systems suitable for deployment in more remote areas. The study further demonstrated that incrementally increasing the number of inputs had a negligible impact on energy use and memory footprint while enhancing the model's accuracy.

This research has certain limitations. Initially, we developed a model using a comprehensive dataset that spans the entirety of Brazil. While this extensive dataset might enhance the model's overall generalization capabilities, Brazil is a vast nation with varied climatic conditions across its regions. Consequently, models that are tailored with data from specific local areas could potentially provide more accurate estimates when utilized in those particular geographical regions. Moreover, the current study was limited to simulated data as input and a simpler implementation, and we focused on the accuracy of the machine learning model, leaving the optimization of the embedded model for future work.

A prospective opportunity involves carrying out a comparable analysis using other

machine learning models like Random Forest and Support Vector Machines, while taking into account various hardware platforms. Additionally, employing strategies like clustering techniques to categorize weather stations based on shared attributes like geographical position and climatological trends, as demonstrated by the authors in (FERREIRA *et al.*, 2019), could be considered to evaluate model performance improvements. This would enable an exploration of the differences, as well as the potential strengths and weaknesses, in constructing a TinyML.

We can also employ the specified methodology with real sensor data to evaluate its influence on model performance in actual irrigation scheduling and create a deep understanding of water cycle control within agricultural systems. This would include examining new parameters and contrasting performance with existing models already documented in the literature, thereby validating the model and its results to ensure its reliability and practical applicability in real-world contexts.

These findings mark an important initial move toward diversifying parameter prediction implementations in agricultural systems. Using new models and optimized methods, particularly those centered on TinyML and machine learning techniques, the research aims to improve computational and system performance.

# References

ABADADE, Y. *et al.* A comprehensive survey on tinyml. *IEEE Access*, IEEE, 2023.

ALLEN, R. G. *et al.* Evapotranspiration information reporting: I. factors governing measurement accuracy. *Agricultural Water Management*, Elsevier, v. 98, n. 6, p. 899–920, 2011.

ALLEN, R. G. *et al.* Crop evapotranspiration-guidelines for computing crop water requirements-fao irrigation and drainage paper 56. *Fao, Rome*, v. 300, n. 9, p. D05109, 1998.

ANA. *Atlas Irrigação*. 2024. Acessed on September 06, 2024. Available at: https://portal1.snirh.gov.br/ana/apps/storymaps/stories/a874e62f27544c6a986da1702a911c6b.

Arduino. *Arduino IDE Software Documentation*. 2024. Accessed in February 10, 2024. Available at: https://docs.arduino.cc/software/ide/.

BU, F.; WANG, X. A smart agriculture iot system based on deep reinforcement learning. *Future Generation Computer Systems*, Elsevier, v. 99, p. 500–507, 2019.

CHEN, M. *et al.* A reinforcement learning approach to irrigation decision-making for rice using weather forecasts. *Agricultural Water Management*, Elsevier, v. 250, p. 106838, 2021.

CHEN, Z. *et al.* Estimating daily reference evapotranspiration based on limited meteorological data using deep learning and classical machine learning methods. *Journal of Hydrology*, Elsevier, v. 591, p. 125286, 2020.

COSTA, J. M. *et al.* Water and heat fluxes in mediterranean vineyards: indicators and relevance for management. *In*: *Water Scarcity and Sustainable Agriculture in Semiarid Environment*. [*S.l.: s.n.*]: Elsevier, 2018. p. 219–245.

DAVID, R. *et al.* Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems*, v. 3, p. 800–811, 2021.

DIAS, S. H. B. *et al.* Reference evapotranspiration of brazil modeled with machine learning techniques and remote sensing. *Plos one*, Public Library of Science San Francisco, CA USA, v. 16, n. 2, p. e0245834, 2021.

DUTTA, L.; BHARALI, S. Tinyml meets iot: A comprehensive survey. *Internet of Things*, Elsevier, v. 16, p. 100461, 2021.

Espressif Systems (Shanghai) Co. *ESP32-S3-DevKitC-1 v1.1 - ESP32-S3 - — ESP-IDF Programming Guide latest documentation*. 2024. Available at: https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/hw-reference/esp32s3/user-guide-devkitc-1.html.

FAROOQUE, A. A. *et al.* Forecasting daily evapotranspiration using artificial neural networks for sustainable irrigation scheduling. *Irrigation Science*, Springer, p. 1–15, 2022.

FERREIRA, L. B.; CUNHA, F. F. da. New approach to estimate daily reference evapotranspiration based on hourly temperature and relative humidity using machine learning and deep learning. *Agricultural Water Management*, Elsevier, v. 234, p. 106113, 2020.

FERREIRA, L. B. *et al.* Estimation of reference evapotranspiration in brazil with limited meteorological data using ann and svm–a new approach. *Journal of Hydrology*, Elsevier, v. 572, p. 556–570, 2019.

GOAP, A. *et al.* An iot based smart irrigation management system using machine learning and open source technologies. *Computers and electronics in agriculture*, Elsevier, v. 155, p. 41–49, 2018.

HONG, C. K. *et al.* Analysis of wind speed prediction using artificial neural network and multiple linear regression model using tinyml on esp32. *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences*, v. 107, n. 1, p. 29–44, 2023.

JIA, W. *et al.* Daily reference evapotranspiration prediction for irrigation scheduling decisions based on the hybrid pso-lstm model. *Plos one*, Public Library of Science San Francisco, CA USA, v. 18, n. 4, p. e0281478, 2023.

LORITE, I. J. *et al.* Using weather forecast data for irrigation scheduling under semi-arid conditions. *Irrigation science*, Springer, v. 33, p. 411–427, 2015.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of machine learning*. [*S.l.: s.n.*]: MIT press, 2018.

MONTELEONE, S.; MORAES, E. A. D.; MAIA, R. F. Analysis of the variables that affect the intention to adopt precision agriculture for smart water management in agriculture 4.0 context. *2019 Global IoT Summit (GIoTS)*, IEEE, p. 1–6, 2019.

MORAIS, J. d. *et al.* Avaliação do método de penman monteith fao 56 com dados faltosos e de métodos alternativos na estimativa da evapotranspiração de referência no submédio vale do são francisco. *Revista Brasileira de Geografia Física*, v. 8, n. 6, p. 1644–1660, 2015.

NAGAPPAN, M.; GOPALAKRISHNAN, V.; ALAGAPPAN, M. Prediction of reference evapotranspiration for irrigation scheduling using machine learning. *Hydrological Sciences Journal*, Taylor & Francis, v. 65, n. 16, p. 2669–2677, 2020.

NAWANDAR, N. K.; SATPUTE, V. R. Iot based low cost and intelligent module for smart irrigation system. *Computers and electronics in agriculture*, Elsevier, v. 162, p. 979–990, 2019.

Normic Semicondutor SA. *Power Profiler Kit II*. 2024. Available at: https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2.

PEREIRA, E. A. M.; SANTOS, J. F. da S.; BARBOZA, E. de A. An energy efficient tinyml model for a water potability classification problem. *Sustainable Computing: Informatics and Systems*, Elsevier, p. 101010, 2024.

PLEVRIS, V. *et al.* Investigation of performance metrics in regression analysis and machine learning-based prediction models. *In*: EUROPEAN COMMUNITY ON COMPUTATIONAL METHODS IN APPLIED SCIENCES. *8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2022)*. [*S.l.: s.n.*], 2022.

RAY, P. P. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, Elsevier, v. 34, n. 4, p. 1595–1623, 2022.

SIDHU, R. K.; KUMAR, R.; RANA, P. S. Machine learning based crop water demand forecasting using minimum climatological data. *Multimedia Tools and Applications*, Springer, v. 79, p. 13109–13124, 2020.

TENSORFLOW. *TensorFlow Lite*. 2024. Available at: https://www.tensorflow.org/lite/ ↩ guide.

Tabela 1: Mean values of Root Mean Square Error (RMSE), $R^2$ Score and Mean Absolute Percentage Error (MAPE) for the proposed ANN models and a reference ANN model proposed in (FERREIRA *et al.*, 2019).

| **Metric** | **(FERREIRA *et al.*, 2019)** | **Model T** | **Model H** | **Model W** | **Embedded Model T** | **Embedded Model H** | **Embedded Model W** |
|---|---|---|---|---|---|---|---|
| RMSE | 0.81 | 0.67 | 0.61 | 0.46 | 0.83 | 0.77 | 0.59 |
| $R^2 Score$ | 0.67 | 0.56 | 0.62 | 0.77 | 0.56 | 0.62 | 0.77 |
| MAPE (%) | - | 17 | 15 | 12 | 17 | 15 | 12 |

Tabela 2: Values of the memory occupied only by the model, memory occupied by the complete embedded program in the ESP32 storage, the time to run 1,000 inferences and the energy consumption to run 1,000 inference for the embedded models and the “Cloud Model".

| **Metric** | **Embedded Model T** | **Embedded Model H** | **Embedded Model W** | **Cloud Model H** |
|---|---|---|---|---|
| Memory occupied by the **model** (Bytes) | 13,104 | 13,368 | 13,568 | - |
| Memory occupied in the **program storage** (Bytes) | 521,813 | 522,069 | 522,277 | 841,165 |
| Time to run 1,000 inferences (ms) | 267.5 | 272.8 | 277.6 | 806,000 |
| Energy consumption to run 1,000 inferences(J) | 0.06303 | 0.0649308 | 0.0658416 | 264 |

Tabela 3: Parameters used for training the ANN model using TensorFlow library and Optuna Optimization Framework.

| **Parameter** | **Model (T)** | **Model (H)** | **Model (W)** |
|---|---|---|---|
| input_layer | 2 | 3 | 4 |
| hidden_layer | 2 | 2 | 2 |
| neurons | 50 | 50 | 50 |
| output_layer | 1 | 1 | 1 |
| loss | MSE | MSE | MSE |
| activation_func | tanh | tanh | tanh |
| epoch | 437 | 361 | 443 |
| batch_size | 32 | 32 | 16 |
| learning_rate | 3.92e-5 | 1.41e-5 | 3.77e-5 |