UNIVERSITÀ DI PARMA
DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

Dottorato di Ricerca in Tecnologie dell'Informazione
XXXVII Ciclo

Armin Mazinani

Artificial Intelligence of Things: Implementation, Performance Analysis, Computational Feasibility

DISSERTAZIONE PRESENTATA PER IL CONSEGUIMENTO
DEL TITOLO DI DOTTORE DI RICERCA

DECEMBER 2024

UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXXVII Ciclo

Artificial Intelligence of Things: Implementation, Performance Analysis, Computational Feasibility

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Gianluigi Ferrari

Dottorando: *Armin Mazinani*

December 2024

To My Family

Contents

List of Figures

List of Tables

List of Acronyms

1D One-dimensional

AdaBoost Adaptive Boosting

Adam Adaptive Moment Estimation

AE AutoEncoder

ALSTM Aggregated LSTM

AI Artificial Intelligence

ANN Artificial Neural Network

AQI Air Quality Index

AR Auto Regressive

ARIMA Automatic Regression Integrated Moving Average

ARMA Auto-Regressive Moving Average

AUC Area Under the ROC Curve

BiGRU Bidirectional GRU

BiLSTM Bidirectional LSTM

BiRNN Bidirectional RNN

BO Bayesian Optimization

BPNN Back Propagation Neural Network

BTC Bitcoin

CDAE Convolutional Denoising AutoEncoder

CPU Central Processing Unit

CNN Convolution Neural Network

DL Deep Learning

DLT Distributed Ledger Technology

DT Decision Tree

ETH Ethereum

FFN Feed Forward Network

FFT Fast Fourier Transform

FW Firmware

GBTR Gradient Boosted Tree Regression

GP Gaussian Process

GRNN Generalized Regression Neural Network

GRU Gated Recurrent Unit

HAR Human Activity Recognition

k-**NN** *k*-Nearest Neighbor

IoT Internet of Things

ITS Intelligent Transportation System

LMU Legendre Memory Unit

LoRa Long Range

LSTM Long Short-Term Memory

MA Moving Average

MACC Multiply and ACCumulate

MAE Mean Absolute Percentage Error

MAPE Mean absolute percentage error

MCC Matthews Correlation Coefficient

MCU Microcontroller Unit

MFCC Mel-frequency Cepstral Coefficient

ML Machine Learning

MLP MultiLayer Perceptron

NLP Natural Language Processing

NN Neural Network

NVM Non-Volatile Memory

ODE Ordinary Differential Equation

P2P Peer-to-Peer

PCA Principal Component Analysis

PCC Pearson Correlation Coefficient

PCEN Per-Channel Energy Normalization

PD Parkinson's Disease

PLP Perceptual Linear Prediction

PM Particulate Matter

PoE Power over Ethernet

PTQ Post-Training Quantization

PUF Physical Unclonable Function

QAT Quantization Aware Training

QDL Quantized DL

QKeras Quantized Keras

R^2 Coefficient of determination

RAM Random Access Memory

RASTA-PLP Reflexive Structural PLP

RBF Radial Basis Function

ReLU Rectified Linear Unit

RF Random Forest

RFID Radio Frequency Identification

RMSE Root Mean Square Error

RMSProp Root Mean Square Propagation

RNN Recurrent Neural Network

RO Read-Only

ROC Receiver Operating Characteristic

RW Read/Write

SARIMAX Seasonal AutoRegressive Integrated Moving Average with eXogenous regressor

SELU Scaled Exponential Linear Unit

SimpleRNN Simple Recurrent Neural Network

SISDR Scale-Invariant Signal-to-Distortion Ratio

SMAPE Symmetric Mean Absolute Percentage Error

SMOTE Synthetic Minority Over-Sampling Technique

STFT Short-Time Fourier Transform

SVM Support Vector Machine

SVR Support Vector Regression

TCN Temporal Convolutional Network

TFLite TensorFlow Lite

WHO World Health Organization

XGBoost eXtreme Gradient Boosting

XRP Ripple

Chapter 1

Introduction

1.1 The role of the Internet of Things (IoT)

As widely known, with the term Internet of Things (IoT) it is common to refer to a network of *smart* devices interconnected with the Internet, able to gather, share, and communicate data with other devices through various protocols (e.g., Bluetooth, ZigBee, Z-Wave, Wi-Fi, Radio Frequency Identification (RFID), etc.). In particular, IoT devices can range from simple nodes equipped with very limited processing power and sensing elements, to advanced embedded systems with multi-functional capabilities. In fact, the rapid development of the IoT paradigm has dramatically changed the way humans and machines interact with the environment, enabling innovations in various fields, including smart healthcare, smart agriculture, smart transportation, and smart cities [2].

As an example, thanks to the adoption of healthcare IoT systems, patients' care has been improved to a large extent because of remote health monitoring, nowadays with the possibility to real-time track the patient's vital signs. New wearable devices have opened to the possibility, with the help of sensors, to detect diseases such as cardiovascular diseases, which can sometimes be very hard to detect at their early stages. In addition to this, IoT-based systems will also provide real-time monitoring of drug interactions to ensure the treatments are safe and effective [3].

Table 1.1: Comparison of tiny IoT devices and IoT devices based on key features.

Feature	Tiny IoT Devices	IoT Devices
Size	Very small	Varies (small to large)
Power Consumption	Extremely low	Can be moderate to high
Computational Power	Limited (e.g., microcontrollers)	Moderate to high (e.g., CPUs/GPUs)
Memory	Minimal (KB to MB)	Higher (MB to GB)
Connectivity	Low-power protocols (e.g., BLE)	Versatile (Wi-Fi, 5G, Ethernet)
Cost	Low	Moderate to high
Applications	Basic sensing and monitoring	Advanced analytics and control

Then, IoT solutions notably play a significant role in the efficiency and productivity of smart agriculture. Automated systems in the animal breeding management have been able to detect animals' health problems, with the feeding process being optimized and earlier detection of diseases being enabled. Furthermore, the IoT also helps the conservation of resources by reducing water consumption and the promotion of sustainable agricultural practices [4].

Finally, in the aim of smart transportation and smart cities, these scenarios will experience (without any doubt) the most significant and extensive application of IoT technologies. As an example, some of the benefits might include, among others: (i) the reduction of congestion, (ii) constant monitoring of road traffic conditions, (iii) the provision of public safety, as well as (iv) control and prediction of air pollution levels. Thus, embedding IoT technologies into the urban environment will enable new functionalities, such as optimizing the traffic flow, improving the air quality, and creating safer and more efficient urban environments in smart cities [5].

Tiny IoT devices are a subset of IoT that are designed particularly for resource-constrained environments differing in size, processing capabilities, power consumption, and targeted applications. Generally, the tiny IoT devices should be optimized for minimal power and cost. The comparison of IoT devices and tiny IoT devices is represented in Table 1.1.

1.1.1 Components of Tiny IoT Devices

Conversely to "general" IoT systems, *tiny* IoT devices present some "basic" components mainly with a particular function while strictly fulfilling the design and energy demands. Thus, the hardware part of a whole device is working effectively even in cases where available limited resources are still present. For the sake of completeness, in the following the main components of tiny IoT systems are listed.

Microcontroller Unit (MCU) An MCU consists of a Central Processing Unit (CPU), a certain amount of Random Access Memory (RAM), and customized input/output peripherals, and is mainly responsible for data processing, device control, and inter-component communication [6].

Sensors On a wide perspective, sensors act as the main data collecting units able to "sense" and acquire environmental data of high importance, such as pollution levels, temperature, motion, humidity, and health indicators. These information are then routed to be processed by the units and decisions are subsequently made [7].

Memory IoT devices use two types of memory: volatile memory (i.e., RAM) and Non-Volatile Memory (NVM). In detail, a NVM retains its information even after a device is switched *off*, which represents the reason why NVMs are well-suited for application code and Firmware (FW) storage. Contrariwise, a RAM is applied to store variables and information during the application's execution, thus allowing for fast data reading and writing for effective dynamic computing [8].

Short- and Long-Range Communication Protocols With regard to short- and long-range communication protocols possibly adopted by tiny IoT devices, the following are representative.

- Wi-Fi
  - Operates on IEEE 802.11 standards (5 GHz frequency).
  - Provides a short range (up to 60 feet indoors).

  - Suitable for home automation and energy management systems.
  - Advanced version (IEEE 802.11ah “HaLow”) has been designed for IoT with unique infrastructure requirements.

- Bluetooth
  - Suitable for low-power indoor tracking and automation.
  - Provides a limited range and security concerns.
  - Appropriate for the wearable IoT applications.

- Long Range (LoRa)
  - Provides long-range transmissions with low power consumption.
  - Capable of transmitting data over distances exceeding 10 km, making it ideal for applications such as smart agriculture and remote monitoring.

Then, depending on the specific requirements of the application running on top the considered tiny IoT device, these network protocols—which are frequently used in IoT devices—offer distinct benefits in terms of data throughput, battery consumption, and range.

Power Sources With regard to the power sources exploitable by tiny IoT devices to support a certain life span for operating on the collected information, they are generally categorized as follows [9].

- Storage
  - Includes batteries and capacitors, storing energy for later use.

- Distribution
  - Includes wired power sources (e.g., Power over Ethernet (PoE)) and electromagnetic power distribution, both representing continuous power suppliers.

- Harvesting
  - Refers to energy sources that capture and convert environmental energy into electrical power, such as solar, wind, and thermal sources.

These elements cooperate together in order to enable tiny IoT devices to effectively carry out certain tasks in specific environments, enabling a variety of applications in fields including industrial automation, healthcare, and environmental monitoring [10].

1.1.2 Artificial Intelligence (AI)-enabled tiny IoT Devices

The integration of Artificial Intelligence (AI) mechanisms in tiny IoT devices marks a milestone, enabling them to locally process data, derive intelligent decisions, and perform autonomously. More in detail, one of the most challenging tasks related to this integration regards the need to provide good performance of AI models carefully considering the constrained resources of tiny IoT devices, such as: limited processing power, memory, and execution time [11].

In order to address these limitations, an evaluation of AI model performance must encompass not only their predictive accuracy but also their computational complexity. In fact, it is crucial to strike a balance between *accuracy* and *resource efficiency* to guarantee dependable and effective performance in environments with limited resources. In order to address these issues, it is necessary to employ techniques such as model quantization, hyperparameter tuning, and the use of lightweight architectures. In fact, these strategies help in reducing the specific model's complexity without affecting its performance. As a result, they are of crucial importance for the proper deployment of AI in IoT systems with energy efficiency [12].

For the sake of clarity and completeness, in the following an overview on different Machine Learning (ML) and Deep Learning (DL) models is provided, in particular focusing on their performance and suitability for deployment on resource-constrained IoT environments.

1.2 Machine Learning (ML) Models

1.2.1 *k*-Nearest Neighbors (*k*-NN)

k-Nearest Neighbor (*k*-NN) is a supervised ML algorithm (often adopted for both classification and predictive regression-like problems) which uses "feature similarity" to predict the values of new data points and a "majority voting mechanism" for classification. In detail, *k* represents the number of nearest neighboring labels taken into account to assign a label to a new data sample, using the Euclidean distance between their feature vectors. At the end, the new data point is assigned the class label that has the highest frequency among the *k* neighboring data points. Unfortunately, due to the use of all features, the *k*-NN algorithm requires a very large amount of processing resources to calculate a distance for each dimension (in the data space).

1.2.2 Adaptive Boosting (AdaBoost)

Similarly to *k*-NN, Adaptive Boosting (AdaBoost) is a supervised ML method which exploits a set of the weak classifiers to create a strong classifier model. More in detail, the goal of AdaBoost is to increase the prediction accuracy through sequential training on diverse subsets of data, then "penalizing" (assigning higher weights to) the misclassified samples.

1.2.3 Decision Tree (DT)

This ML mechanism adopts a set of *if-then-else* rules to predict future values, then creating a hierarchical tree structure with three types of nodes: (i) the *root node*, corresponding to the best predictor; (ii) *interior nodes*, representing the features; and (iii) *leaf nodes*, representing the outcome. The branches of the tree represent *if-then* conditions. Unfortunately, Decision Tree (DT) has some shortcomings, such as its high computational complexity and low flexibility, as well as the fact that small data changes can imply big differences in tree structure and cause instability.

1.2.4 Random Forest (RF)

Exploiting the concepts recalled in Subsection 1.2.3, the Random Forest (RF) algorithm creates multiple DT-based classifiers by training them on various subsets of the candidate dataset. In detail, RF aggregates the predictions of these DTs in order to determine the final prediction, typically through a voting mechanism. Then, by averaging the predictions, RF improves the prediction accuracy while also addressing overfitting concerns.

1.3 Deep Learning (DL)

In the following, a detailed analysis and description of widely used DL models for sequential data processing, with a focus on their applications in prediction and classification tasks, is provided.

1.3.1 Simple Recurrent Neural Network (SimpleRNN)

A Simple Recurrent Neural Network (SimpleRNN) is a Neural Network (NN) architecture specifically designed to analyze and capture dependencies within sequential data. In order to understand the relationship between previous and current data, SimpleRNN performs recursively, operating one operation (per series' element) to calculate (per neuron) the current (at time t) output (y_t) and the current hidden state value (h_t), depending on the current input features (x_t) and the previous (at time $t-1$) hidden state value (h_{t-1}). These quantities, which can generally be vectors, enable the network to retain and exploit past information through the hidden state, effectively serving as a memory for future predictions. In detail, as shown in Figure 1.1, a SimpleRNN performs the following operations:

$$h_t = \tanh(\mathbb{W}_h h_{t-1} + \mathbb{W}_x x_t) \tag{1.1}$$

$$y_t = \mathbb{W}_y h_t \tag{1.2}$$

where: $\mathbb{W}_h$, $\mathbb{W}_y$, and $\mathbb{W}_x$ are weight matrices for hidden, output, and input states, respectively; and $\tanh(x) \triangleq (e^{2x}-1)/(e^{2x}+1)$ is used for data normalization (element-

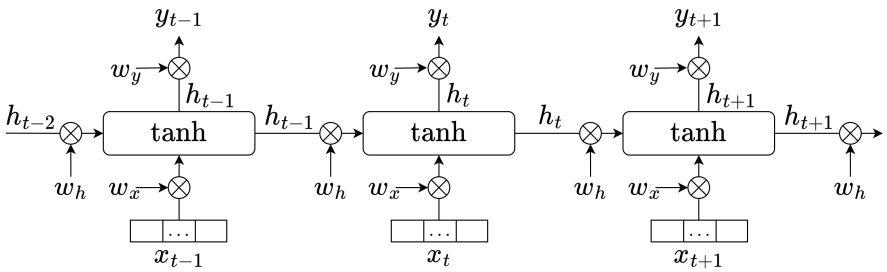

Figure 1.1: Simplified mathematical model of an SimpleRNN.

wise for the vector), returning a predicted vector h_t with elements in the (real) interval $[-1, 1]$.

SimpleRNNs are not efficient in learning long-term correlations because of vanishing or exploding gradient problems in long sequences [13]—these problems correspond to drastic decrease or increase observed at each layer during back-propagation. In order to overcome these SimpleRNNs' drawbacks and limitations, Long Short-Term Memory (LSTM) [14] and Gated Recurrent Unit (GRU) [15] have been proposed.

1.3.2 Long Short-Term Memory (LSTM) Networks

An LSTM network corresponds to an advanced version of a Recurrent Neural Network (RNN) able to detect long-term dependencies using cell state, in addition to the hidden state used in RNNs. As detailed in Subsection 1.3.1, the short-term memory and the long-term memory are represented by the *hidden state* and the *cell state*, respectively, with each LSTM cell consisting of three types of gates determining the importance of the information to be remembered or forgotten: (i) the *forget gate*, determining whether information from previous time steps is retained or forgotten; (ii) the *input gate*, specifying which new information has to be stored in the cell state; and (iii) the *output gate*, determining the final output of the LSTM cell at each time

step t. In detail, an LSTM network performs the following operations:

$$f_t = \sigma_g(\mathbb{W}_f x_t + \mathbb{U}_f h_{t-1} + b_f) \tag{1.3}$$

$$i_t = \sigma_g(\mathbb{W}_i x_t + \mathbb{U}_i h_{t-1} + b_i) \tag{1.4}$$

$$o_t = \sigma_g(\mathbb{W}_o x_t + \mathbb{U}_o h_{t-1} + b_o) \tag{1.5}$$

$$\tilde{c}_t = \sigma_c(\mathbb{W}_c x_t + \mathbb{U}_c h_{t-1} + b_c) \tag{1.6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{1.7}$$

$$h_t = o_t \odot \sigma_h(c_t) \tag{1.8}$$

where: $x_t \in \mathbb{R}^d$ is the input vector of the LSTM cell; $f_t \in (0,1)^h$ is the forget gate's activation vector; $i_t \in (0,1)^h$ is the input/update gate's activation vector; $o_t \in (0,1)^h$ is the output gate's activation vector; $h_t \in (-1,1)^h$ is the hidden state vector; $\tilde{c}_t \in (-1,1)^h$ is the cell input activation vector; $c_t \in \mathbb{R}^h$ is the cell state vector; $\mathbb{W} \in \mathbb{R}^{h \times d}$, $\mathbb{U} \in \mathbb{R}^{h \times h}$, and $b \in \mathbb{R}^h$ are weight matrices and bias vector; d and h refer to the number of input features and number of hidden units, respectively; the initial values are c_0 = 0 and h_0 = 0; and the operator $\odot$ denotes the element-wise product.

1.3.3 Gated Recurrent Unit (GRU) Networks

A GRU network can be seen as an improved version of an LSTM network, but with a simpler structure, since (unlike LSTM) the memory cell is not used and the forget gate is removed to reduce complexity. Therefore, a GRU cell includes the following two gates: (i) the *update gate*, determining how much of the past information (from previous time steps) needs to be passed to the next GRU cell; (ii) the *reset gate*, deciding how much of the past information (from the previous steps) will be disregarded. In detail, a GRU network performs the following operations:

$$z_t = \sigma(\mathbb{W}_z x_t + \mathbb{U}_z h_{t-1} + b_z) \tag{1.9}$$

$$r_t = \sigma(\mathbb{W}_r x_t + \mathbb{U}_r h_{t-1} + b_r) \tag{1.10}$$

$$\hat{h}_t = \phi(\mathbb{W}_h x_t + \mathbb{U}_h(r_t \odot h_{t-1}) + b_h) \tag{1.11}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \tag{1.12}$$

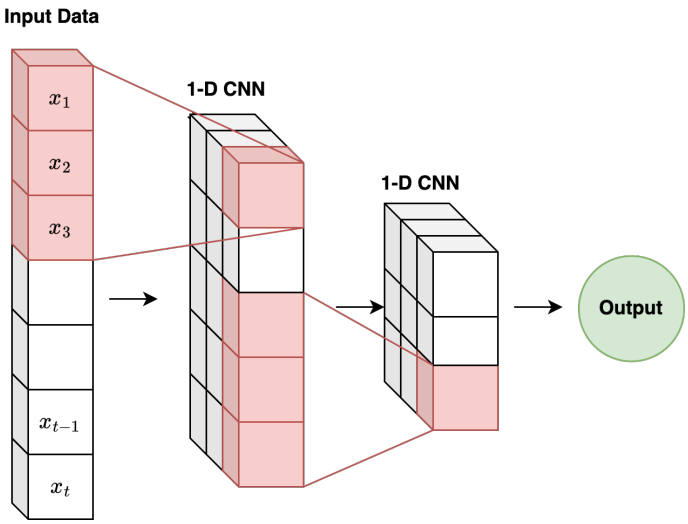

Figure 1.2: 1D-CNN architecture featuring two convolutional layers.

where: x_t and h_t are input and output vectors, respectively; $\hat{h}_t$ is the candidate activation vector; z_t is the update gate vector; r_t is the reset gate vector; $\mathbb{W}$, $\mathbb{U}$, and b are weight matrices and bias vector.

1.3.4 Convolutional Neural Networks (CNNs)

Convolution Neural Networks (CNNs) are known for their extensive image processing capabilities, thanks to their well-known automatic feature extraction capability from high-dimensional data (e.g., images and videos). More in detail, a CNN consists of two main layers, denoted as *convolution* and *pooling*. Convolution is applied to the input data using a kernel to produce a feature map, with the kernel size usually smaller than the input data. Then, after a convolution operation, pooling is performed to reduce the dimension. In general, One-dimensional (1D)-CNNs are mostly used for time series data, as shown in Figure 1.2.

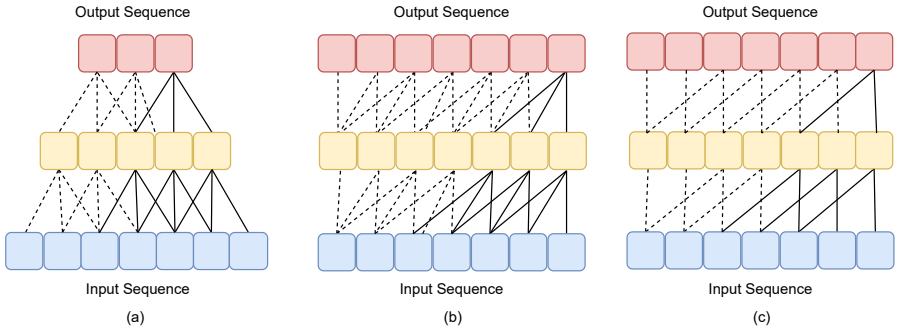

Figure 1.3: (a) Typical convolutional network with kernel size equal to 3; (b) causal convolutional network with kernel size equal to 3; (c) dilated causal convolutional network with kernel size equal to 3 and dilation rate equal to 2.

1.3.5 Temporal Convolutional Networks (TCNs)

Temporal Convolutional Network (TCN) is a particular type of 1D-CNN effective in analyzing sequential data [16], which incorporates three components: (i) *causal convolution*; (ii) *dilated convolution*; (iii) *residual connections*. In detail, a TCN allows (using causal convolutions) to prevent data from leaking from the future to the past, allowing the architecture to generate an output sequence with length similar to any input sequence using zero padding [17]. Then, in Figure 1.3 the difference between typical convolution, causal convolution, and dilated convolution, is shown.

Causal Convolution

Causal convolutions limit predictions to past observed data, with the convolution operation at time epoch t only applying to data preceding time epoch t (namely: x_0, $x_1, \ldots, x_{t-1}, x_t$). This ensures that the convolution outcome at time t is unaffected by future information.

Dilated Convolution

The dilated convolution allows TCN to expand its receptive field exponentially without increasing the model complexity [18].

Residual Connection

It is well-known that the size of the convolution kernel, dilation factor, and network depth determine the receptive field of a TCN. Moreover, the accuracy of the training set will be saturated or worsened as the network deepens and the model's parameters increase, resulting in gradient vanishing or exploding. Therefore, in order to address these issues, the TCN incorporates residual connections in its output layer, which help the propagation of data across the layers [19].

1.3.6 Legendre Memory Unit (LMU)

Legendre Memory Unit (LMU) is a recently introduced RNN using less computational resources to preserve long-range time dependencies. In detail, LMUs break down the time history into d Ordinary Differential Equations (ODEs) exploiting Legendre polynomials and Fourier attributes to orthogonalize the continuous-time history of its encoded input signal (denoted as $u_t \in \mathbb{R}$) [20]. For the sake of completeness, a LMU (whose block architecture is shown in Figure 1.4) can be defined as follows:

$$u_t = e_x^T x_t + e_h^T h_{t-1} + e_m^T m_{t-1} \tag{1.13}$$

$$m_t = \bar{A} m_{t-1} + \bar{B} u_t \tag{1.14}$$

$$h_t = f(\mathbb{W}_x x_t + \mathbb{W}_m m_t + \mathbb{W}_h h_{t-1}) \tag{1.15}$$

where: $f(\cdot)$ is a non-linear function (such as tanh); $\mathbb{W}_x \in \mathbb{R}^{n_h \times n_x}$, $\mathbb{W}_h \in \mathbb{R}^{n_h \times n_h}$, $\mathbb{W}_m \in \mathbb{R}^{n_h \times d}$ represent the weight matrices; $e_x \in \mathbb{R}^{n_x}$, $e_h \in \mathbb{R}^{n_h}$, $e_m \in \mathbb{R}^d$ are the encoding vectors; x_t represents the input at the current time step t; $m_t \in \mathbb{R}^d$ denotes the unit's linear memory; h_t represents the non-linear hidden state; $\bar{A} \in \mathbb{R}^{d \times d}$ and $\bar{B} \in \mathbb{R}^{d \times 1}$ correspond to the discretized matrices provided by ODEs; n_h and n_x are the dimension of the hidden state and input vectors, respectively.

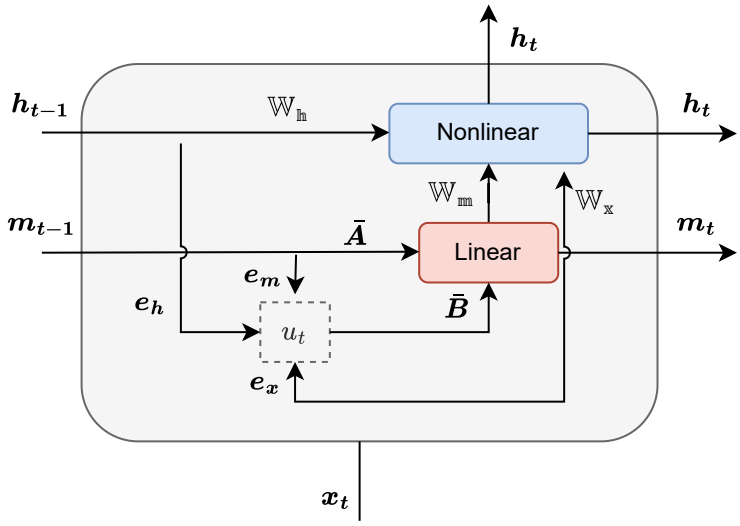

Figure 1.4: Simplified mathematical model of a LMU.

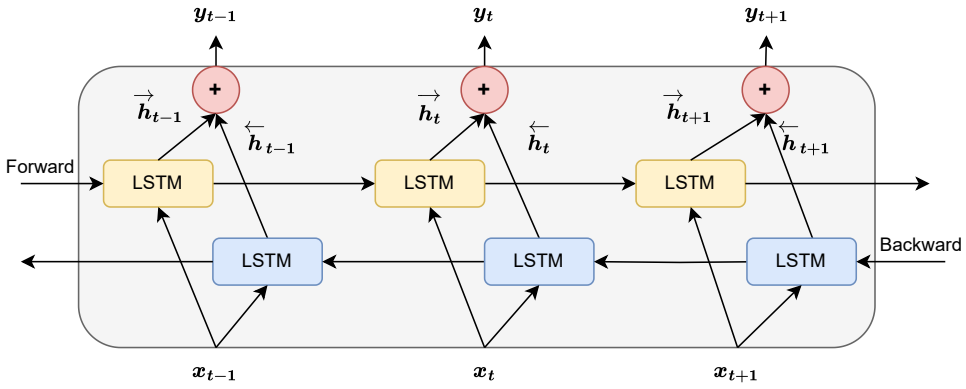

Figure 1.5: Simplified mathematical model of a BiRNN.

1.3.7 Bidirectional RNNs (BiRNNs)

Unlike the DL models discussed in Subsection 1.3.1÷1.3.3 (namely: SimpleRNN, LSTM, and GRU), in a Bidirectional RNN (BiRNN) data sequences are processed in two directions, namely *forward* and *backward*, as shown in Figure 1.5. This requires

to adopt and use two layers: the *first* layer uses the input data from start to end, while the *second* layer processes the input data reversely. Therefore, using two RNN layers to bidirectionally process the input data is expected to enhance the prediction accuracy.

1.3.8 Transformers

The Transformer model [21,22], introduced in 2017, revolutionized the processing of sequential data, with particular regard to Natural Language Processing (NLP) tasks. This is due to the fact that, unlike traditional sequence models (e.g., RNNs), Transformers are better suited to handle long-range dependencies. This is mainly due to their reliance on the so-called *attention mechanisms*, rather than sequential processing. In particular, a key component of the Transformer is the *scaled dot-product attention mechanism*, allowing this model to dynamically weigh the importance of different tokens in the input sequence—e.g., single words in a sentence to be analyzed by a NLP system.

From an analytical point of view, a Transformer operates by transforming the input into three matrices: *query* (denoted as $\mathbb{Q}$), *key* (denoted as $\mathbb{K}$), and *value* (denoted as $\mathbb{V}$) matrices. In detail, these matrices are expedient to calculate the following attention function:

$$\text{Attention}(\mathbb{Q},\mathbb{K},\mathbb{V}) = \text{softmax}\left(\frac{\mathbb{Q}\cdot\mathbb{K}^T}{\sqrt{d_k}}\right)\mathbb{V} \tag{1.16}$$

where: $\mathbb{Q}$ represents the query matrix, corresponding to a token in the input sequence; $\mathbb{K}$ corresponds to the key matrix; $\mathbb{V}$ represents the value matrix; the term $1/\sqrt{d_k}$ (where d_k represents the dimensionality of the keys) serves as scaling factor, ensuring stable gradients by normalizing the dot product between $\mathbb{Q}$ and $\mathbb{K}^T$.

It should be highlighted how Transformers use multiple instances of scaled dot-product attention mechanisms, obtaining the so-called *multi-head attention*. More in detail, on the basis of this mechanism, several attention functions (denoted as attention heads) are calculated in parallel, each using a different projection of the $(\mathbb{Q},\mathbb{K},\mathbb{V})$ matrices tuple. Thus, each attention head operates in a lower-dimensional subspace,

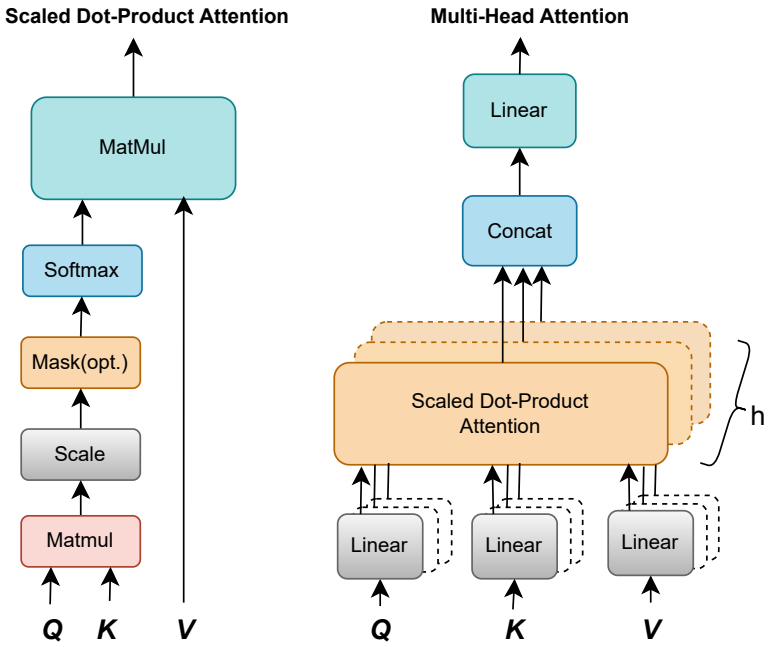

Figure 1.6: Scaled dot-product (left) and multi-head attention (right) mechanisms.

enabling the Transformer model to capture different aspects of the input sequence, while the outputs from these parallel attention functions are then concatenated and passed through a linear layer to produce the final output. Hence, the multi-head attention mechanism can be analytically represented as

$$\text{MultiHead}(\mathbb{Q},\mathbb{K},\mathbb{V}) = \text{Concat}(\text{head}_1,\text{head}_2,\ldots,\text{head}_h)\mathbb{W}^{(\text{O})} \tag{1.17}$$

where: h corresponds to the number of attention heads, and $\mathbb{W}^{(\text{O})}$ corresponds to a learnable output weight matrix [23–25].

For the sake of clarity, a graphical representation of scaled dot-product attention and multi-head attention mechanisms are shown in Figure 1.6.

1.4 Thesis Organization

Given the AI, ML and DL mechanisms described in the previous sections, in order to enable their application on tiny IoT devices—often facing significant resource constraints, such as limited computational power, memory, and energy—it is necessary the development of AI models that incorporate efficient approaches to address such limitations. The main objective of this Thesis dissertation is to explore AI models in the context of resource-constrained IoT devices for a variety of applications. In particular, this dissertation aims to develop, optimize, and deploy ML/DL models intended for embedded systems, considering a number of *trade-offs* between accuracy, computational efficiency, and constraints during deployment that are critical for the real-world evaluations. Thus, investigating the balance between computational complexity and prediction accuracy in the collaboration between AI and tiny IoT devices will provide a deeper understanding of their potential and assist in selecting the most feasible and efficient approaches depending on the application.

Hence, this Thesis aims at highlighting how embedded AI techniques can be effectively adapted and applied in heterogeneous domains, consistently focusing on their feasibility for constrained IoT systems. To this end, research activities span several domains—including environmental monitoring, healthcare, audio processing, human activity recognition, and financial forecasting—chosen to highlight different facets and challenges of deploying AI models on tiny IoT devices. While these areas might seem very different at first glance, they share some common characteristics, such as: real-time data streams, limited device resources, and the need for high predictive performance. This Thesis work positions them as case studies within a unified study on embedded intelligence.

First, the significance of AI in predicting outdoor environmental parameters, with a particular emphasis on air quality, is explored. Subsequently, indoor applications of AI, including audio processing, audio stream denoising, Parkinson's Disease (PD) detection using audio, and human activity recognition, are investigated. Finally, the potential of AI in the financial sector, specifically focusing on cryptocurrency price prediction, is examined.

To this end, each chapter reflects a specific task of the application of intelligent mechanisms to IoT devices—in most of the cases, being them tiny IoT nodes—performed in various and heterogeneous scenarios, then highlighting how different models can adapt and provide impact with embedded AI.

Chapter 2 presents the performance of different embedded ML/DL and quantized models concerning air quality prediction, based on data acquired from an indoor travelers transit area at the Brindisi airport in Brindisi, Italy. In detail, an analysis of different ML and DL models with respect to short-term and long-term predictions of Particulate Matter (PM)—with particular interest on PM2.5—where Bayesian optimization was used for hyperparameter tunning that yields the optimum performance, is presented. Then, among the compared models, it is shown that a particular model concatenation (namely, CNN-BiGRU) yields the best predictive performance, with a good balance between computational complexity. In this regard, CNN-Bidirectional GRU (BiGRU) has better potential to be deployed on resource-constrained devices. Further deployability of these models is explored by performing 8-bit quantization to see the impact of quantization on the prediction performance. Additionally, the chapter explores air quality forecasting for Beijing, leveraging atmospheric pollution data to compare multiple DL architectures in terms of both predictive accuracy and feasibility of deployment on resource-constrained devices.

The performance and deployability of several tiny audio denoising algorithms in resource-constrained contexts are benchmarked in **Chapter 3**, in detail aiming at introducing models providing strong audio denoising at a low level of complexity, making them suitable for use on restricted IoT devices. In particular, the computation complexity fingerprint of each model has been used to assess the performance of the DL model in addition to the quality of the improved signals in comparison to the reference (ground truth) signals.

Chapter 4 compares the performance of different ML and DL models for early detection of PD using a publicly available dataset of voice speech in Italian language. In detail, different algorithms have been evaluated in order to determine which are best in terms of the accuracy versus computational resources. It carries out the analytical comparison concerning both predictive performance and related practical chal-

lenges concerning their effective deployment on low-resources IoT devices. Quantization was used to overcome the limitations of computational and memory resources in embedded systems. Quantization reduces the computational complexity and memory footprint of DL models effectively while preserving the accuracy for efficient operation on edge devices.

Chapter 5 focuses on human activity classification within the tight resources provided by tiny IoT devices. In particular, novel methodologies that can be used to realize high accuracy in activity recognition while considering the edge devices' limited computation and energy capabilities, are presented, then investigating the *trade-off* between predictive accuracy and computational efficiency, providing a wide-ranging analysis of various models of DL in this scope. Moreover, this Chapter discusses the challenges of deploying DL models on resource-constrained devices and points out effective DL classifiers that provide a good balance between accuracy and resource requirements.

Chapter 6 discusses the use of DL models for the prediction of cryptocurrency prices. In detail, a comparative study is performed in order to determine the prediction capability of those models in highly volatile financial markets for three types of cryptocurrencies. The obtained analysis gives more insight into the performance of these models on both short- and long-term forecasting with substantial details about their capability in finding the challenges in cryptocurrency predictions. Apart from assessing predictive accuracy, the computational complexity for each model is comprehensively evaluated. The execution time, memory usage, and number of Multiply and ACCumulates (MACCs) are the metrics analyzed here, with the key objective of highlighting the *trade-offs* between accuracy and complexity. This holistic approach provides further understanding of the strengths and limitations of each model and is critical in giving insights on the accuracy-complexity *trade-offs* for each of the models. For the sake of completeness, while the cryptocurrency market may seem distant to the preceding topics discussed in this Thesis dissertation (which concentrate on AI-driven predictions for small IoT devices), forecasting in financial markets, including cryptocurrencies, exhibits similarities with IoT ecosystems, since they both involve real-time, low-latency predictions, and the capability to deal with dynamic

data effectively to yield highly accurate results.

Finally, in **Chapter 7** an overall summary of all the results achieved in this Thesis, is shown, showing how embedded AI can be effectively ported to resource-constrained contexts by addressing these numerous challenges.

Chapter 2

Air Quality Prediction via Embedded ML/DL and Quantized Models

2.1 Introduction

Environmental contamination (and, in particular, air pollution) has been a commonly discussed issue in recent years, with PMs—short particles or droplets suspended in the atmosphere having the potential to be breathed and penetrate the bloodstream—being a major concern due to their harmful impacts on human health [26,27]. In fact, the World Health Organization (WHO) estimates that illnesses caused by air pollution are responsible for more than 7M premature deaths annually [28]. Furthermore, according to [29], fine PMs' air pollution contributes to about 3% of global mortality from cardiopulmonary diseases, 5% of global mortality from lung, trachea and bronchial cancer, and 1% of global mortality from acute respiratory infections in children under the age of five. All of this amounts, annually, to 6.4 million life years lost and 0.8 million premature deaths. Then, air pollution poses risks also to plants and animals: with the air mainly composed of nitrogen (N_2, about 78%), oxygen (O_2, about 21%, particularly important for plants and animals' growth and devel-

opment), carbon dioxide (CO_2, in small percentage, useful for the photosynthesis of green plants), water vapor, and other compounds, an excessive increase of pollutants in the air could alter the correct development and lead to pathologies [30]. This further motivates the need to define proactive long-term strategies to mitigate pollution (as much as possible), especially for population well-being [31,32]. Therefore, such strategies might be enabled through enhanced monitoring systems defined on the basis of IoT-oriented well-known architectures and widely adopted processing and communication patterns—e.g., based on ML, DL, and quantized models [33]. In particular, this allows to take properly into account non-linear relationships between input variables (i.e., time series [34]) and process large amounts of data, in the end obtaining a higher prediction accuracy than that with traditional approaches [35].

In this study, a comparative performance evaluation (in terms of *trade-offs* between accuracy and complexity) of different ML/DL models (namely: RNN, CNN, LSTM, GRU, LMU, TCN, BiGRU, Bidirectional LSTM (BiLSTM), CNN-GRU, CNN-LSTM, CNN-BiGRU, and CNN-BiLSTM) used to predict $PM_{2.5}$-related Air Quality Index (AQI) [36], is discussed. Then, different evaluation metrics (namely: Root Mean Square Error (RMSE); Mean Absolute Percentage Error (MAE); Mean absolute percentage error (MAPE); coefficient of determination, denoted as R^2) and various values of the time lag—corresponding to the number of past observations to be considered to predict the next time step observation value—are considered to maximize estimation accuracy. Finally, a Post-Training Quantization (PTQ) technique is applied to evaluate the feasibility of further optimizing (as much as possible) DL models on *tiny* IoT devices—being resource-constrained in terms of memory and computational capabilities.

Hence, two experiments, focused on the prediction of indoor and outdoor $PM_{2.5}$ levels, are detailed in the following. In particular, the *first* experiment investigates the $PM_{2.5}$ concentration prediction using data collected inside the transit area of the Brindisi Airport, in the south of Italy, while the *second* study focuses on forecasting $PM_{2.5}$ levels in the city of Beijing, China, on the basis of outdoor air quality measurements. On the overall, the first case study contributes on the following aspects: (i) a comparative analysis of different ML and DL models for predicting air qual-

ity (in terms of $PM_{2.5}$ concentration) will be presented; (ii) the impact of time lag on the prediction accuracy will be investigated; (iii) the complexity and the memory space required to deploy and run the considered models on *tiny* IoT devices will be considered, in particular evaluating the adoption of PTQ techniques.

2.2 Literature Review

In the following, an overview on air quality prediction techniques proposed in the literature is presented, focusing on ML/DL approaches and on the use of IoT devices, respectively.

2.2.1 Air Quality Prediction through ML and DL Approaches

Several studies focus on RNN architectures [37], such as LSTM [38] and GRU [39], due to their ability to capture temporal dependencies in air quality samples. These models have shown to successfully predict pollutant concentrations with high accuracy.

In [40], a comprehensive comparison between ML algorithms—including Support Vector Machine (SVM), k-NN, and RF—to predict atmospheric $PM_{2.5}$ levels in the city of Isfahan, Iran, gathering data from 7 stations in the time period 2011–2019, with a total amount of 9 features, is presented. According to the obtained results, Artificial Neural Networks (ANNs) hit a 91.1% accuracy, followed by RF, SVM, and k-NN.

Authors in [41] conduct a comparison between LSTM and Seasonal AutoRegressive Integrated Moving Average with eXogenous regressor (SARIMAX) models [42] for $PM_{2.5}$ prediction in Bangkok in the time period 2017–2018. Their results show that LSTM achieves lower (in terms of 24 hour average value) RMSE and MAE values equal to 6.04 units and 4.86 units, respectively, while SARIMAX returns values equal to 7.99 units and 5.74 units, respectively.

In [43], a model, denoted as Aggregated LSTM (ALSTM), allows to forecast the concentration of $PM_{2.5}$ in five distinct industrial air quality sites in Taiwan. In detail, the adopted dataset consists of 17 environmental features collected in the time period

2012–2017, while the proposed model consists of stacked LSTM layers followed by fully-connected dense layers. The ALSTM model demonstrates superior accuracy when compared to LSTM, Gradient Boosted Tree Regression (GBTR) [44], and Support Vector Regression (SVR) [45] models.

Authors in [46] target the prediction of $PM_{2.5}$ in eight Korean cities, taking into account a dimensionality problem caused by a dataset with numerous variables and limited observations, and reducing it using RNN, LSTM, and BiLSTM models with Principal Component Analysis (PCA) [47]. The obtained results show that using PCA with LSTM and BiLSTM reduces RMSE and MAE by 16.6% and 33.3%, respectively.

In [48], a CNN-LSTM model is proposed in order to predict $PM_{2.5}$ in Beijing considering spatio-temporal correlations, over a dataset composed of 20,757 pollutant concentration and meteorological parameters observations and collected in the time period January 2015–March 2020. According to the obtained results, CNN-LSTM outperforms MultiLayer Perceptron (MLP) and LSTM. Moreover, a combination of Convolutional and BiGRU (CBGRU) model for $PM_{2.5}$ short-time prediction is proposed in [49]. In detail, the chosen dataset is composed of 43,800 8-feature hourly observations collected in the time period January 2010–December 2014, while the proposed model is composed of two 1D convolutional layers followed by two Bi-GRU layers. According to the obtained results, CBGRU provides a lower prediction error in comparison to DL models (including LSTM, GRU, and RNN). Similarly, the performance of LSTM, GRU, and BiLSTM are compared in [50], in order to predict $PM_{2.5}$ in three Korean cities (namely: Seoul, Daejeon, and Busan), using a 1-hour interval dataset collected in the time period May 2014–December 2021. The presented results show that BiLSTM demonstrates the highest performance in long-term predictions.

In [51], a stacked LSTM model is proposed to predict the next hour $PM_{2.5}$ concentration in the city of Istanbul, Turkey, training the model over a dataset collected in the time period January 2015–November 2019. According to the obtained results, the proposed model outperforms LSTM, RNN, and GRU, with an RMSE equal to 25.20 units.

An MLP-based model is proposed in [52] to predict the air pollutant hourly concentration (e.g., $PM_{2.5}$) by considering air quality data of the capital of Chile between 2014 and 2015. In [53], the authors apply a RNN-based model for $PM_{2.5}$ concentration prediction, using pollution data from the city of Seoul, South Korea, for a 3-year period (from 2015 to 2017), while an LSTM-based model for short-term $PM_{2.5}$ forecasting in Taiwan has been proposed in [54]. In the latter work, it is highlighted that LSTM has a lower error rate with respect to other methods, such as SimpleRNN and BiLSTM.

The authors in [55] compare the efficiency of LSTM and SimpleRNN in $PM_{2.5}$ prediction in Guangzhou city, experimentally verifying that LSTM performs better. In [56], the authors use LSTM on an IoT device for air pollution level forecasting and evaluate the prediction accuracy. Finally, BiLSTM- and GRU-based models for the prediction of $PM_{2.5}$ concentration in Beijing, China, are proposed in [57] and in [58], respectively.

Finally, a 1D-CNN-BiLSTM model with parallel input from the target and neighboring monitoring stations is presented in [59] for the hourly prediction of $PM_{2.5}$. As a result, the suggested method chooses neighboring stations based on distance and wind statistics, proving to be effective when a combined use of target and chosen monitoring stations is performed.

2.2.2 Air Quality Prediction on Resource-Constrained Devices

A tiny ML model to be used to predict the AQI using trained DT, RF, and eXtreme Gradient Boosting (XGBoost) models is proposed in [60]. Although faced with computational limitations, the authors achieve an accuracy of 75.2% adopting XGBoost, that outperforms both RF and DT while complying with a 2 MB memory limitation. Similarly, in [61] a low-cost device—based on a Raspberry Pi Pico as the main micro controller—for air quality monitoring and prediction is proposed. In detail, two models (composed of 2D convolutional layers making an AutoEncoder (AE)) are used: the first model forecasts CO_2, temperature, and humidity, by analyzing six hours of past observations; the second model (namely, a model imputer) is used for missing values imputation. Then, with model predictor's and model imputer's sizes

of 107,708 and 126,536 B, respectively, the light models' size decrease by 47.7% and 56.6%, respectively.

Finally, in [62] a comparative analysis and performance evaluation (in terms of computational complexity) of different ML and DL models on a tiny IoT device to forecast the $PM_{2.5}$ concentration is proposed. According to the obtained results, GRU provides the most accurate prediction with respect to the other considered models.

2.3 Air quality prediction in an indoor environment

The first scenario explores the use of different ML and DL models to predict $PM_{2.5}$ concentrations, exploiting a dataset fulfilled of air quality measurements gathered from an indoor transit area for travelers at the Brindisi Airport, in southern Italy [63].

2.3.1 Proposed Model

Based on the ML and DL methods described in Section 2.2, the following discussion highlights the various steps defined in the proposed comparative performance model, including data preparation (detailed in Subsection 2.3.1), Bayesian Optimization (BO) (detailed in Subsection 2.3.1), and the sliding window technique (detailed in Subsection 2.3.1).

Data Preparation

The data preparation step includes data correlation analysis (Subsection 2.3.1) and data normalization (Subsection 2.3.1), as follows.

Data Correlation Analysis

In order to perform a data correlation analysis, the Pearson Correlation Coefficient (PCC) [64] has been adopted. In detail, PCC (denoted as ρ) is a statistical metric accurately measuring importance and trend of the dependence between two variables,

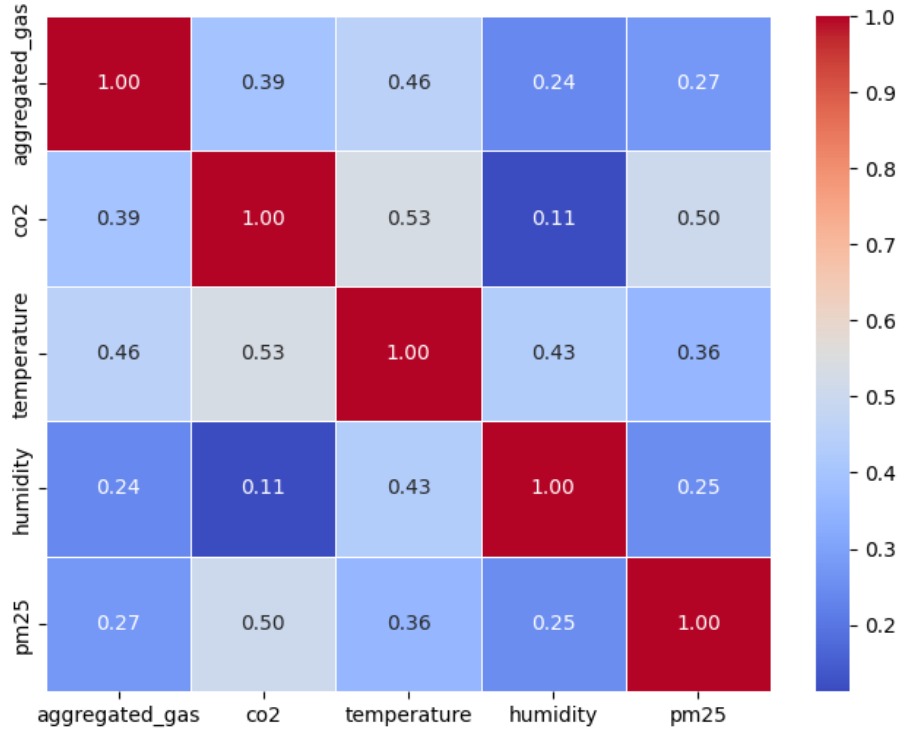

Figure 2.1: PCC of the features of the adopted dataset.

and defined as

$$\rho = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{2.1}$$

where: n corresponds to the number of observations; X_i and Y_i are individual data points; $\bar{X}$ and $\bar{Y}$ represent the means of the variables X and Y, respectively. To this end, PCC can range from -1 to 1: two variables with the same direction of variation lead to a positive correlation ($\rho = 1$); two variables with opposite directions of variation lead to a negative correlation ($\rho = -1$); finally, the absence of a linear dependency between the considered variables leads to $\rho = 0$. For the sake of completeness, the PCC obtained on the features still present in the adopted dataset—further described in Subsection 2.3.3—is shown in Figure 2.1.

Data Normalization

Data normalization eases the training process of NNs by ensuring that the different features representing the candidate dataset are mapped to a comparable scale, thus accelerating the convergence of different algorithms while enhancing the stability and efficiency of the proposed model. For this reason, the MinMaxScaler function [65], adopted in the proposed performance analysis, allows to map input data within a specific range (typically $[0,1]$ or $[-1,1]$) and can be defined as

$$x_{\text{scaled}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2.2}$$

where: x and x_{scaled} represent original and scaled value of a specific feature, respectively; $x_{\min}$ and $x_{\max}$ correspond to the minimum and maximum values of that feature, respectively.

Bayesian Optimization (BO)

BO represents a popular method for hyperparameter tweaking in DL models, as it allows performance optimization, reducing overfitting and underfitting, and improving generalization, resource efficiency, and durability [66]. These benefits become more and more evident when compared to traditional techniques (e.g., grid search and random search), which show significant limitations. In particular, grid search becomes computationally inefficient because of an exponential growing in the evaluations as the number of hyperparameters increases, while random search may waste resources by exploring poorly performing regions and fails to focus on optimal configurations [67]. In contrast, BO performs a global optimization of black-box functions by employing a probabilistic model, typically a Gaussian Process (GP), to more effectively explore the search space [68]. To this end, the hyperparameter optimization task can be formally defined as:

$$x^{\star} = \arg\min_{x \in X} f(x) \tag{2.3}$$

where $f(x)$ represents the objective function to be minimized, and x denotes the vector of hyperparameter configurations within the search space X. Consequently, the

Table 2.1: Accuracy of the considered ML and DL models in terms of the ratio r between GPs and RFs.

Model	RMSE Ratio r [%]			
	$w = 5$	$w = 10$	$w = 15$	$w = 20$
BiGRU	0,43%	0,07%	1,05%	-1,18%
BiLSTM	-1,93%	-0,47%	3,58%	-0,32%
CNN	1,85%	2,37%	1,20%	-0,09%
CNN-BiGRU	-0,91%	2,06%	1,13%	-7,35%
CNN-BiLSTM	-1,89%	-1,96%	-0,57%	-17,28%
CNN-GRU	-0,07%	0,22%	-0,17%	-1,09%
CNN-LSTM	3,29%	5,77%	2,31%	8,49%
GRU	-1,29%	-1,44%	0,49%	1,20%
LSTM	-1,62%	6,95%	3,23%	5,84%
RNN	3,45%	0,83%	0,58%	-0,21%

set of hyperparameters $x^\star$ will yield the minimal objective function value observed thus far.

On the basis of this modeling, in the following a performance comparison of the BO technique using different surrogate models, namely GPs and RFs [69,70], is detailed. Then, in order to compare these two models, the accuracy is evaluated through the RMSE, chosen as optimization metric and described in Subsection 2.3.4. Hence, a ratio (denoted as r) between GPs and RFs, with BO serving as the baseline model, is calculated as follows:

$$r = 1 - \left(\frac{\text{RMSE}_{\text{GP}}}{\text{RMSE}_{\text{RF}}} \right) . \tag{2.4}$$

Finally, in Table 2.1 the RMSE ratio r obtained by all the considered ML/DL models is shown, where it can be noted how GPs yield an average improvement (in terms of model accuracy) equal to 0.37%, at the same time not increasing the model's complexity.

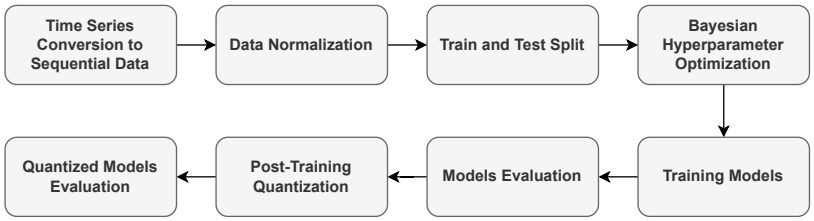

Figure 2.2: Flowchart of the proposed prediction process.

Sliding Window Technique

Finally, a sliding window technique has been adopted in the proposed model to convert time series data into sequential data, in order to forecast $PM_{2.5}$ levels for both short-term and long-term time horizons. In particular, this approach involves the evaluation of several time lags to identify the setting which enables the model to adequately represent temporal dependencies. The selected time lag allows to identify patterns and relationships between past $PM_{2.5}$ concentrations and future values by examining historical data through the sliding window. By integrating the relevant information from several time intervals, the sliding window approach improves the model's precision in predicting $PM_{2.5}$ levels across different time horizons.

For the sake of completeness and clarification, a graphical representation of the proposed prediction process (returning the experimental results further detailed in Section 2.3.2) is shown in Figure 2.2, while the detailed architectures of the proposed models are listed in Table 2.2.

2.3.2 Results

2.3.3 Dataset

The experimental dataset (publicly available at [71]) is composed of air quality measurements collected in an interior travelers transit area at the Brindisi airport, in the south of Italy [63] in the time period October 2022–October 2023. In detail, three inexpensive commercial sensors—namely, Adafruit MiCS5524, Sensirion SCD30, and

Table 2.2: Bayesian hyperparameters for the different time lags $\mathscr{W}$.

Model	$\mathscr{W} = 5$	$\mathscr{W} = 10$	$\mathscr{W} = 15$	$\mathscr{W} = 20$
BiGRU	neurons = 113 activation = relu	neurons = 128 activation = tanh	neurons = 32 activation = relu	neurons = 128 activation = relu
BiLSTM	neurons = 92 activation = tanh	neurons = 128 activation = relu	neurons = 32 activation = relu	neurons = 128 activation = relu
CNN	filters = 118 kernel_size = 3 activation = relu	filters = 128 kernel_size = 3 activation = relu	filters = 128 kernel_size = 3 activation = relu	filters = 128 kernel_size = 3 activation = tanh
CNN-BiGRU	neurons/filters = 68 kernel_size = 3 activation = relu	neurons/filters = 46 kernel_size = 3 activation = relu	neurons/filters = 128 kernel_size = 3 activation = relu	neurons/filters = 32 kernel_size = 3 activation = relu
CNN-BiLSTM	neurons/filters = 73 kernel_size = 3 activation = relu	neurons/filters = 128 kernel_size = 3 activation = tanh	neurons/filters = 128 kernel_size = 3 activation = tanh	neurons/filters = 128 kernel_size = 3 activation = relu
CNN-GRU	neurons/filters = 114 kernel_size = 3 activation = relu	neurons/filters = 83 kernel_size = 3 activation = relu	neurons/filters = 100 kernel_size = 3 activation = relu	neurons/filters = 128 kernel_size = 3 activation = relu
CNN-LSTM	neurons/filters = 98 kernel_size = 3 activation = relu	neurons/filters = 66 kernel_size = 3 activation = tanh	neurons/filters = 110 kernel_size = 3 activation = tanh	neurons/filters = 116 kernel_size = 3 activation = tanh
GRU	neurons = 32 activation = relu	neurons = 128 activation = relu	neurons = 84 activation = relu	neurons = 89 activation = relu
LMU	neurons = 96 activation = relu	neurons = 64 activation = relu	neurons = 128 activation = tanh	neurons = 128 activation = tanh
LSTM	neurons = 94 activation = relu	neurons = 128 activation = tanh	neurons = 128 activation = tanh	neurons = 56 activation = relu
RNN	neurons = 110 activation = relu	neurons = 128 activation = relu	neurons = 95 activation = tanh	neurons = 126 activation = relu
TCN	neurons/filters = 70 kernel_size = 3 activation = relu	neurons/filters = 119 kernel_size = 3 activation = tanh	neurons/filters = 128 kernel_size = 3 activation = relu	neurons/filters = 115 kernel_size = 3 activation = relu

Sensirion SPS30—have been used to collect air quality data, such as CO_2, temperature, humidity, aggregated gas levels, and $PM_{2.5}$, with a 2 sec sampling period. By employing the sliding window mechanism (detailed in Subsection 2.3.1) there has been the flexibility to re-sample the dataset at different sampling periods. The different combinations of sampling period (denoted as t_{sampling}, dimension: [s]), prediction horizon (denoted as s, dimension: [h]), and number of predictions (denoted as n_{pred},

Table 2.3: Combinations of sampling time t_{sampling}, prediction horizon s, and number of predictions n_{pred}, applied on the chosen air quality dataset.

$t_{\text{samplingC}}$	Prediction horizon s [h]	n_{pred} [num]
1 h	8	8
30 min	3	6
2 min	1	30

adimensional), are listed in Table 2.3.

In the following, This study will evaluate the performance using the dataset re-sampled as a 1-hour dataset ($t_{\text{sampling}} = 1$ hour). The performance has also been investigated using other sampling periods (the results are not shown for lack of space). The chosen combinations guarantee a sufficient level of prediction granularity. As expected, re-sampling the dataset with $t_{\text{sampling}} = 2$ min allows to predict $PM_{2.5}$ variations in the subsequent hour with a finer detail than in the case with $t_{\text{sampling}} = 1$ hour. Considering $t_{\text{sampling}} = 30$ min allows to improve the trend estimation accuracy.

2.3.4 Evaluation Metrics

The performance of the considered algorithms is evaluated on the basis of RMSE, MAE, MAPE, and Coefficient of determination (R^2), defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.5}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.6}$$

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{2.7}$$

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{2.8}$$

where: n corresponds to the number of data points in the test set; y_i represents the actual value; $\hat{y}_i$ represents the corresponding predicted value; and $\bar{y} = 1/n\sum_{i=1}^{n} y_i$ is

the mean value of data points.

These metrics collectively encompass the comprehensive evaluation of a model's performance, highlighting feature relationships, predicting value deviations, and temporal disparities.

As shown in Figure 2.3, RMSE has been designated as benchmark metric for both training and testing phases. Then, each metric is assessed for various values of the time lag (denoted as $\mathscr{W}$), leading to a fair performance comparison of the considered ML and DL models. In fact, less architectural complex models may experience good overall performance with small values of $\mathscr{W}$, but may face challenges in identifying correlations for long-term predictions, particularly with larger input dimensions, compensating with higher complexity layers. In most cases, increasing the historical knowledge yields to improved accuracy across various models, as shown in Figure 2.4, with CNN-BiGRU returning the best accuracy among the considered models.

It might also be reasonable for model predictions to become less accurate for increasing values of the prediction horizon s: if s is extended, then the model needs to identify additional correlations within the data and this may become increasingly challenging and computationally heavy. As a reference, in Figure 2.5 all the metrics for a given time lag $\mathscr{W}$ and dataset are shown, while Figure 2.6 shows the increasing prediction error as a function of the time horizon. It can be observed that, by increasing s, ML and DL models lose their ability to predict variations, while they still capture the trend in long-term predictions.

Finally, in Figure 2.7 an analysis of the performance of various models across three key metrics—namely, average MAE, MAPE, and RMSE—evaluated over all the considered time lags $\mathscr{W} = \{5, 10, 15, 20\}$ and all the considered prediction horizons $s \in \{1, \ldots, 8\}$, is provided. The obtained results show that CNN-BiGRU achieves (on average) the lowest average error, thus highlighting its superior prediction performance with respect to the other considered DL models.

2.3.5 Model Complexity

In order to better focus on the integration of ML and DL models into IoT nodes (often represented by limited, if not constrained, resources), in addition to the performance shown in Subsection 2.3.4, the computational complexity of the considered models has been further investigated considering the number of MACC operations (denoted as n_{MACC}, adimensional), the RAM usage (dimension: [KiB]), and the execution time (denoted as t_{exec}, dimension: [ms]), defined as follows:

$$t_{\text{exec}} = \frac{n_{\text{MACC}} \cdot 11}{f_{\text{board}}} \tag{2.9}$$

where: f_{board} corresponds the frequency of the STM32F469I-DISCO board [72], equal to 180 MHz; n_{MACC} is multiplied by 11 because of the requirement of 11 cycles per MACC (on average, with minimum and maximum values equal to 8.688 and 16.487, respectively, as further detailed in Table 2.5) on that board—this value has been achieved considering the average n_{MACC} for all of the evaluated models, except for LMU and TCN.

The above metrics have been evaluated exploiting the STM32CubeMX tool [8], which allows a seamless import of AI models onto different tiny IoT board, such as STM32F469I-DISCO board [73]. This has been kept as reference board for the evaluation of the models' complexities which are shown in Table 2.4. As can be seen from Table 2.4, among all the models deployable on the chosen IoT board, CNN is the most computationally-efficient (in terms of n_{MACC}) for all time lags, with GRU and RNN ranking as second and third best performing models. At the opposite, CNN-BiLSTM has the higher computational complexity, closely followed by BiLSTM. Then, models with balanced computational efficiency across various time lags include CNN-BiGRU, BiGRU, GRU, and LSTM.

Finally, even if LMU and TCN represent novel architectures for time-series prediction (with LMU achieving a high accuracy with a low complexity), for completeness it should be highlighted that they are currently not natively supported by ST Edge AI Developer Cloud [74], a remote tool exploited to run the considered ML and DL models on a remote tiny IoT board directly in the cloud (before being deployed on real devices). Nevertheless, knowing the functional parameters of the chosen IoT

board, it has been possible to estimate n_{MACC} and RAM usage also for TCN and LMU.

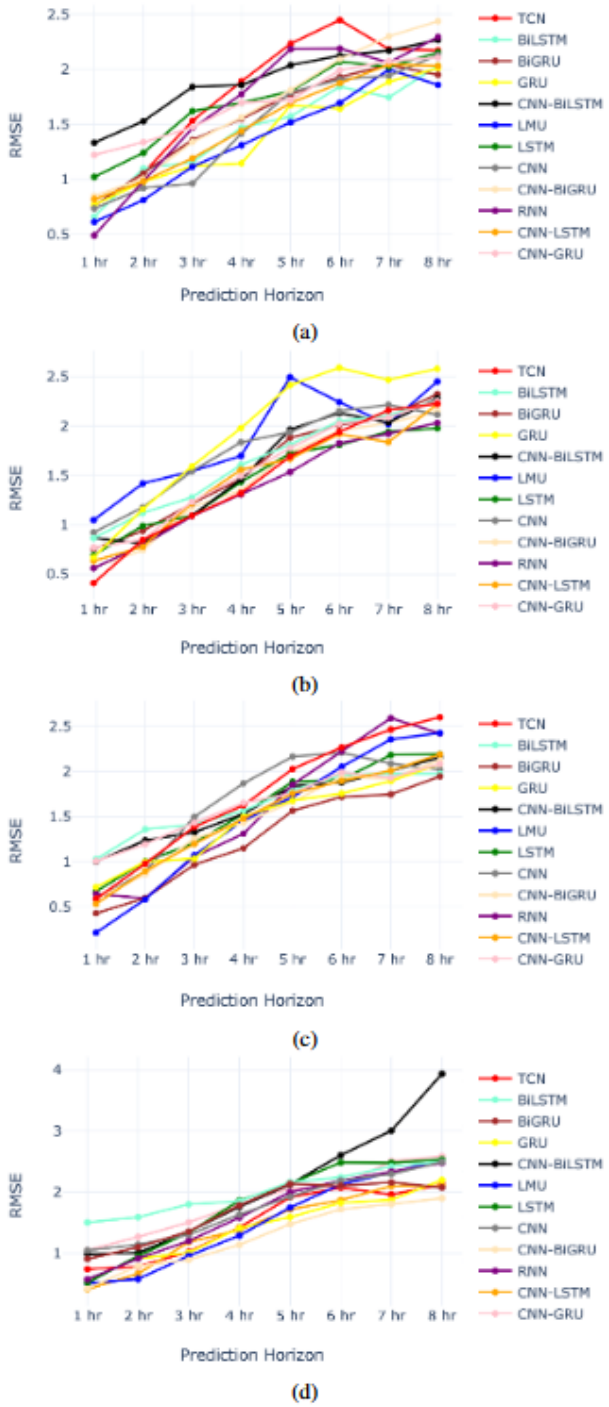

Figure 2.3: Performance of the considered ML and DL models on the basis of RMSE (with lower values of RMSE being the better) for different time lags $\mathscr{W}$: (a) $\mathscr{W} = 5$, (b) $\mathscr{W} = 10$, (c) $\mathscr{W} = 15$, (d) $\mathscr{W} = 20$.

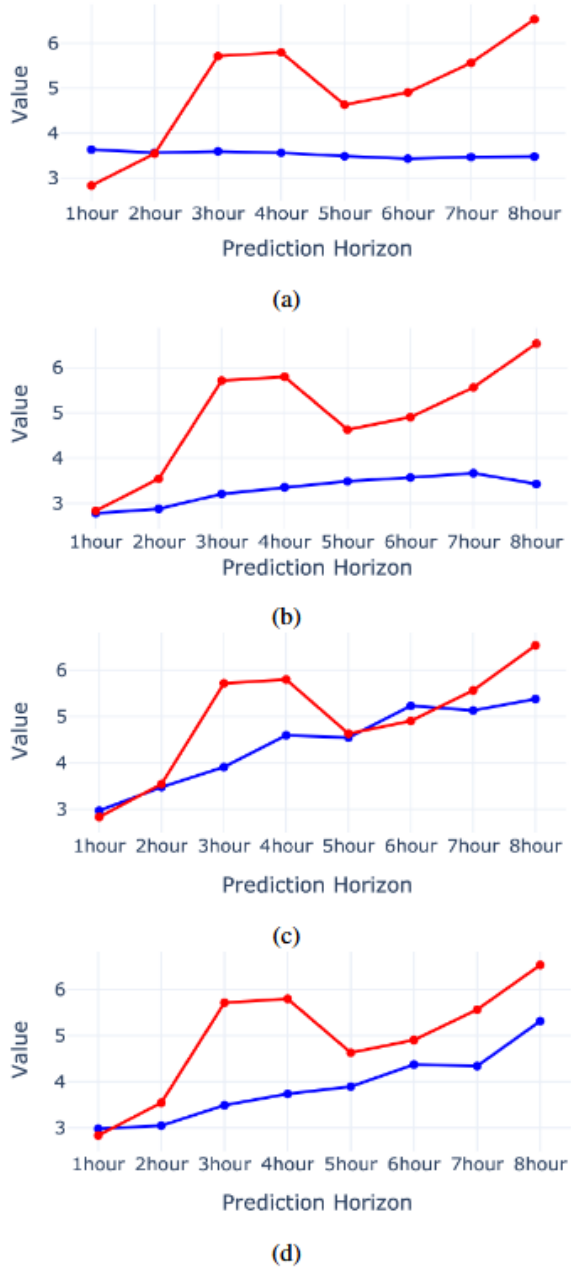

Figure 2.4: Performance of the BiGRU model for different time lags $\mathscr{W}$: (a) $\mathscr{W} = 5$, (b) $\mathscr{W} = 10$, (c) $\mathscr{W} = 15$, (d) $\mathscr{W} = 20$.

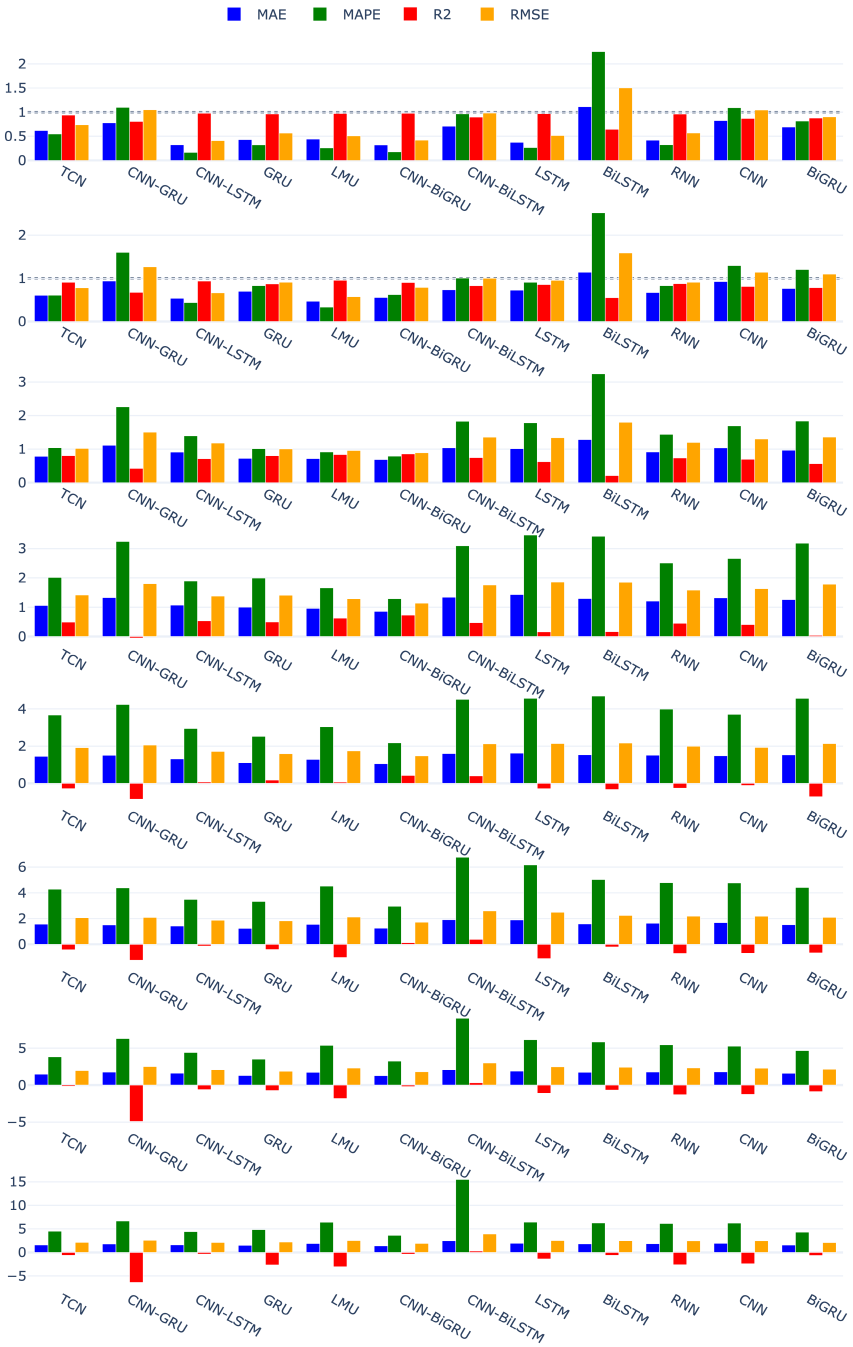

Figure 2.5: Evaluated metrics for each analyzed ML and DL model considering a time lag $\mathscr{W} = 20$, with (a) 1-hour prediction horizon ($s = 1$) and (b) 8-hour prediction horizon ($s = 8$).

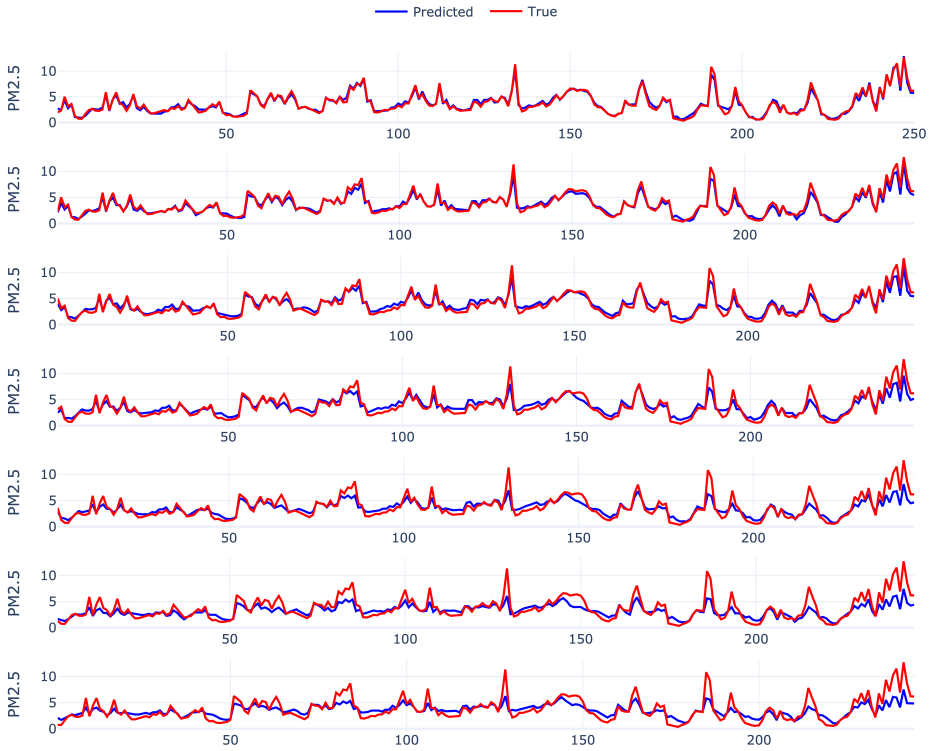

Figure 2.6: Prediction performance of the CNN-BiGRU model with a time lag $\mathscr{W} = 20$ across each prediction hour, from 1-hour ($s = 1$) to 8-hour prediction ($s = 8$).

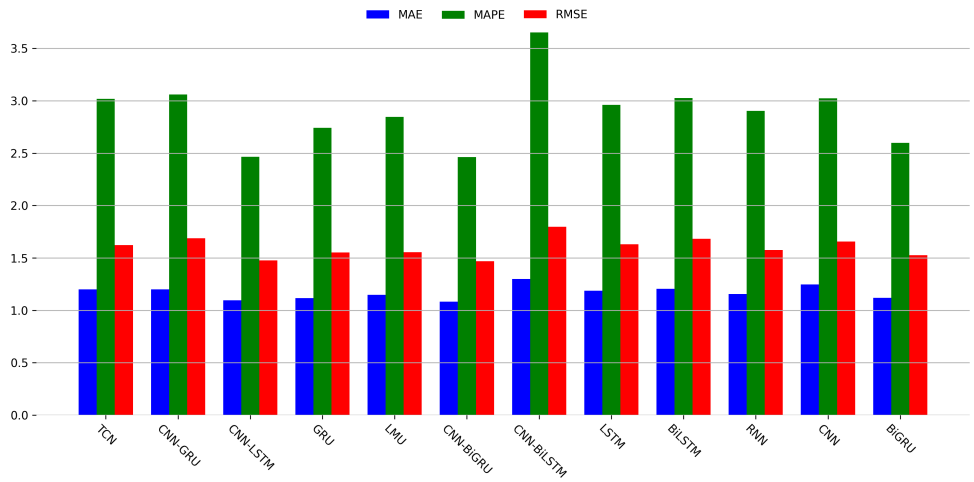

Figure 2.7: Average MAE, MAPE, and RMSE of the evaluated DL models considering all the time lags $\mathscr{W} = \{5, 10, 15, 20\}$ and all the prediction horizons $s \in \{1, \ldots, 8\}$.

Table 2.4: MACC operations n_{MACC}, RAM usage, and execution time t_{exec} returned by the analyzed ML and DL models, as a function of different time lags $\mathscr{W}$.

Model	n_{MACC}				RAM (KiB)				t_{exec} (ms)			
	$\mathscr{W}=5$	$\mathscr{W}=10$	$\mathscr{W}=15$	$\mathscr{W}=20$	$\mathscr{W}=5$	$\mathscr{W}=10$	$\mathscr{W}=15$	$\mathscr{W}=20$	$\mathscr{W}=5$	$\mathscr{W}=10$	$\mathscr{W}=15$	$\mathscr{W}=20$
BiGRU	442,047	1,079,656	112,040	2,103,912	6.76	4.2	4.43	7.54	24.67	61.55	9.07	116.00
BiLSTM	389,816	1,425,512	150,440	2,800,232	6.59	7.94	4.66	8.14	26.13	82.86	13.78	161.5
CNN	49,546	112,616	161,256	227,816	4.57	6.14	7.64	8.64	2.87	7.37	9.92	15.17
CNN-BiGRU	185,404	217,546	2,640,104	236,296	6.73	6.69	14.31	7.08	10.79	14.03	134.90	16.30
CNN-BiLSTM	278,526	2,186,216	3,502,056	4,829,416	7.35	12.44	14.93	17.43	16.45	119.6	185.1	255.3
CNN-GRU	267,662	357,585	824,804	1,845,480	7.58	7.99	1094	15.64	14.01	19.93	45.65	95.91
CNN-LSTM	257,256	305,486	1,329,014	2,030,916	7.43	7.36	12.26	15.02	14.24	19.72	76.13	115.9
GRU	20,616	547,688	354,164	522,534	3.45	6.45	5.07	5.27	1.63	30.06	21.29	32.46
LMU	97,892	87,652	112,228	112,228	//	//	//	//	6.11	5.35	6.85	6.85
LSTM	207,374	724,972	1,067,752	286,936	5.85	7.05	7.05	4.70	12.84	44.36	65.62	23.47
TCN	568,400	1,642,676	1,900,544	1,534,100	//	//	//	//	34.73	100.3	116	93.75
RNN	89,424	205,928	164,929	366,008	3.81	4.12	3.92	4.39	4.388	9.94	9.72	17.82

Table 2.5: Evaluation cycle per MACC for various values of the time lags $\mathscr{W}$.

Model	$\mathscr{W} = 5$	$\mathscr{W} = 10$	$\mathscr{W} = 15$	$\mathscr{W} = 20$
BiGRU	10.045	10.261	14.571	9.924
BiLSTM	12.065	10.462	16.487	10.381
CNN	10.426	11.779	11.073	11.985
CNN-BiGRU	10.475	11.608	9.197	12.416
CNN-BiLSTM	10.630	9.847	9.513	9.515
CNN-GRU	9.421	10.032	9.962	9.354
CNN-LSTM	9.963	11.619	10.310	10.272
GRU	14.231	9.879	10.820	11.181
LSTM	11.145	11.013	11.062	14.723
RNN	8.836	8.688	10.608	8.763

2.3.6 Post-Training Quantization (PTQ)

Finally, in order to further optimize DL models predicting $PM_{2.5}$ values on tiny IoT devices, an additional investigation has been performed applying a PTQ technique exploiting the TensorFlow Lite (TFLite) library [75]. In detail, by implementing PTQ, a significant decrease (in terms of models' sizes) has been accomplished at the cost of a minimal prediction accuracy decrease. Nevertheless, despite this slight performance reduction, the benefits of reduced model size and improved computational efficiency outweigh the accuracy *trade-off*. In fact, as highlighted in Table 2.6 (comparing the sizes of original and TFLite-compressed DL models), PTQ provides significant benefits in practical situations, especially with resource-constrained systems when computational effectiveness is crucial.

As a matter of fact, PTQ with TFLite shows a great potential if applied to efficient and real-time $PM_{2.5}$ monitoring systems, by reducing the computational complexity during model deployment. Finally, it can be noted that reducing the size of the models by means of quantization does not impact heavily on the accuracy, as shown in Figure 2.8, where a (negligible) 1% accuracy reduction is experienced by CNN-BiGRU

Table 2.6: Comparison (in terms of mean size) between original DL models and their corresponding TFLite-compressed versions.

Model	Mean original model size [KiB]	Mean TFLite model size [KiB]	Model size reduction [%]
BiGRU	1347.28	455.86	66.16
BiLSTM	1484.96	499.86	66.34
CNN	1308.85	431.29	67.04
CNN-BiGRU	1157.80	390.12	66.30
CNN-BiLSTM	3143.70	1052.67	66.51
CNN-GRU	1178.98	392.40	66.71
CNN-LSTM	1246.75	414.85	66.72
GRU	551.09	183.82	66.64
LSTM	873.60	291.91	66.58
RNN	537.26	177.76	66.91

despite a 66% average model's size's reduction.

2.3.7 Conclusions

This study presents a comparative study of air quality estimation, specifically targeting resource-constrained IoT devices. To achieve this objective, several ML and DL models (namely: RNN, CNN, LSTM, GRU, LMU, TCN, BiGRU, BiLSTM, CNN-GRU, CNN-LSTM, CNN-BiGRU, CNN-BiLSTM) have been considered for both short-term and long-term prediction of the $PM_{2.5}$ value, exploiting Bayesian optimization to select the best hyperparameters for model training. Among the analyzed algorithms, CNN-BiGRU shows the highest predictive accuracy and provides an effective trade-off between computational complexity (in terms of number of MACCs), memory usage (in terms of RAM), and execution time. In general, while some DL models may excel in one of the considered performance metrics, they often perform

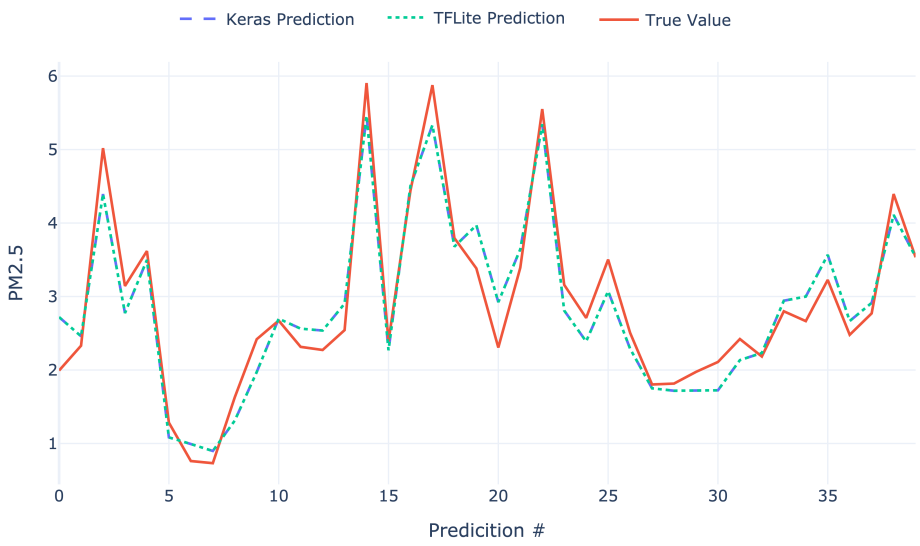

Figure 2.8: Comparison (in terms of prediction accuracy) between true $PM_{2.5}$ values and predicted values obtained through original and quantized CNN-BiGRU models.

poorly in others. Finally, in order to investigate the impact of 8-bit quantization on prediction performance, PTQ has been applied to the considered models: quantized CNN-BiGRU achieves approximately a 66% model size reduction with a prediction accuracy reduction equal to only 1% (on average).

2.4 Air quality prediction in an outdoor environment

The second scenario focuses on the prediction of $PM_{2.5}$ values using a dataset containing air quality observations collected in Beijing [76]. Various ML/DL models are applied and compared to evaluate their prediction accuracy and their suitability for deployment on resource-constrained edge devices. The main goal of this study is to jointly investigate accuracy and complexity on tiny IoT implementations to identify the algorithm guaranteeing the best trade-off.

2.4.1 Dataset Analysis

A dataset from University of California Irvine ML Repository [76], containing 43,820 hourly-collected data (namely: $PM_{2.5}$ concentration, dew point, temperature, pressure, combined wind direction, cumulated wind speed, cumulated hours of snow, and cumulated hours of rain), is used as reference dataset in the current study. In order to use this dataset as an input for the ML and DL algorithms of interest and to improve data quality for more accurate predictions, a data transformation is expedient. This is relevant also because duplicate or missing values may give an incorrect view of the overall data statistical distribution, as outliers and inconsistent data usually tend to impair the model's overall learning, thus leading to inaccurate predictions.

The observation timestamp should *first* be assigned as dataset index. Since the input dataset had 2,067 missing $PM_{2.5}$ concentration values, each of these missing values has been filled with the data associated with the previous timestamp. Alternative ways for filling the missing values can be considered, such as using the average value of the $PM_{2.5}$ column or exploiting the KNNImputer library [77]. Nevertheless, experimental evaluations using the GRU model returned an RMSE (dimension: [ppm]) equal to 25.1 and 26.32 for the average $PM_{2.5}$ value and the KNNImputer library, respectively. However, these approaches lead to RMSE values greater than that obtained associating the previous timestamp, which is equal to 24.67. Therefore, filling the missing value at a given timestamp with the value associated with the previous timestamp is the best strategy. Furthermore, on the basis of the $PM_{2.5}$ values contained in the dataset, an air quality label for each record has been determined, as shown in Table 2.7. The distribution of data, according to the chosen labeling, is shown in Figure 2.9.

In order to evaluate the data and achieve the appropriate prediction accuracy, it is necessary to investigate the correlation between the most influential features. As a result, the Pearson correlation coefficient R between all possible pairs of types of data (e.g., $PM_{2.5}$ and dew, $PM_{2.5}$ and temperature, etc.) is shown in Table 2.8.

After preprocessing, a time series is obtained, defined as a collection of observations indexed over a period of time. The following general notation is used for a time

Table 2.7: Air quality according to $PM_{2.5}$ values.

$PM_{2.5}$	**Air Quality**
0 – 12.0	Good
12.1 – 35.4	Moderate
35.5 – 55.4	Unhealthy for sensitive groups
55.5 – 150.4	Unhealthy
150.5 – 250.4	Very Unhealthy
250.5 – 500.4	Hazardous

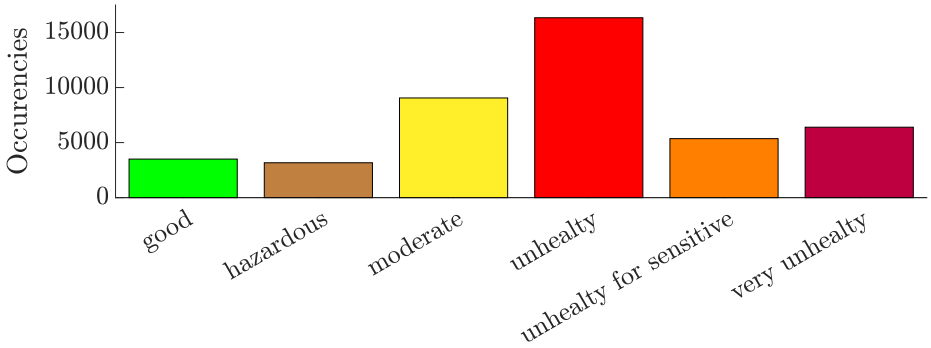

Figure 2.9: Distribution (in terms of occurrences) of the air quality labels applied to the input dataset.

Table 2.8: Correlation coefficient R between pairs of types of data available in the dataset.

	$PM_{2.5}$	Dew	Temperature	Pressure	Wind direction	Wind speed	Snow hours	Rain hours
$PM_{2.5}$	1.00	0.17	−0.09	−0.05	0.20	−0.25	0.03	−0.04
Dew	0.17	1.00	0.83	−0.75	0.24	−0.31	−0.02	0.12
Temperature	−0.08	0.83	1.00	−0.80	0.19	−0.16	−0.09	0.06
Pressure	−0.07	−0.75	−0.78	1.00	−0.17	0.19	0.08	−0.08
Wind direction	0.20	0.24	0.19	−0.17	1.00	−0.21	0.02	−0.06
Wind speed	−0.25	−0.29	−0.16	0.17	−0.1	1.00	0.03	−0.02
Snow hours	0.03	−0.04	−0.10	0.06	0.02	0.01	1.00	−0.02
Rain hours	−0.04	0.12	0.04	−0.08	−0.06	−0.01	−0.2	1.00

series:

$$\{x_t\} \quad t \in T \tag{2.10}$$

where: $T \subseteq \mathbb{N}^+$ is the observation period; and x_t is the specific observation at time t.

As time series forecasting corresponds to a technique using both past (historical) and present data to predict the value at a specific future instant.

2.4.2 Performance Evaluation

In the following, a performance evaluation of the ML and DL algorithms, in terms of both prediction accuracy and computational complexity, is performed. In detail, the implementation and training of these ML and DL algorithms is based on the use of the Keras framework [78], with TensorFlow [79] as back-end and the scikit-learn library [80]. ADAM and RMSProp optimizers are used to check and compare the prediction performance in the case of DL algorithms.

Evaluation Metrics

In order to evaluate the *accuracy* of a prediction algorithm, proper performance metrics need to be selected as loss functions. To this end, MSE, MAE, RMSE, R^2, and Symmetric Mean Absolute Percentage Error (SMAPE), defined as follows:

$$\mathrm{MSE}(y,\hat{y}) \triangleq \frac{1}{N}\sum_{i=0}^{N-1}(y_i-\hat{y}_i)^2 \tag{2.11}$$

$$\mathrm{MAE}(y,\hat{y}) \triangleq \frac{1}{N}\sum_{i=0}^{N-1}|y_i-\hat{y}_i| \tag{2.12}$$

$$\mathrm{RMSE}(y,\hat{y}) \triangleq \sqrt{\mathrm{MSE}(y,\hat{y})} \tag{2.13}$$

$$R^2(y,\hat{y}) \triangleq 1-\frac{\sum_{i=0}^{N-1}\left|y_i-|\hat{y}_i|\right|}{\sum_{i=0}^{N-1}|y_i-\bar{y}|} \tag{2.14}$$

$$\mathrm{SMAPE}(y,\hat{y}) \triangleq \frac{100\%}{N}\sum_{i=0}^{N-1}\frac{\left|y_i-|\hat{y}_i|\right|}{\frac{|y_i|-|\hat{y}_i|}{2}} \tag{2.15}$$

where: N is the total number of samples in the test set; $y=(y_0,\ldots,y_{N-1})$ is the actual (target) values vector; $\hat{y}=(\hat{y}_0,\ldots,\hat{y}_{N-1})$ is the vector of predicted values; and $\bar{y} \triangleq \sum_{i=0}^{N-1} y_i/N$ is the arithmetic average of actual values.

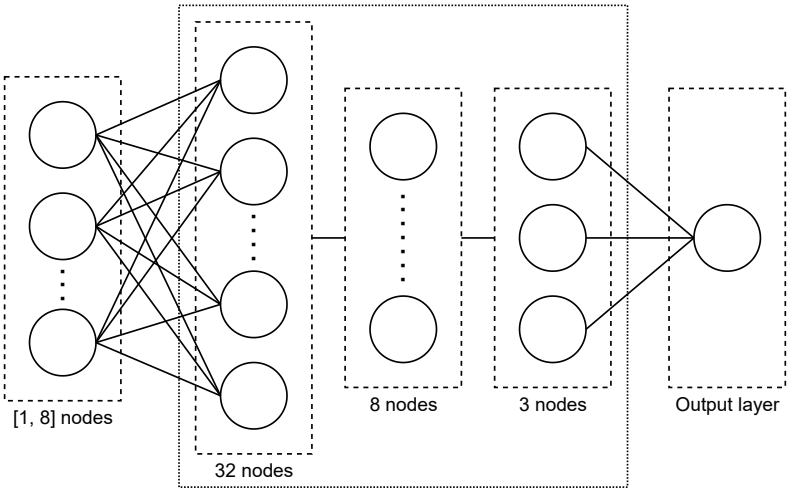

Figure 2.10: NN architecture.

In order to evaluate the complexity, the following metrics are considered: MACC operations; the size of the expected Read/Write (RW) memory chunk used to store the intermediate inference computing values ("RAM"); and the size of generated Read-Only (RO) memory spaces to store the weight/bias parameters ("ROM/Flash").

Proposed Model

In order to perform a fair performance analysis of the ML and DL algorithms, the same input dataset and the same architecture are considered. In particular, as shown in Figure 2.10, the common architecture consists of: three hidden layers with 32, 8, and 3 neurons, respectively; an input layer containing a number of neurons equal to the number of features contained in the considered dataset, which is 8. The ReLU function [81] (defined as $x^+ \triangleq \max(0,x)$, where x is the input of a neuron) is considered as the activation function in these NNs, while the total number of samples is 43,820 and the sizes of the training and test sets are equal to 35,040 and 8,780 samples, respectively. The following parameters, identified through various experiments, have been set to train the NNs: learning rate equal to 0.0015; batch size equal to 72; and number of epochs equal to 100.

Table 2.9: Experimental accuracy performance of k-NN, DTR and SVR algorithms.

	k-NN	DTR	SVR
RMSE	**31.87**	38.93	47.1754
MAE	**20.58**	21.28	40.1736
SMAPE	**31.90**	38.93	58.87
R^2	**0.87**	0.814	0.7471

Table 2.10: Experimental accuracy performance with the ADAM optimizer, considering a 1-step ahead prediction horizon ($s = 1$) and a time lag $\mathscr{W} = 1$.

	MLP	SimpleRNN	LSTM	BiLSTM	GRU
RMSE	57.78	30.32	29.43	30.84	**27.44**
MAE	45.22	19.70	17.27	18.30	**13.18**
SMAPE	59.25	28.30	26.47	27.81	**18.77**
R^2	0.591	0.887	0.894	0.883	**0.907**

Table 2.11: Experimental accuracy performance with the RMSProp optimizer, considering a 1-step ahead prediction horizon ($s = 1$) and a time lag $\mathscr{W} = 1$.

	MLP	SimpleRNN	LSTM	BiLSTM	GRU
RMSE	61.30	29.60	29.97	30.45	**24.67**
MAE	49.60	21.15	21.99	22.63	**13.76**
SMAPE	57.38	34.96	36.25	37.04	**22.29**
R^2	0.540	0.8928	0.890	0.886	**0.925**

Accuracy Assessment

The accuracy performance of ML algorithms (namely, k-NN, DTR and SVR) is shown in Table 2.9: it can be concluded that k-NN is the best performing algorithm in terms of air quality prediction accuracy. Table 2.10 and Table 2.11 show the experimental results of prediction algorithms by considering two different optimizers (ADAM and RMSProp), a 1-step ahead prediction horizon ($s = 1$), and a time lag $\mathscr{W} = 1$. In this case, GRU outperforms the other algorithms in terms of air quality prediction accuracy.

Since ANNs and ML algorithms do not perform very well with respect to DL algorithms, in the following the focus will only be on RNNs. In Table 2.12, extends the performance analysis of Table 2.11 considering the RMSProp optimizer and various

Table 2.12: Experimental accuracy performance of DL algorithms with the RMSProp optimizer, considering a 1-step ahead prediction horizon ($s = 1$) and various values of $\mathscr{W}$.

Method	Metric	$\mathscr{W} = 4$	$\mathscr{W} = 12$	$\mathscr{W} = 24$
GRU	RMSE	20.342	**20.027**	20.724
	MAE	11.422	**11.004**	11.718
	R^2	0.951	**0.948**	0.958
	SMAPE	18.1	**17.4**	18.2
LSTM	RMSE	22.154	21.189	22.527
	MAE	12.985	12.105	12.718
	R^2	0.945	0.947	0.949
	SMAPE	29.4	20	21.3
SimpleRNN	RMSE	21.218	21.541	21.817
	MAE	12.722	12.467	12.678
	R^2	0.940	0.946	0.946
	SMAPE	21	19.4	23.5
BiLSTM	RMSE	21.675	21.018	20.989
	MAE	12.271	12.380	11.973
	R^2	0.947	0.948	0.948
	SMAPE	20.2	25.2	21.4
CNN	RMSE	21.893	21.631	22.266
	MAE	12.359	12.48	12.963
	R^2	0.896	0.912	0.880
	SMAPE	21.5	21.2	21.6

values for the time lag $\mathscr{W}$ (namely: 4, 12, and 24): the best performing algorithm is GRU, with RMSE minimized in correspondence to a time lag $\mathscr{W} = 12$. This might be intuitive, as $\mathscr{W} = 12$ allows GRU to predict the air quality considering half a day as previous time steps, and leads as a reasonable trade-off among short (i.e., 1 hour) and long (i.e., 24 hours) time lags.

Furthermore, an additional performance evaluation with a 3-step ahead prediction horizon (i.e., $s = 3$), whose results are shown in Table 2.13, has been performed. Even in this case, GRU guarantees the highest accuracy. Finally, for the sake of clarity, in Figure 2.11 the real data samples are directly compared with the values predicted by GRU (i.e., the best algorithm) with $\mathscr{W} = 1$ and $s = 1$. A very high estimation accuracy can be observed.

To conclude, on the basis of the experimental results obtained in predicting the

Table 2.13: Experimental accuracy performance of DL algorithms with the RMSProp optimizer, considering a 3-step ahead prediction horizon ($s = 3$) and various values of $\mathscr{W}$.

Method	Metric	$\mathscr{W} = 4$	$\mathscr{W} = 12$	$\mathscr{W} = 24$
GRU	RMSE	22.463	**22.012**	22.327
	MAE	13.549	**13.253**	13.684
	R^2	0.835	**0.813**	0.8159
	SMAPE	34.8	**34.0**	35.5
LSTM	RMSE	24.035	24.777	24.862
	MAE	16.403	16.764	16.137
	R^2	0.779	0.774	0.784
	SMAPE	38.4	41.6	37.2
SimpleRNN	RMSE	24.298	24.428	24.117
	MAE	16.427	16.752	16.766
	R^2	0.766	0.772	0.728
	SMAPE	39.1	42.2	39.1
BiLSTM	RMSE	22.675	23.162	22.231
	MAE	13.625	14.112	13.7
	R^2	0.816	0.0872	0.825
	SMAPE	35.1	39.2	36.3
CNN	RMSE	25.035	24.862	25.471
	MAE	15.707	15.412	15.601
	R^2	0.766	0.787	0.794
	SMAPE	48.1	49.2	46.3

air quality with 1-step and 3-step ahead prediction horizons ($s = 1$ and $s = 3$, respectively) and taking into account different time lags $\mathscr{W}$, it has been highlighted how GRU allows to take into account short-term and long-term dependencies, generally returning a more accurate prediction with respect to other methods.

Computational Complexity Assessment

In addition to the experimental results presented in Section 2.4.2, time and computational complexities are other relevant criteria for the selection and application of prediction algorithms for IoT applications: the lower the algorithm complexity, the highest the applicability to devices with limited resources [82].

The DL algorithms listed in Table 2.12 have been further evaluated, in terms of

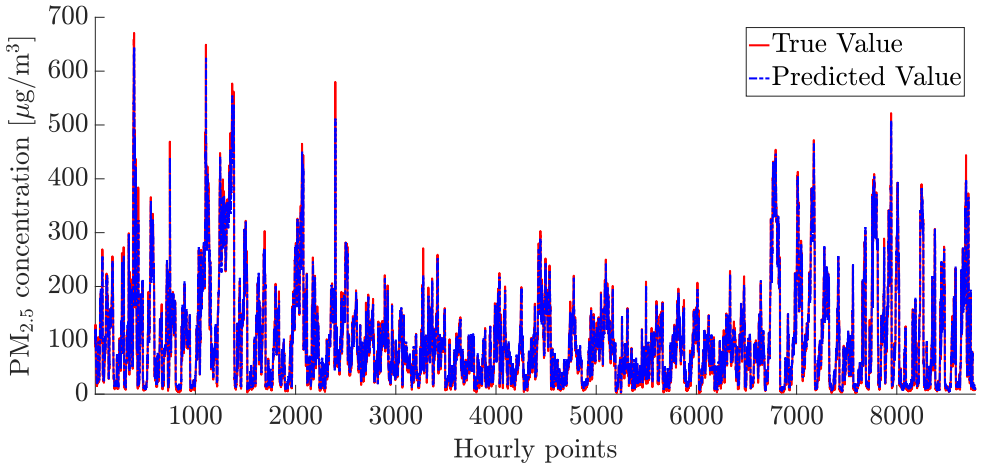

Figure 2.11: $PM_{2.5}$ concentration prediction results obtained by GRU, with a time lag $\mathscr{W} = 1$, and considering a 1-step ahead prediction horizon ($s = 1$).

computational complexity, through the STM32Cube tool [8] on the STM B-L475E-IOT01A1 board [83]. The obtained experimental results are shown in Table 2.14, considering the metrics introduced in Subsection 2.4.2. From the obtained results, it can be concluded that SimpleRNN has the lowest overall complexity, whereas GRU (the best performing algorithm, in terms of prediction accuracy, as shown in Subsection 2.4.2) has an high complexity in comparison to SimpleRNN.

Joint Accuracy-Complexity Trade-Off

Finally, in order to investigate the joint computational-accuracy performance, it is of interest to consider a performance metric which captures jointly the computational complexity and the accuracy. For this purpose, the following are considered:

- the normalized SMAPE, defined as $\frac{\text{SMAPE}}{100} \in (0,1)$, as error metric;
- various computational complexity metrics: weights, total RAM, MACCs, and execution time.

Table 2.14: Experimental computational complexity of the RNNs of interest.

Criterion	SimpleRNN	LSTM	BiLSTM	GRU
Total Flash [KiB]	**20.48**	41.81	36.30	38.13
Weights [KiB]	**6.28**	22.03	14.03	16.90
Library [KiB]	**14.21**	19.78	22.28	21.23
Total RAM [KiB]	**3.50**	4.25	4.60	4.03
Activations [KiB]	**1.12**	1.75	1.31	1.62
Library [KiB]	**2.38**	2.50	3.29	2.40
MACCs [num]	**31,794**	127,026	77,874	94,002
Execution time [ms]	**0.023**	0.092	0.060	0.101

Table 2.15: Joint accuracy-computational performance (in terms of ξ) of the RNNs.

Criterion c	SimpleRNN	LSTM	BiLSTM	GRU
Weights [KiB]	1.19	1.53	1.702	**0.8866**
Total RAM [KiB]	**2.125**	7.9308	5.1911	3.718
MACCs [num]	**10,809.96**	45,729.36	28,813.38	20,680.44
Execution time [ms]	**0.00782**	0.03312	0.0222	0.02222

Denoting as c one of the selected computational complexity metrics, the joint complexity-accuracy performance metric, defined as ξ, can be expressed as follows:

$$\xi = \frac{\text{SMAPE}}{100} c\,. \tag{2.16}$$

The obtained results are shown in Table 2.15, where it can be seen that, depending on the considered computational complexity metric, the RNNs which guarantee the best trade-off are SimpleRNN and GRU.

2.4.3 Conclusions and Future Works

This study presents an experimental comparison of ML algorithms (k-NN, DTR, SVR, ANN) and DL algorithms (SimpleRNN, LSTM, BiLSTM, CNN, GRU) for air quality prediction, focusing on the accuracy-complexity trade-off for their applicability to IoT systems. According to the experimental results, the DL algorithms are superior (in terms of prediction accuracy) to the ML algorithms, when dealing with time series data since they can learn better long-term dependencies and relationships.

In comparison to KNN, when GRU is used, the RMSE is reduced by about 15%. Depending on the selected computational complexity metric, SimpleRNN (total RAM, MACCs, execution time) or GRU (weights) are the ones guaranteeing the best trade-off. Additionally, in future research activities mobile tiny IoT devices [84] could be used to predict air quality.

Chapter 3

Deep Neural Quantization for Speech Detection of Parkinson Disease

3.1 Introduction

PD is a neuro-degenerative disorder that impacts the central nervous system and has a profound effect on the well-being of over 10% of elderly individuals globally [85]. Unfortunately, the number of people affected by PD is expected to rise from about 4M in 2005 to approximately 9M in 2030 [86]. More in detail, the initial phase of the PD is marked by symptoms including rapid eye movement sleep behavior disturbance, hyposmia (diminished sense of smell), constipation, and speech changes. Furthermore, the PD impacts the motor system, resulting in symptoms such as hand and leg tremors, bradykinesia (slowness in movement), and disruption of bodily balance and cognitive issues, such as depression and nervousness [87,88]. The onset of PD is likely the result of a combination of genetic and environmental variables, interacting with each other during the process of brain aging [89]. The presence of PD is mostly declared on the basis of clinical observations because there is currently a lack of accurate and proven biomarkers to diagnose the course of the disease [90].

In order to try to fill this "gap" in diagnosis accuracy, the adoption of enhanced mechanisms should be considered. To this end, DL techniques represent effective "tools" to early detect PD, because of their ability to efficiently analyze intricate and multi-dimensional data, such as medical images, speech recordings, or gait signals, which are commonly used in diagnosing PD [91].

Given these potentialities, in this study an experimental performance evaluation of different DL models in diagnosing PD, based on voice signals, is presented and discussed. In detail, in addition to "canonical" ML models (namely: *k*-NN; AdaBoost; DT; RF and DL mechanisms (namely: MLP; LSTM; GRU; CNN; LMU; TCN, also Quantized DL (QDL) one-dimensional CNN (1D-CNN) and MLP models (namely: Q1D-CNN and QMLP) have been considered and evaluated on a STM32U5 tiny IoT MCU.[1] The achieved results allow to demonstrate the deployability of the considered ML/DL/QDL models on off-the-shelf and inexpensive MCU. In particular, RF and 1D-CNN outperform the other ML/DL algorithms and QDL models enable a substantial flash memory saving (compared to non-quantized models) while maintaining a relatively high accuracy.

3.2 Literature Review

With regards to the use of ML and DL voice analysis methods to detect PD, in [92] different DL models are employed in order to identify PD looking at two distinct symptoms, namely gait and speech disorders. In detail, the proposed model is composed of 1D-Conv layers and is compared with SVM, XGBoost, and MLP models, in the end achieving (by looking at the outcome obtained by analyzing speech data) an accuracy equal to 89.15%, while the considered reference models return accuracies of 81.16%, 77%, and 85.60%, respectively. Similarly, in [93,94] various ML models, including SVM, RF, *k*-NN, and logistic regression, for PD early recognition, are considered: RF achieves a detection accuracy and a sensitivity equal to 91.83% and 0.95, respectively.

[1] B-U585I-IOT02A Discovery Kit. `https://www.st.com/en/evaluation-tools/b-u585i-iot02a.html`.

In [95], the authors explore the application of SVM models with various kernel types for the classification of PD based on voice samples. More in detail, Perceptual Linear Prediction (PLP) and Reflexive Structural PLP (RASTA-PLP) are used for voice extraction, then obtaining that the highest classification accuracy (equal to 74%) is achieved by adopting a PLP feature extractor and an SVM model with a Radial Basis Function (RBF) kernel.

A novel approach for detecting PD from speech signals, involving a combination of 2D-Conv and 1D-Conv layers, is discussed in [96], using two datasets consisting of recordings of voices in Spanish and Chinese languages. According to the results obtained with both these languages, the proposed models return accuracies of 75.3% and 92%, respectively. An enhanced version of the AlexNet model, incorporating Mel-frequency Cepstral Coefficients (MFCCs) features, is employed in [97] to accurately predict the presence of PD by analyzing voice samples, in the end returning a notable level of precision, with a classification accuracy equal to 95%.

Finally, in [98] DL models (including CNNs and Bidirectional LSTM, Bi-LSTM) are used for PD detection, in detail applying a 10-fold cross validation to train the models. On the basis of the achieved results with a short sentence voice signal, CNNs and Bi-LSTM reach classification accuracies equal to 83.52% and 84.29%, respectively.

3.3 Dataset

The reference dataset adopted in this study for PD detection corresponds to a dataset comprising voice recordings of Italian speech and vowel sounds obtained from a total of 65 people, divided as follows: (i) 15 young healthy individuals aged 19-29 years, including 13 men and 2 women, kept as “reference” because of their neuromotor characteristics helping in defining a reliable lower limit of the “systematic” error rate for each word composing the dataset; (ii) 22 senior individuals aged 60-77 years, including 10 men and 12 women, all being in healthy condition and none reporting particular speech or language disorders; and (iii) 28 individuals (19 men and 9 women) aged 40-80 years and diagnosed with PD [99,100].

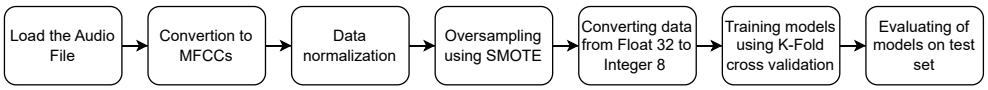

Figure 3.1: Proposed PD early detection procedure.

Overall, the reference dataset comprises 373 audio recordings from individuals without PD and 437 audio recordings from individuals diagnosed with PD. All audio recordings were collected at a 16 KHz sampling rate. On the practical side, the protocol followed to generate the dataset required the completion of the following tasks: (i) reading a phonetically balanced text (spaced by a pause); (ii) executing the syllables `pa` and `ta` (with an intermediate pause); (iii) two phonations of the vowels `a`, `e`, `i`, `o`, `u`; and (iv) reading a list of some phonetically balanced words and phonetically balanced sentences (spaced by a pause).

3.4 Proposed model

An overview of the operational procedure applied to the reference dataset is provided to prepare it as input for the ML and DL models, as illustrated in Figure 3.1. Next, the models selected for the experimental comparison are described.

3.4.1 Data Pre-Processing

Feature Extraction

To extract relevant features from the audio recordings in the reference dataset, the MFCCs are initially derived from the voice samples, with 40 MFCCs extracted for each sample. MFCCs are widely used in audio signal processing for tasks such as voice recognition and sound classification [101].

Table 3.1: Network architecture of the evaluated DL models.

1D-CNN	**GRU**
Conv1D (filters = 128, kernel_size = 3)	GRU (layer_size = 128)
MaxPooling1D	Dropout (0.2)
Conv1D (filters = 128, kernel_size = 3)	Dense (layer_size = 128, activation = relu)
MaxPooling1D	Dense (layer_size = 64, activation = relu)
Dense (layer_size = 128, activation = relu)	Dropout (0.2)
Dense (layer_size = 64, activation = relu)	Dense (layer_size = 24, activation = relu)
Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 24, activation = relu)	Dense (layer_size = 1, activation = relu)
Dropout (0.2)	
Dense (layer_size = 1, activation = relu)	
LSTM	**MLP**
LSTM (layer_size = 128)	Dense (layer_size = 128, activation = relu)
Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 128, activation = relu)	Dense (layer_size = 128, activation = relu)
Dropout (0.2)	Dense (layer_size = 64, activation = relu)
Dense (layer_size = 64, activation = relu)	Dropout (0.2)
Dense (layer_size = 24, activation = relu)	Dense (layer_size = 24, activation = relu)
Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 1, activation = relu)	Dense (layer_size = 1, activation = relu)
LMU	**TCN**
LMU (layer_size = 128)	TCN (filters = 128, kernel_size = 3, dilations = [4, 8, 16])
Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 128, activation = relu)	Dense (layer_size = 128, activation = relu)
Dense (layer_size = 64, activation = relu)	Dense (layer_size = 64, activation = relu)
Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 24, activation = relu)	Dense (layer_size = 24, activation = relu)
Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 1, activation = relu)	Dense (layer_size = 1, activation = relu)

Data Normalization

The data normalization (considering the MFCCs features mentioned before) is performed on each of the 810 audio recording composing the dataset, as follows:

$$\psi_{\text{norm}_i} = \frac{\psi_i - \mu_i}{\sigma_i + \varepsilon} \tag{3.1}$$

where: ψ_i represents the 40 MFCCs extracted from the i-th audio recording comprised in the reference dataset ($i \in 1, \ldots, 810$); μ_i denotes the mean value calculated

over all the 40 MFCCs ψ_i; σ_i represents the standard deviation of the 40 MFCCs; ε is a small constant (equal to $1 \cdot 10^{-10}$) added to the denominator to prevent division by zero.

Synthetic Minority Over-Sampling Technique (SMOTE)

SMOTE is a technique used to handle imbalanced class distributions in classification problems [102]. In detail, since conventional ML techniques tend to prioritize the dominant class, resulting in decreased results when one class is significantly under-represented, SMOTE addresses this issue by generating synthetic samples for the minority class. Since, as detailed in Section 3.3, the reference dataset is imbalanced, SMOTE is applied to balance the class distribution and ensure equal representation for each class. finally obtaining 437 data samples for both healthy and unhealthy classes.

3.4.2 Data Conversion to 8-bit Integer Representation

Given the need, as detailed in Section 3.1, to be able to run the ML/DL/QDL models on tiny constrained IoT devices and due to their general well-known restrictions (i.e., in terms of restricted memory resources of embedded devices), a data conversion from their input form (typically, floating point) to a corresponding 8-bit integer representation was beneficial, in the end resulting in reduced memory usage and power consumption. To this end, since a quantization of the models themselves might be useful, especially looking for profiling the performance trends with respect to non-quantized models, in the current study data quantization has been implemented by applying a Quantization Aware Training (QAT) [103] during the models' training, with floating point values being converted to a smaller range of distinct (usually integers) values. For the sake of clarity, it should be highlighted that, unlike post-training quantization techniques, QAT integrates the impact of quantization directly into the training process, without causing considerable accuracy degradation.

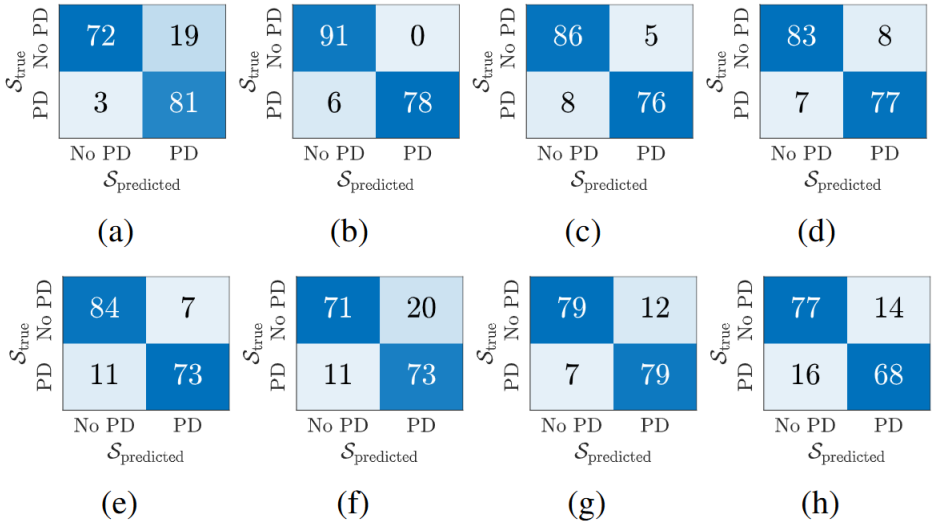

Figure 3.2: Confusion matrices of the considered DL and QDL models: (a) LSTM, (b) 1D-CNN, (c) MLP, (d) GRU, (e) LMU, (f) TCN, (g) Q1D-CNN, (h) QMLP.

3.4.3 k-Fold Cross Validation

In order to evaluate the performance and the accuracy of the discussed ML/DL methods, in this study, the k-fold cross-validation method has been utilized, which splits the reference training dataset into $k = 10$ equal-size subsets. Then, each subset is used as validation set, while the remaining $k - 1$ subsets are used for training, maximizing in this way the usage of available data—this is effective for studies with limited data. Moreover, this k-fold approach provides several advantages, preventing overfitting, guaranteeing reliable accuracy, and allowing model generalization. At the end, the training and the test sets were composed of 629 and 175 samples, respectively. For the sake of completeness, a brief summary of the architecture of the DL models of interest is presented in Table 3.1.

Table 3.2: Experimental performance results of the considered ML, DL, and QDL models.

	Model	Accuracy	Precision	Recall	F1 Score	ROC_{AUC}
ML	AdaBoost	0.9257	0.9176	0.9285	0.9230	**0.9833**
	DT	0.8628	0.8571	0.8571	0.8571	0.8576
	k-NN	0.8914	**0.9710**	0.7961	0.8758	0.9650
	RF	**0.9428**	0.9302	**0.9523**	**0.9411**	0.9827
DL	LSTM	0.8742	0.8100	**0.9642**	0.8804	0.9480
	1D-CNN	**0.9657**	**1.0000**	0.9285	**0.9629**	**0.9869**
	MLP	0.9257	0.9382	0.9047	0.9212	0.9774
	GRU	0.9142	0.9058	0.9166	0.9112	0.9631
	LMU	0.8971	0.9125	0.8690	0.8902	0.9625
	TCN	0.8228	0.7849	0.8690	0.8248	0.8767
QDL	Q1D-CNN	**0.9028**	**0.8681**	**0.9404**	**0.9028**	**0.9675**
	QMLP	0.8285	0.8292	0.8095	0.8192	0.9272

Table 3.3: Computation complexity of the considered ML, DL, and QDL models.

	Model	n_{MAC} [num]	Flash [KiB]	RAM [KiB]	t_{exec} [ms]	Memory Usage [MB]
ML	AdaBoost	2004	128	89	0.7401	0.049
	DT	11	10	2	0.01233	0.004
	k-NN	-	-	-	-	0.110
	RF	1134	130.3	1.99	0.6065	0.429
DL	LSTM	2,694,099	384	7	149.2	1.116
	1D-CNN	1,005,651	761	23	52.41	2.240
	MLP	31,955	135	4	1.390	0.409
	GRU	2,018,259	320	7	129.7	0.929
	LMU	52,840	421	–	–	1.286
	TCN	1,894,129	1,070	–	–	3.218
QDL	Q1D-CNN	917,272	114.15	–	–	1.463
	QMLP	31,256	31.37	–	–	0.432

3.4.4 Performance Metrics

The performance of the considered ML, DL, and QDL models has been evaluated relying on the following metrics.

- *Accuracy*, defined as the ratio between the total number of correct decisions

and the total number of decisions:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{3.2}$$

where: TP is the number of true positives; TN is the number of true negatives; FP is the number of false positives; and FN is the number of false negatives.

- *Precision*, defined as the ratio between correctly predicted positive observations and the number of all predicted positive observations:

$$Precision = \frac{TP}{TP+FP}. \tag{3.3}$$

- *Recall*, defined as the ratio between the number of correctly identified positive observations and the total number of actual positive observations:

$$Recall = \frac{TP}{TP+FN}. \tag{3.4}$$

- *F1 score*, corresponding to the harmonic mean of precision and recall:

$$\begin{aligned} F1 &= \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \\ &= \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \end{aligned} \tag{3.5}$$

- *Matthews Correlation Coefficient (MCC)*, corresponding to a measure between the observed and predicted binary classifications, and ranging between -1 and 1 [104]:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}. \tag{3.6}$$

In particular, MCC values have the following meanings: $+1$ means perfect prediction; 0 identifies a no better than random prediction: and -1 stands for total disagreement between prediction and observation

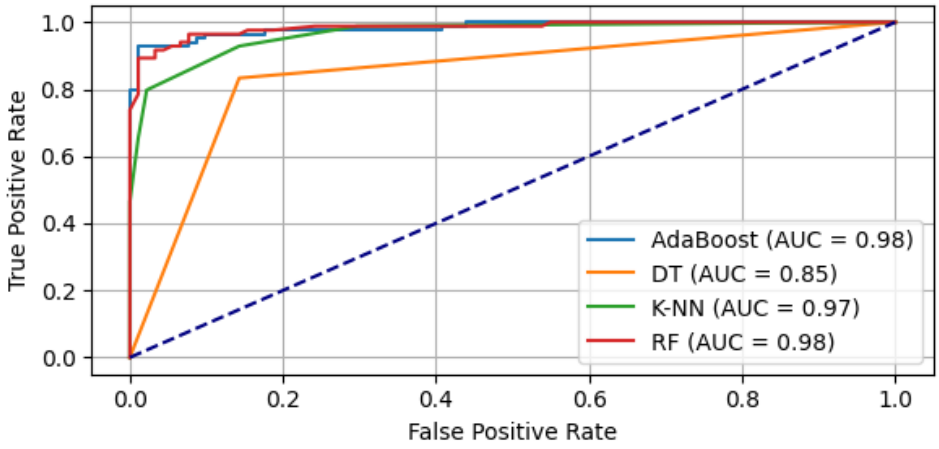

Figure 3.3: ROC curves of the considered ML models.

3.5 Results and Discussion

The performance and accuracy of the ML models (namely: k-NN, AdaBoost, DT, RF) and DL models (namely: MLP, LSTM, GRU, 1D-CNN, LMU, TCN) are evaluated for the early detection of Parkinson's disease (PD) using voice speech recordings. For the sake of clarity, in Figure 3.2 the confusion matrices returned by each DL and QDL model of interest—comparing the true values, $\mathscr{S}_{\text{true}}$, with the predicted ones, $\mathscr{S}_{\text{predicted}}$—are shown, while the performance metrics (detailed in Subsection 3.4.4) returned by all involved models, as well as the **Area Under the ROC Curve!** (**Area Under the ROC Curve!**)—measuring the entire two-dimensional area underneath the entire Receiver Operating Characteristic (ROC) curve—are listed in Table 3.2. Moreover, the ROC curves themselves—representing the trade-off between TP rate and FP rate for different threshold values—for the considered ML, DL, and QDL models are shown in Figure 3.3 (ML) and Figure 3.4 (DL and QDL), respectively.

On the basis of the results presented in Table 3.2, it can be observed that RF exhibits a superior performance (with accuracy equal to 94.28%) with respect to the

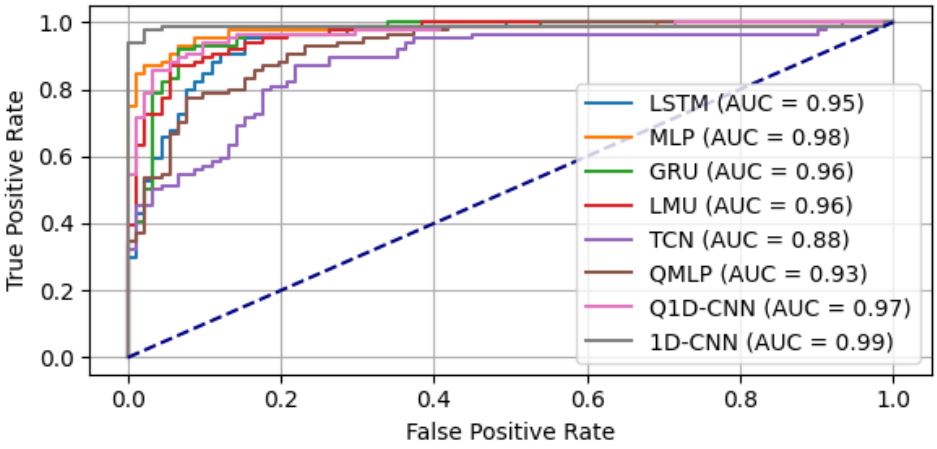

Figure 3.4: ROC curves of the considered DL and QDL models.

other considered ML models, while, on the other hand, 1D-CNN attains an accuracy equal to 96.58%. This is also confirmed by looking at the MCCs of the considered models, shown in Figure 3.5, where the obtained performance results identify RF and 1D-CNN as models yielding the highest values, equal to 0.897 and 0.933, respectively.

Based on the results presented in Table 3.2, Quantized Keras (QKeras)[2] is applied. to quantize 1D-CNN and MLP models using 4-bit activation and 8-bit weights (in the end obtaining Q1D-CNN and QMLP models), since they outperformed the others DL models. The effectiveness of this quantization strategy is confirmed also by the experimental computational complexity detailed in Table 3.2, in terms of (i) the amount of MACC operations (denoted as n_{MAC}), (ii) the required FLASH and RAM memory, (iii) the execution time (denoted as t_{exec}), and the memory used to store the model on the device (dimension: [MB]), in order to determine which model is most suitable for implementation inside tiny (constrained) IoT devices. More in detail, the experimental computational complexity results have been derived by exploiting the

[2]Deep Quantized Neural Network Support. `https://wiki.st.com/stm32mcu/wiki/AI:Deep_Quantized_Neural_Network_support`.

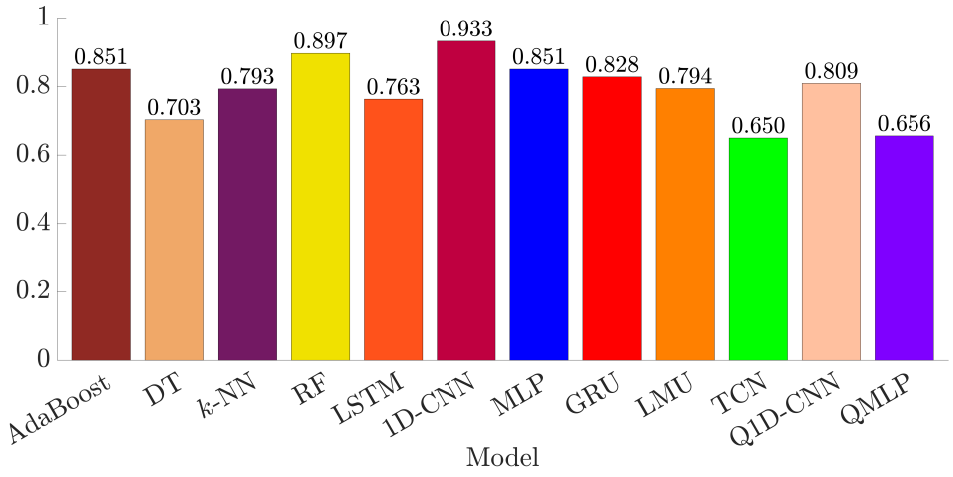

Figure 3.5: MCCs of the considered ML/DL/QDL models.

STM32Cube AI Developer Cloud.[3] Unfortunately, as of today, this cloud platform is not yet able to support some ML/DL models (namely: *k*-NN, LMU, TCN, QMLP, and Q1D-CNN) due to their lack of compatibility—this is the reason why some results in Table 3.3 are missing. Moreover, it should be noted that the weights for ML and DL models are represented as 32-bit floating point (`float32`), while QDL models are represented as 8-bit signed integer (`int8`).

Looking at the results presented in Table 3.3, it can be noted that both Q1D-CNN and QMLP demonstrate a substantial reduction in terms of flash memory usage, if compared to their non-quantized counterparts—namely, an approximate reduction of 84.99% in the case of Q1D-CNN/1D-CNN, and an approximate reduction of 76.77% in the case of QMLP/MLP—while maintaining a relatively high accuracy equal to 90.28% and 82.85%, respectively. Moreover, Q1D-CNN requires 34.6% less memory than 1D-CNN.

[3] STM32Cube.AI Developer Cloud. `https://stm32ai-cs.st.com/`.

3.6 Conclusions

In this study, a comparison between different ML and DL models (as well as a quantized version of two DL mechanisms) for early PD detection, applied to a public Italian voice speech dataset, has been presented. The deployability of selected DL models over tiny constrained IoT devices has been assessed exploiting the STM32Cube AI Developer Cloud online tool. On the basis of the achieved experimental results, 1D-CNN is the best performing DL model in PD detection, with accuracy, recall, and Area Under the ROC Curve (AUC) equal to 96.57%, 96.29%, and 0.9869, respectively. Subsequently, the introduction of Q1D-CNN and QMLP (in order to rely on quantized DL for this kind of health disorder recognition) highlighted how a NN model's quantization yields a reduction of flash memory footprint on the order of 84.99% and 76.77% with respect to 1D-CNN and MLP, respectively, but also (unfortunately) an accuracy decrease by 6.73% and 11.08% with respect to 1D-CNN and MLP, respectively.

Chapter 4

Benchmarking Tiny Audio Denoising with Deployability Constraints

4.1 Introduction

Speech enhancement is recognized as being a crucial domain of study and implementation in the field of digital signal processing and audio engineering. Telecommunications, voice-controlled devices, hearing aids, and visual entertainment are just a few of the many contexts where speech enhancement is required [105]. Background noise, reverberation, and interference are well-known environmental factors that can cause speech signals to deteriorate [106], and where spectral subtraction [107], Wiener filtering [108], and signal subspace approach [109] are notable classical methods of speech enhancement.

Recently, DL has emerged as an effective approach for addressing these problems. In fact, using DL methods, the noisy input signal can be effectively represented and used to rebuild a clean signal [110]. Notable benefits of DL models compared to traditional methods are (i) their ability to capture non-linear correlations between noisy and clean speech signals, (ii) their adaptability to various forms of noise and

acoustic situations, and (iii) their training on extensive datasets [111].

On the basis of these benefits, this study proposes a comparison of various DL models, namely (i) LSTM, as proposed by the MLPerf Tiny Working Group (of the MLCommons Association) [112], TCN [17], (iii) LMU [20], and (iv) a combination of attention mechanisms with CNNs, aiming at assessing their performance in terms of speech enhancement and computational complexity.

4.2 Literature Review

Focusing on key contributions and findings from prior research in the field of audio enhancement, the Tiny Recurrent U-Net [113] is specifically crafted for real-time audio enhancement applications. In detail, the model utilizes Per-Channel Energy Normalization (PCEN) [114] as input to process audio files. The network architecture is an encoder/decoder with a fusion of one-dimensional CNN (1D-CNN) and Gated Recurrent Unit (GRU) layers, chosen to balance computational complexity. Notably, the model is designed with a minimal parameter count of only 0.38 million, making it well-suited for deployment on embedded devices with limited computational resources.

In [115], the authors propose a specific model—namely, a fully Convolutional Denoising AutoEncoder (CDAE) [116] model—for audio enhancement of audio files, with a considered sample rate for audio files equal to 44.1 kHz, and the dataset being composed of 100 sound files. This approach provided better results in comparison to Feed Forward Networks (FFNs) using only a small number of parameters: with CDAE the number of parameters needed by the NN decreased to 4,169,499. This leads to reduced computational complexity and faster inference times, making the method more practical for real-time applications.

Finally, in [117] a lightweight audio denoising model for removing babble noise and enhancing audio quality is proposed. In detail, the proposed model consists of repeated units of convolution, batch normalization, and ReLU activation. The network consists of a total of 16 blocks, which corresponds to a total of 33,000 parameters, which makes it appropriate to execute on edge devices.

4.3 Dataset

4.3.1 Dataset Description

The dataset provided by MLCommons consists of 60,001 clean audio files, serving as reference audio, and of 60,001 noisy audio files, with all audio files being 30 s long, with a sample rate of 16,000 Hz in mono format. To this end, Figure 4.1 shows a comparative analysis of clean and noisy audio signals in both time (waveforms on the left) and frequency (spectrograms on the right) domains.

4.3.2 Dataset Pre-processing

In this stage, the Short-Time Fourier Transform (STFT) is computed for both the clean audio file and its corresponding noisy audio file. The STFT is evaluated using specified parameters, including a sampling rate of 16,000 Hz, a frame length of 256 samples (window size), and a Fast Fourier Transform (FFT) length of 512. In order to preserve the main characteristics of each audio's STFT, both magnitude information (real part) and phase information (imaginary part) are retained, with real and imaginary parts of the i-th component of the STFT ($i \in \{0,\ldots,256\}$) being then normalized (by considering frame length) as follows:

$$r_i = \frac{x_i}{\max(|x|)} \cdot \begin{cases} \frac{1}{\max(|x|)} & \text{if } \max(|x|) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.1}$$

where: $\max(|x|) \triangleq \max_{i=0,\ldots,256}(|x_i|)$; $r_i, i \in \{0,\ldots,256\}$ is the normalized value of the element x_i in the array x. There are 48,001 audio files set aside for training, 6,001 for validation, and 5,999 for testing in the MLCommons dataset. The pre-processed data comprises STFT representations of both clean and corresponding noisy audio signals. The provided dataset structure has been pre-processed specifically for supervised learning. In this context, the models are trained to remove noise from the input audio signal by utilizing the matching clean audio STFT representations.

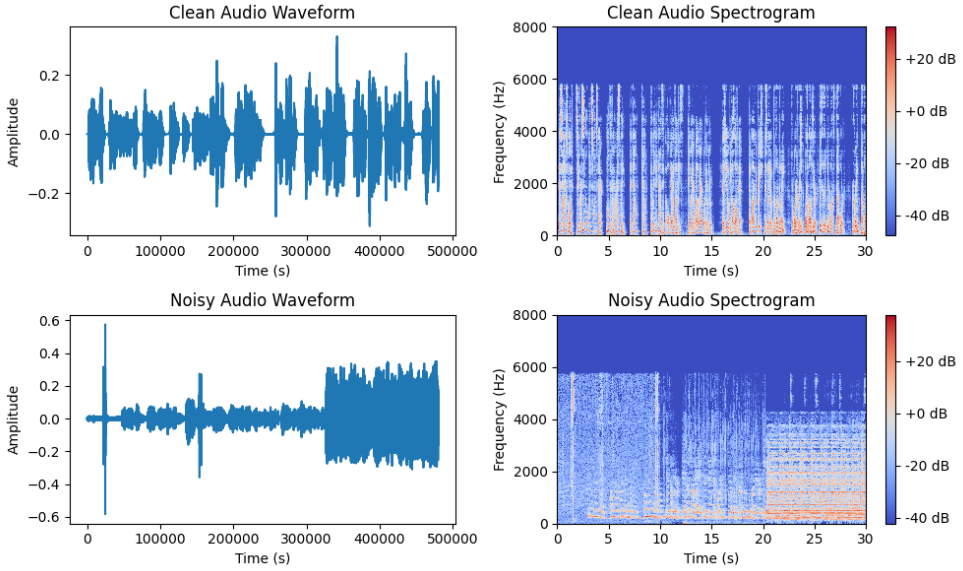

Figure 4.1: Waveforms (left) and spectrograms (right) of clean and noisy audio signals.

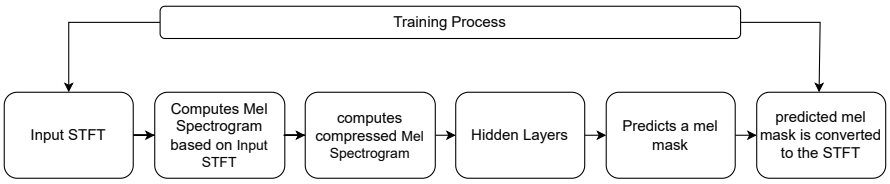

Figure 4.2: Training process.

4.4 Experimental Results

4.4.1 Training

Table 4.1: Architectural Configurations of Evaluated Speech Enhancement Models

Model	1st layer	2nd layer	3rd layer	4th layer	5th layer	6th layer
M_{LSTM}	LSTM layer size: 256 activation: tanh	LSTM layers size: 256 activation: tanh	Batch Normalization	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid	
$M_{\text{SepConv1D}}$	SeparableConv1D filters: 256 kernel size: 2	SeparableConv1D filters: 256 kernel size: 2	Batch Normalization	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid	
M_{TCN_1}	TCN filters: 256 dilations: [2,4,8,16,32]	TCN filters: 256 dilations: [2,4,8,16,32]	Batch Normalization	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid	
M_{TCN_2}	TCN filters: 256 dilations: 8	TCN filters: 256 dilations: 8	Batch Normalization	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid	
M_{Conv1D_1}	Conv1D filters: 256 kernel size: 2	Conv1D filters: 256 kernel size: 2	Batch Normalization	Attention layer	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid
M_{Conv1D_2}	Conv1D filters: 256 kernel size: 10	Conv1D filters: 256 kernel size: 10	Batch Normalization	Attention layer	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid
M_{Conv2D}	Conv2D filters: 256 kernel size: (2,2)	Conv2D filters: 256 kernel size: (2,2)	Batch Normalization	Attention layer	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid
M_{LMU}	LMU layer size: 256 activation: tanh	LMU layer size: 256 activation: tanh	Batch Normalization	Dense layer size: 128 activation: relu	Dense layer size: 128 activation: sigmoid	

The DL models were trained by converting the STFTs of the audio recordings into a compressed Mel spectrogram. The latter corresponds to a visual representation of the power spectrum of a signal, specifically in the Mel-frequency domain, and captures the perceptual characteristics of human hearing. This is computed using a fixed number of 128 MFCCs, with the minimum value for the frequency range of the Mel filter bank set at 40 Hz, the compression power of the Mel set at 0.3, and the learning rate for training the models set to 0.001. The training process of the audio enhancement model is shown in Figure 4.2.

Since the goal of this study is to investigate multiple DL models to evaluate their efficacy in enhancing audio quality and reducing noise, Table 4.1 provides a concise overview of the architecture of the chosen deep networks.

4.4.2 Evaluation

In order to evaluate the performance of the models, the Scale-Invariant Signal-to-Distortion Ratio (SISDR) [118] is selected as relevant performance metric, as it quantifies the quality of the predicted signal by comparing it with the target signal as follows:

$$a = \frac{\varepsilon + |\langle \text{reference}, \text{estimate} \rangle|}{\text{RSS} + \varepsilon} \tag{4.2}$$

$$e_{\text{true}} = a \cdot \text{reference} \tag{4.3}$$

$$e_{\text{res}} = \text{estimate} - \text{e}_{\text{true}} \tag{4.4}$$

$$S_{\text{ss}} = \sum |e_{\text{true}}|^2 \tag{4.5}$$

$$S_{\text{nn}} = \sum |e_{\text{res}}|^2 \tag{4.6}$$

$$\text{SISDR (dB)} = 10 \log_{10} \left(\frac{\varepsilon + S_{\text{ss}}}{\varepsilon + S_{\text{nn}}} \right) \tag{4.7}$$

where: a is the normalization factor; $\varepsilon = 1 \times 10^{-8}$; reference is the reference signal; estimate is the estimated signal; RSS is the power of the reference signal; e_{true} is the estimated true source signal; e_{res} is the residual signal. Spectral loss [119] is used as a loss function in the implemented DL models.

Table 4.2: Performance and efficiency comparison of speech enhancement models.

Model	Parameters [num]	FLASH [MB]	Avg. SISDR	Relative SISDR (compared to M_{LSTM})	n_{MACC} [num]	t_{exec} [s]
M_{LSTM}	969,984	3.7	**8.912**	-	1,018,368	0.0539
$M_{\text{SepConv1D}}$	150,016	0.57	**6.203**	2.709	688,128	0.0364
M_{TCN_1}	4,709,888	17.97	6.698	2.214	33,391,104	1.7707
M_{TCN_2}	772,608	2.95	7.361	1.551	4,555,264	0.2413
M_{Conv1D_1}	247,552	0.94	7.682	1.23	3,457,024	0.1830
M_{Conv1D_2}	1,033,984	3.94	7.255	1.657	38,109,184	2.0175
M_{Conv2D}	444,160	1.69	6.712	2.2	835,584	0.0442
M_{LMU}	642,048	2.45	8.613	0.299	133,056	0.0070

The experimental results are shown in Table. 4.2, where it can be seen that using LSTM (M_{LSTM}), as proposed by MLCommons Tiny, provides better SISDR than the other models. However, LMU (M_{LMU}) differs from it by only 3.41%, i.e., it is very close.

4.4.3 Computational Complexity Assessment

The computational complexity of the DL models has been assessed on the basis of their (i) estimated MACC operations, (ii) estimated execution time t_{exec}, and (iii) flash memory size, in order to ascertain the most appropriate model for deployment within tiny Internet of Things (IoT) MCU devices. More in detail, the estimated execution time for each implemented model has been calculated as follows:

$$t_{\text{exec}} = \frac{n_{\text{MACC}} \cdot 9}{f_{\text{board}}} \tag{4.8}$$

where: n_{MACC} represents the number of MACC operations; $f_{\text{board}} = 170$ MHz is the frequency of the reference board [120]; and the multiplying factor 9 is selected according to the average number of cycles per floating point MACC on ARM Cortex-M4, measured on STM32 MCUs and after some benchmarking activities. The results of evaluating the computational complexity of each model are shown in Table 4.2, where it can be seen how the adoption of LMU (M_{LMU}) achieves a reduction of almost 87% of MACC operations, about 33.8% of FLASH memory, and 87% of the inference time.

4.5 Conclusion

This study presents a comparison among eight DL models, showing that both LSTM and LMU demonstrate superior performance in enhancing speech signals. In particular, LSTM achieves the highest SISDR (3.41% higher than that with LMU), indicating its superiority in enhancing speech signals. However, LMU stands out by showcasing a substantial reduction of 40.69% in the number of parameters, at the benefit of the FLASH memory of the MCU. This reduction suggests enhanced computational efficiency and reduced memory requirements, making LMU an appealing *trade-off* for deployment scenarios targeting computational-constrained resources. The selection between LSTM and LMU should be carried out on the basis of a careful consideration of the *trade-offs* between SISDR performance, model complexity, and computational and energy efficiency, aligning with the specific requirements of the speech enhancement task.

Chapter 5

Accurate Classification of Sport Activities Under Tiny Deployability Constraints

5.1 Introduction

Nowadays, the computing devices used on a daily basis are embedding greater and greater intelligence and capabilities, as the society moves closer and closer to an era dominated by automation and AI. Significant developments in cloud computing and smart technologies over the past years have allowed for their integration into many facets of our daily lives [121]. A remarkable result of this integration is the ability to use computing power to gain a deeper understanding of human behavior and to make reliable predictions about it. Wearable sensor-based Human Activity Recognition (HAR) is an interesting research area, whose practical goal revolves around the recognition and classification of human activities by gathering data from sensors attached to the body. These sensors include accelerometers, gyroscopes, magnetometers, heart rate monitors, and, in general, any device capable of capturing different elements of human motion and physiological signals [122].

By providing better insights into a person's personality and psychological state,

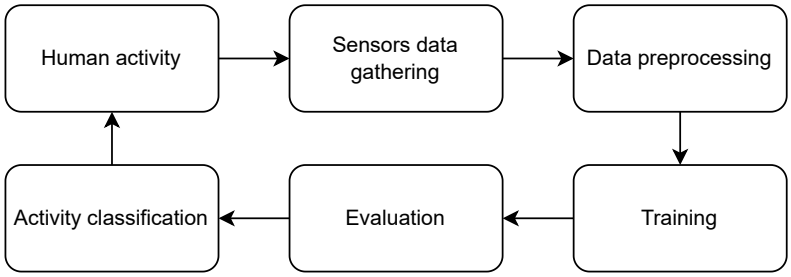

Figure 5.1: HAR general workflow.

HAR plays a significant role in interpersonal relationships and human interaction. In addition, the global community is concerned about the increasing number of elderly people in the present era: by 2050, the population aged 65 and older is expected to increase from 461 M to 2 B. As a result of this significant growth, notable implications will be felt in the social and healthcare sectors [123]. Therefore, this makes HAR an emerging and promising method for tracking old individuals' physical, functional, and cognitive well-being, with the goal of recognizing controlled and uncontrolled behaviours of a person [124]. Hence, the diversity and complexity of daily activities, as well the necessity to ensure the accuracy of the sensors and the produced data are a few of the difficulties involved with HAR [125,126]. A general workflow of an HAR procedure is shown in Figure 5.1.

Given the high interest in HAR-based analyses, the adoption of AI-based mechanisms to improve such behavioral classification is attractive. In particular, the adoption of DL, which is a sub-field of AI and an effective technique for making forecasts in a variety of fields (e.g., computer vision, NLP, finance, and healthcare) can represent a viable and useful solution. In particular, DL models (especially NNs) have the ability to learn intricate patterns and relationships in data, then making precise predictions [127].

HAR data correspond, in general, to time series, i.e., series of data points or observations gathered over time (e.g., on daily, monthly, or yearly bases). Besides HAR, time series are frequently exploited in heterogeneous disciplines, including finance, economics, weather forecasting, stock market analysis, and many others [128]. Thus,

HAR data can be considered as time series corresponding to sensor-based values collected over a period of time [129].

This study evaluates various DL algorithms to identify the most accurate one under constraints of tiny deployability, using a publicly available dataset consisting of 19 distinct daily activities. The experimental results show that a RNN given by the combination of a 1D-CNN with Bi-GRU performs better than LSTM, GRU, and the recently introduced LMU, returning an average accuracy within 0.2% of that of BiGRU (the RNN with highest accuracy) with an execution time more than 4 times shorter.

5.2 Literature Review

The performance of SVM and k-NN classifiers in human activity classification, considering their applications to two public datasets, is studied in [130]. In [131], the authors propose an activity recognition classifier to categorize 5 movements and 27 hand gestures by considering deep CNN and LSTM: the presented simulation results reveal a F1 score for the two classifiers equal to, respectively, 0.93 and 0.958. In [132], a BiLSTM architecture for the classification of human activities by using data collected from a smartphone is presented, showing that BiLSTM achieves the highest recognition accuracy (around 93%). In [133], the average accuracy is increased by a 0.93% by using a stacked LSTM for human activity recognition based on smartphone data. Finally, in [134] LSTM and CNN models are used to classify human activities, showing, on the basis of experimental results, that CNN is more accurate than LSTM.

5.3 Sport HAR Dataset

The reference dataset adopted here for human activity classification corresponds to a dataset comprising motion sensor data related to 19 daily sports activities, each performed by 4 subjects in their own daily life style for 5 min [1]. In detail, the dataset chosen for activity classification is composed of 500,000 rows, each one with

Table 5.1: Human activities contained in the sport HAR dataset [1].

Class	Activity
A_0	Sitting
A_1	Standing
A_2	Lying on back
A_3	Lying on right side
A_4	Ascending stairs
A_5	Descending stairs
A_6	Standing in an elevator still
A_7	Moving around in an elevator
A_8	Walking in a parking lot
A_9	Walking on a treadmill with a speed of 4 km/h in flat position
A_{10}	Walking on a treadmill with a speed of 4 km/h in 15 degree inclined position
A_{11}	Running on a treadmill with a speed of 8 km/h
A_{12}	Exercising on a stepper
A_{13}	Exercising on a cross trainer
A_{14}	Cycling on an exercise bike in horizontal position
A_{15}	Cycling on an exercise bike in vertical position
A_{16}	Rowing
A_{17}	Jumping
A_{18}	Playing basketball

45 columns that have been considered as *features*. The sensor units exploited for data collection are a 25 Hz sampling rate and are placed on torso, right arm, left arm, right leg, and left leg of each subject.

For the sake of completeness, Table 5.1 lists the different activities (identified with a short notation for readability) contained in the dataset. Illustrative representations of data related to activities A_0 and A_1 (performed by two different subjects) are shown in Figure 5.2 and Figure 5.3, respectively.

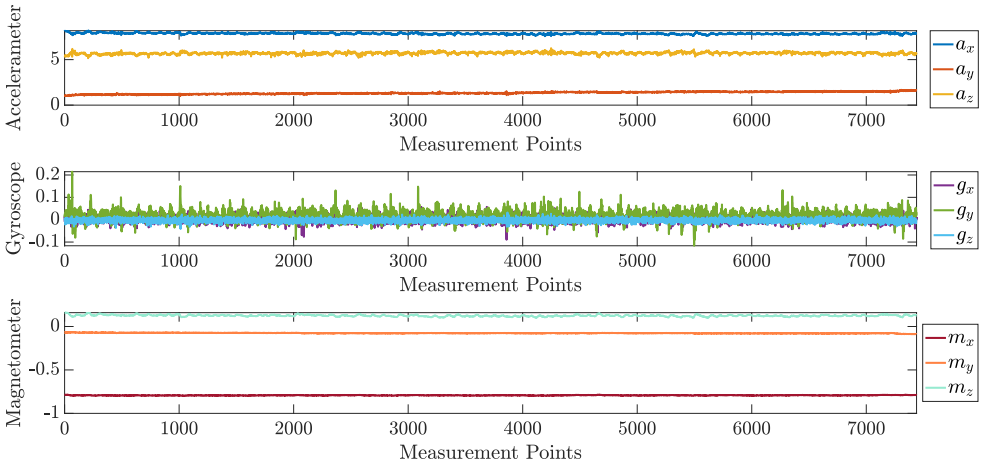

Figure 5.2: Accelerometer, gyroscope and magnetometer data related to activity A_0 carried out by a person.

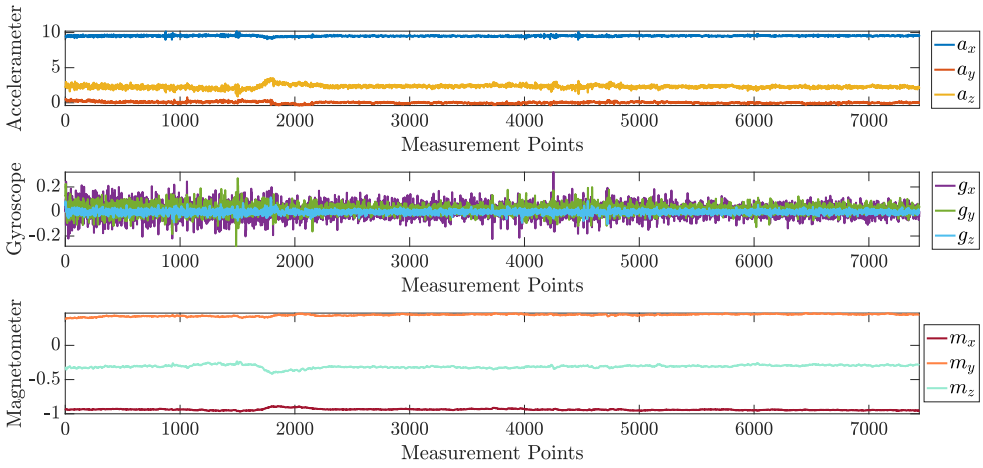

Figure 5.3: Accelerometer, gyroscope and magnetometer data related to activity A_1 carried out by a person.

5.4 Proposed Model

5.4.1 Considered DL Models

DL models, including LSTM, GRU, BiGRU and LMU, have been used to classify and recognize activities inside the dataset introduced in Section 5.3. The same architecture is considered for all models in this study, including: (i) three layers with 100, 50, and 30 neurons, respectively; (ii) the ReLU function [81] (defined as $x^+ \triangleq \max(0,x)$, where x is the input of a neuron) as activator function; (iii) the output layer containing 19 neurons (corresponding to the number of activities available in the dataset) for activity detection; and (iv) the softmax activation function [135] in the output layer.

In detail, the softmax activation, mostly used in multi-class classification problems, takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers, as follows:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = \{1,2,\ldots,K\} \tag{5.1}$$

where: z_i corresponds to the elements of the input vector $\mathbf{z} = (z_1,\ldots,z_K) \in \mathbb{R}^K$; K is the number of classes; and e^{z_i} is the standard exponential function to be applied to each element of the input vector.

The models have been trained for 50 epochs (corresponding to the number of times the models should have a full iteration over samples) on the 70% ($n_{\text{train}} =$ 350,000 samples) of the dataset, while the remaining 30% ($n_{\text{test}} =$ 150,000 samples) has been used to evaluate the DL methods' performance. Thus, considering a 2-second time window at a 25 Hz sampling rate, the corresponding sliding window (lag) equals to 50 samples, and consequently both the training and test sets have been transformed into sequences of observations, with each sequence containing 50 samples.

In order to re-scale the data with a zero mean and a standard deviation equal to 1, the standard scaler has been used [136].

Furthermore, two additional models are evaluated, corresponding to a combi-

nation of 1D-CNN with BiGRU and GRU, respectively. Specifically, the 1D-CNN-BiGRU model has the following structure:

- a 1D-CNN layer with ReLu activation function, with filter size equal to 64 and kernel size equal to 3 [137];
- a 25 neuron BiGRU layer with ReLu activation function;
- a 30 neuron fully-connected layer with ReLu activation function;
- a 19 neuron output layer with softmax activation function.

Instead, 1D-CNN-GRU has the following structure:

- a 1D-CNN layer with ReLu activation function, filter size equal to 64 and kernel size equal to 3;
- a 50 neuron GRU layer with ReLu activation function;
- a 30 neuron fully-connected layer with ReLu activation function;
- a 19 neuron output layer with softmax activation function.

5.4.2 Performance Metrics

The performance of the considered algorithms are evaluated on the basis of the following metrics.

- *Accuracy*, defined as the ratio between the total number of correct decisions and the total number of decisions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

where: TP is the number of true positives; TN is the number of true negatives; FP is the number of false positives; and FN is the number of false negatives.

- *Precision*, defined as the ratio between correctly predicted positive observations and the number of all predicted positive observations:

$$Precision = \frac{TP}{TP+FP}. \tag{5.3}$$

- *Recall*, defined as the ratio between the number of correctly identified positive observations and the total number of actual positive observations:

$$Recall = \frac{TP}{TP+FN}. \tag{5.4}$$

- *F1 score*, corresponding to the harmonic mean of precision and recall:

$$\begin{aligned} F1 &= \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \\ &= \frac{2 \cdot TP}{2 \cdot TP + FP + FN}. \end{aligned} \tag{5.5}$$

5.4.3 k-Fold Cross Validation

The performance and classification accuracy of the discussed DL methods are evaluated using the k-fold cross-validation method. The results of the validation with $k = 10$ (thus splitting the dataset into 10 equal parts) are shown in Table 5.2, where it can be observed that BiGRU attained (on average) the highest classification accuracy (approximately 99%) across the 10 folds of the cross-validation.

5.4.4 Computational Complexity Assessment

The complexity of the involved DL models is evaluated in terms of execution time and RAM requirements to determine the most suitable model for implementation in resource-constrained IoT devices. In detail, the STM32Cube [8] tool on the NUCLEO-G474RE [120] is used for complexity evaluation. For the sake of completeness, it should be noted that, due to the novelty of LMU, it was not possible to implement LMU in the aforementioned NUCLEO-G474RE IoT node: therefore, the complexity of this method could not be evaluated in terms of weights and RAM parameters.

Table 5.2: k-fold cross validation accuracy of the considered DL methods.

	BiGRU	1D-CNN-BiGRU	1D-CNN-GRU	LSTM	GRU	LMU
Fold 1	99.0%	98.5%	96.1%	98.9%	98.5%	97.3%
Fold 2	98.9%	98.6%	96.1%	98.8%	98.5%	96.9%
Fold 3	99.1%	98.3%	97.0%	98.7%	98.6%	97.2%
Fold 4	99.0%	98.4%	96.8%	98.6%	98.5%	97.1%
Fold 5	99.0%	98.2%	94.9%	98.7%	98.4%	97.3%
Fold 6	98.0%	98.6%	97.4%	98.7%	98.5%	97.3%
Fold 7	99.9%	98.2%	96.7%	98.7%	98.5%	97.2%
Fold 8	99.0%	98.3%	97.0%	98.7%	98.5%	97.2%
Fold 9	98.0%	98.3%	96.7%	98.7%	98.6%	97.2%
Fold 10	98.9%	98.3%	96.8%	98.8%	98.6	97.1%
Accuracy (avg)	**99.0**%	98.4%	96.6%	98.7%	98.5%	96.6%
Std Dev	±0.042	±0.140	±0.651	±0.075	±0.084	±0.110

Table 5.3: Accuracy and computational complexity of the considered DL methods.

Model	Accuracy	n_{MACC} [num]	Weights [Byte]	RAM [Byte]	t_{exec} [s]	Parameters P [num]	n_{train}/P
BiGRU	**99.1%**	4,382,534	401,476	12,200	0.232	100,369	**3.48**
1D-CNN-BiGRU	98.9%	**1,065,430**	**96,652**	13,088	**0.056**	**24,163**	14.48
1D-CNN-GRU	98.9%	1,245,890	112,892	13,688	0.065	28,223	12.4
LSTM	98.6%	2,932,534	263,476	12,200	0.155	65,569	5.33
GRU	97.7%	2,192,534	205,076	**11,800**	0.116	51,269	6.82
LMU	97.4%	2,807,070	240,076	–	0.148	60,019	5.83

The estimated execution time of each model implemented on NUCLEO-G474RE has been calculated as follows:

$$t_{\text{exec}} = \frac{n_{\text{MACC}} \cdot 9}{f_{\text{board}}} \tag{5.6}$$

where: n_{MACC} corresponds to the number of MACCs; $f_{\text{board}} = 170$ MHz is the frequency of the NUCLEO-G474RE board; and the number of MACCs is multiplied by 9 as 9 cycles per MACC are needed.

In order to also jointly investigate the computational complexity and the accuracy, it is of interest to consider a performance metric which jointly captures these two metrics. To this end, the following are considered:

- the normalized accuracy, defined as $Accuracy/100 \in (0,1)$, where *Accuracy* has been defined in Eq. (5.2);
- various computational complexity metrics: weights, total RAM, MACCs, and

execution time.

Denoting as c one of the selected computational complexity metrics, the joint complexity-accuracy performance metric, defined as ξ, can be expressed as follows:

$$\xi^{(c)} = \frac{Accuracy}{100}\, c\,. \tag{5.7}$$

5.5 Results and Discussion

The performance and accuracy of the DL models detailed in Section 5.4 (namely: LSTM, GRU, LMU, BiGRU, 1D-CNN-BiGRU, 1D-CNN-GRU) are evaluated for human activity classification. The performance metrics discussed in Subsection 5.4.2 (namely: precision, accuracy, recall, and F1 score) are considered and support values of each class (namely, the number of occurrences of each specific class in the dataset), together with the computational complexity discussed in Subsection 5.4.4. The obtained results are shown in Table 5.3 and Table 5.4, respectively. On the basis of these performance results, BiGRU is the most accurate DL algorithm, returning a 99.1% classification accuracy. This is due to the fact that it can exploit input data in both forward and backward directions simultaneously, increasing, in turn, the learning rate of the model and, thus, the prediction accuracy. However, when taking into account jointly the accuracy and the computational complexity, the DL algorithm to be preferred is 1D-CNN-BiGRU, as it guarantees a slight accuracy degradation (0.2% in Table 5.3) with a remarkable complexity reduction (in terms of n_{MACC}, weights, t_{exec}, parameters P, and ratio n_{train}/P in Table 5.3). The joint complexity-accuracy performance in Table 5.4 clearly shows the superiority of 1D-CNN-BiGRU. Consequently, with respect to other analyzed models, 1D-CNN-BiGRU can be deemed as the most appropriate under tiny deployability constraints when considering prediction accuracy and complexity (in terms of n_{MACC}).

For the sake of completeness, the confusion matrices (highlighting the number of correct and incorrect classifications) returned by LSTM, GRU, LMU, BiGRU, 1D-CNN-BiGRU, and 1D-CNN-GRU methods for each activity class contained in the reference dataset (summarized in Table 5.1) are shown in Figure 5.4, Figure 5.5,

Table 5.4: Joint complexity-accuracy of the considered DL methods.

Model	$\xi^{(n_{\text{MACC}})}$	$\xi^{(\text{Weights})}$	$\xi^{(\text{RAM})}$	$\xi^{(t_{\text{exec}})}$
BiGRU	4,343,091	397,863	12,090	0.229
1D-CNN-BiGRU	**1,053,710**	**95,589**	12,944	**0.055**
1D-CNN-GRU	1,232,185	111,650	13,537	0.065
LSTM	2,891,479	259,787	12,033	0.153
GRU	2,142,106	200,359	**11,529**	0.113
LMU	2,734,086	233,834	–	0.144

Figure 5.6, Figure 5.7, Figure 5.8, and Figure 5.9, respectively, while performance results of each DL algorithm, in terms of all considered metrics, are shown in Table 5.5, Table 5.6, Table 5.7, Table 5.8, Table 5.9, and Table 5.10, respectively.

5.6 Conclusions

This study presents an experimental comparison of traditional DL models (LSTM, GRU, LMU, BiGRU) with two novel DL models, which combine 1D-CNN with BiGRU and GRU, respectively, for human activity classification using a publicly available dataset. The obtained experimental results show that the BiGRU model outperforms the other DL methods, showing a 99.1% accuracy in the classification of the test data. Overall, according to the achieved results, 1D-CNN-BiGRU provides the lowest complexity in regard to the execution time. Then, by considering a 98.9% accuracy, 1D-CNN-BiGRU (even in comparison with 1D-CNN-GRU) could be reasonably considered for implementation on tiny IoT devices (as well as the other discussed DL models) in the future, in order to be able to classify and predict human activities directly inside wearable devices worn by subjects—e.g., forecasting the behavior of patients and elderly people, thus allowing to daily track and monitor their progresses.

For the sake of completeness, the pseudo-code representation of the 1D-CNN-BiGRU model proposed in Section 5.4 is shown in Algorithm 1.

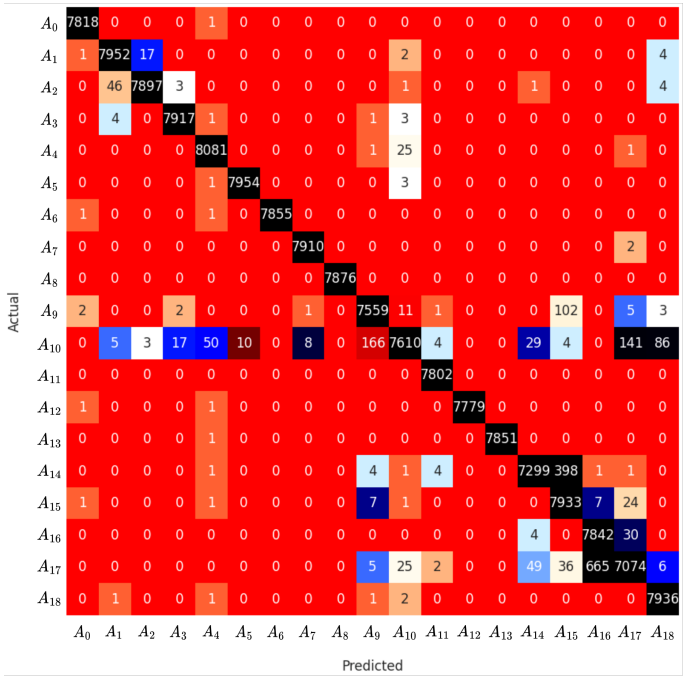

Figure 5.4: Confusion matrix returned by the LSTM model for the classification of the evaluated activity classes.

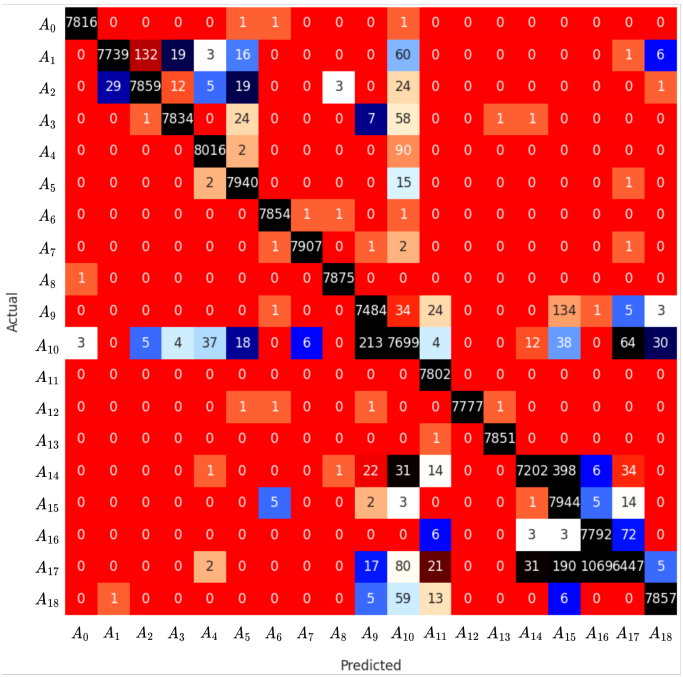

Figure 5.5: Confusion matrix returned by the GRU model for the classification of the evaluated activity classes.

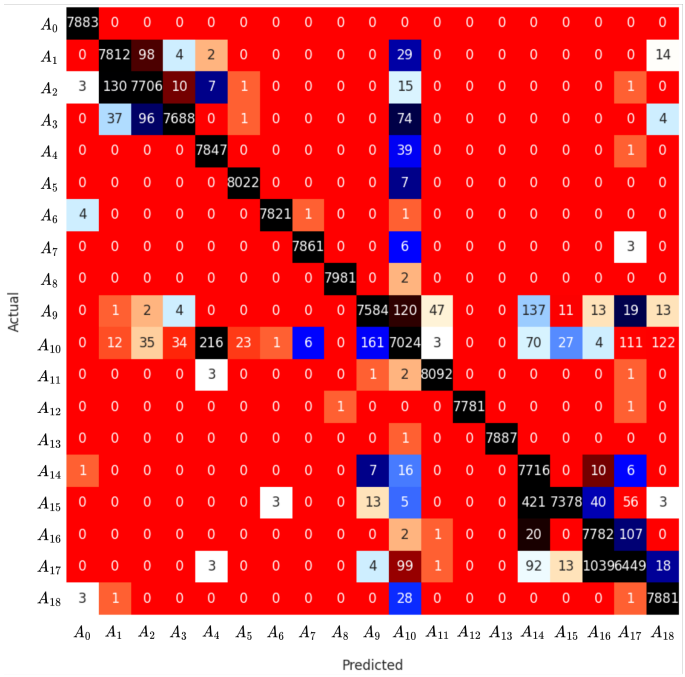

Figure 5.6: Confusion matrix returned by the LMU model for the classification of the evaluated activity classes.

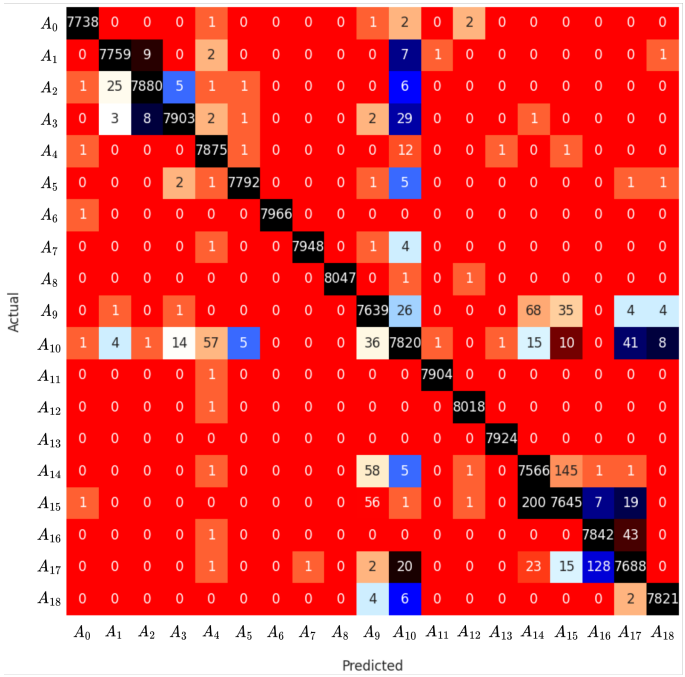

Figure 5.7: Confusion matrix returned by the BiGRU model for the classification of the evaluated activity classes.

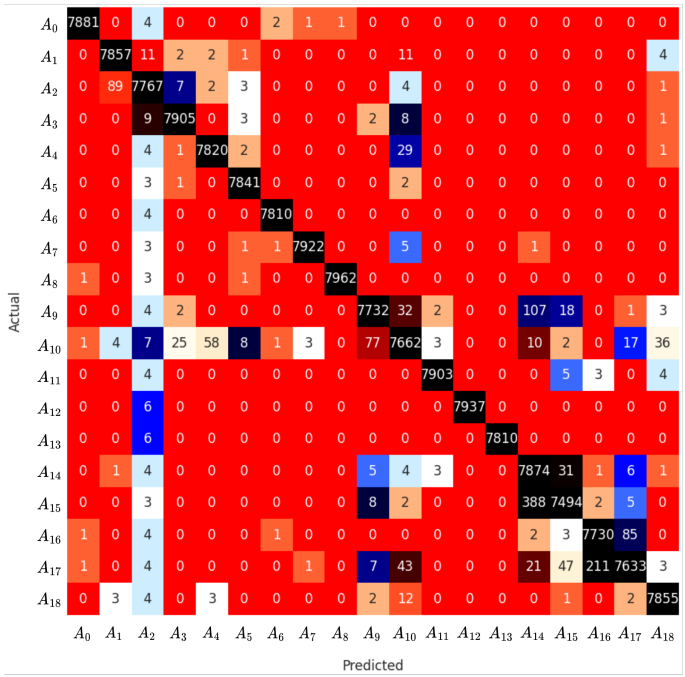

Figure 5.8: Confusion matrix returned by the 1D-CNN-BiGRU model for the classification of the evaluated activity classes.

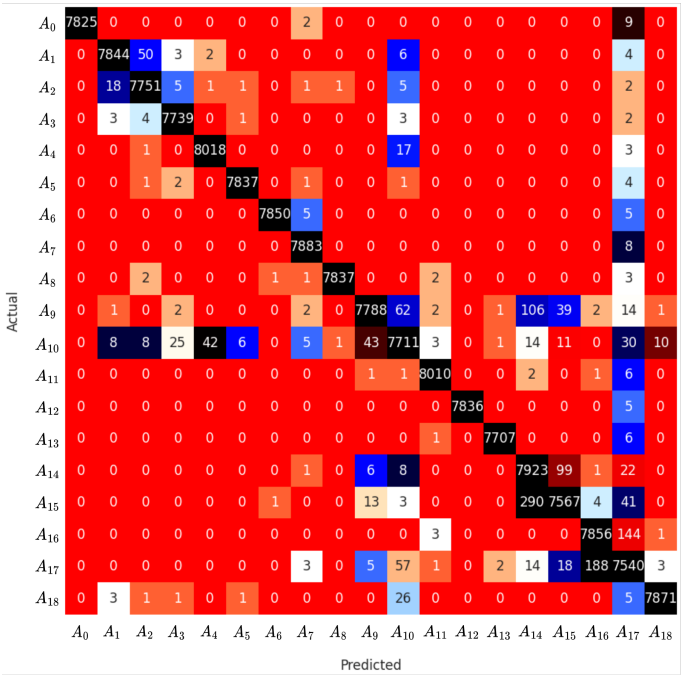

Figure 5.9: Confusion matrix returned by the 1D-CNN-GRU model for the classification of the evaluated activity classes.

Table 5.5: LSTM classification report for all activity classes, in terms of precision, recall, F1-score, and support.

Class	*Precision*	*Recall*	*F1*	Support
A_0	0.999233	0.999872	0.999553	7819
A_1	0.993007	0.996991	0.994995	7976
A_2	0.997474	0.993084	0.995274	7952
A_3	0.997229	0.998864	0.998046	7926
A_4	0.992752	0.99667	0.994707	8108
A_5	0.998744	0.999497	0.999121	7958
A_6	1	0.999745	0.999873	7857
A_7	0.998863	0.999747	0.999305	7912
A_8	1	1	1	7876
A_9	0.976111	0.983476	0.97978	7686
A_{10}	0.99037	0.935694	0.962256	8133
A_{11}	0.998592	1	0.999296	7802
A_{12}	1	0.999743	0.999871	7781
A_{13}	1	0.999873	0.999936	7852
A_{14}	0.988756	0.946815	0.967332	7709
A_{15}	0.936268	0.994858	0.964674	7974
A_{16}	0.920963	0.995683	0.956867	7876
A_{17}	0.97197	0.899771	0.934478	7862
A_{18}	0.987187	0.99937	0.993242	7941
Accuracy	0.9863	0.9863	0.9863	0.9863
Macro avg	0.986712	0.986303	0.986242	150,000
Weighted avg	0.986707	0.9863	0.986238	150,000

Table 5.6: GRU classification report for all activity classes, in terms of precision, recall, F1-score, and support.

Class	*Precision*	*Recall*	*F1*	Support
A_0	0.999488	0.999616	0.999552	7819
A_1	0.996138	0.970286	0.983042	7976
A_2	0.982744	0.988305	0.985516	7952
A_3	0.995552	0.988393	0.991959	7926
A_4	0.993801	0.988653	0.99122	8108
A_5	0.989902	0.997738	0.993804	7958
A_6	0.998855	0.999618	0.999237	7857
A_7	0.999115	0.999368	0.999242	7912
A_8	0.999365	0.999873	0.999619	7876
A_9	0.965428	0.973718	0.969556	7686
A_{10}	0.943852	0.946637	0.945242	8133
A_{11}	0.989474	1	0.994709	7802
A_{12}	1	0.999486	0.999743	7781
A_{13}	0.999745	0.999873	0.999809	7852
A_{14}	0.993379	0.934233	0.962899	7709
A_{15}	0.911741	0.996238	0.952118	7974
A_{16}	0.87817	0.989335	0.930444	7876
A_{17}	0.97108	0.82002	0.88918	7862
A_{18}	0.994305	0.989422	0.991858	7941
Accuracy	0.977967	0.977967	0.977967	0.977967
Macro avg	0.97906	0.977937	0.977829	150,000
Weighted avg	0.978984	0.977967	0.977808	150,000

Table 5.7: LMU classification report for all activity classes, in terms of precision, recall, F1-score, and support.

Class	*Precision*	*Recall*	*F1*	Support
A_0	0.998607	1	0.999303	7883
A_1	0.977355	0.98153	0.979438	7959
A_2	0.970896	0.978788	0.974826	7873
A_3	0.993282	0.973165	0.98312	7900
A_4	0.971404	0.994928	0.983025	7887
A_5	0.996893	0.999128	0.998009	8029
A_6	0.999489	0.999233	0.999361	7827
A_7	0.99911	0.998856	0.998983	7870
A_8	0.999875	0.999749	0.999812	7983
A_9	0.976062	0.953842	0.964824	7951
A_{10}	0.940295	0.894891	0.917031	7849
A_{11}	0.993615	0.999136	0.996368	8099
A_{12}	1	0.999743	0.999871	7783
A_{13}	1	0.999873	0.999937	7888
A_{14}	0.912488	0.994843	0.951887	7756
A_{15}	0.993135	0.931683	0.961428	7919
A_{16}	0.875563	0.983569	0.926429	7912
A_{17}	0.954559	0.835579	0.891115	7718
A_{18}	0.978399	0.99583	0.987037	7914
Accuracy	0.974633	0.974633	0.974633	0.974633
Macro avg	0.975317	0.97444	0.974306	150,000
Weighted avg	0.975427	0.974633	0.974463	150,000

Table 5.8: BiGRU model classification report for all activity classes, in terms of precision, recall, F1-score, and support.

Class	*Precision*	*Recall*	*F1*	Support
A_0	0.999354	0.999225	0.99929	7744
A_1	0.995765	0.997429	0.996596	7779
A_2	0.997721	0.995075	0.996396	7919
A_3	0.997224	0.994213	0.995716	7949
A_4	0.991189	0.997972	0.994569	7891
A_5	0.998974	0.99859	0.998782	7803
A_6	1	0.999874	0.999937	7967
A_7	0.999874	0.999246	0.99956	7954
A_8	1	0.999752	0.999876	8049
A_9	0.979359	0.982129	0.980742	7778
A_{10}	0.984391	0.975792	0.980073	8014
A_{11}	0.999747	0.999873	0.99981	7905
A_{12}	0.999377	0.999875	0.999626	8019
A_{13}	0.999748	1	0.999874	7924
A_{14}	0.961006	0.972744	0.966839	7778
A_{15}	0.973761	0.964061	0.968887	7930
A_{16}	0.982953	0.99442	0.988654	7886
A_{17}	0.985767	0.975882	0.9808	7878
A_{18}	0.998213	0.998468	0.998341	7833
Accuracy	0.991833	0.991833	0.991833	0.991833
Macro avg	0.991812	0.991822	0.991809	150,000
Weighted avg	0.991846	0.991833	0.991831	150,000

Table 5.9: 1D-CNN-BiGRU model classification report for all activity classes, in terms of precision, recall, F1-score, and support.

Class	*Precision*	*Recall*	*F1*	Support
A_0	0.999493	0.998986	0.999239	7889
A_1	0.987805	0.99607	0.99192	7888
A_2	0.988923	0.986536	0.987728	7873
A_3	0.995216	0.997099	0.996157	7928
A_4	0.991756	0.995291	0.993521	7857
A_5	0.997583	0.999235	0.998408	7847
A_6	0.99936	0.999488	0.999424	7814
A_7	0.999369	0.998613	0.998991	7933
A_8	0.999874	0.999372	0.999623	7967
A_9	0.987106	0.97861	0.98284	7901
A_{10}	0.980548	0.968158	0.974313	7914
A_{11}	0.998989	0.99798	0.998484	7919
A_{12}	1	0.999245	0.999622	7943
A_{13}	1	0.999232	0.999616	7816
A_{14}	0.937046	0.992938	0.964183	7930
A_{15}	0.985923	0.948368	0.966781	7902
A_{16}	0.972694	0.987733	0.980156	7826
A_{17}	0.98503	0.957596	0.97112	7971
A_{18}	0.993172	0.996574	0.99487	7882
Accuracy	0.9893	0.9893	0.9893	0.9893
Macro avg	0.989468	0.989322	0.989316	150,000
Weighted avg	0.989459	0.9893	0.9893	150,000

Table 5.10: 1D-CNN-GRU model classification report for all activity classes, in terms of precision, recall, F1-score, and support.

Class	*Precision*	*Recall*	*F1*	Support
A_0	1	0.998596	0.999298	7836
A_1	0.995811	0.991782	0.993792	7909
A_2	0.99143	0.995633	0.993527	7785
A_3	0.995114	0.998323	0.996716	7752
A_4	0.994419	0.997388	0.995901	8039
A_5	0.998853	0.998853	0.998853	7846
A_6	0.999745	0.998728	0.999236	7860
A_7	0.997343	0.998986	0.998164	7891
A_8	0.999745	0.998853	0.999299	7846
A_9	0.991344	0.971072	0.981104	8020
A_{10}	0.976076	0.973857	0.974965	7918
A_{11}	0.998504	0.998629	0.998566	8021
A_{12}	1	0.999362	0.999681	7841
A_{13}	0.999481	0.999093	0.999287	7714
A_{14}	0.948976	0.983002	0.96569	8060
A_{15}	0.978407	0.95555	0.966843	7919
A_{16}	0.975658	0.981509	0.978575	8004
A_{17}	0.960143	0.96284	0.961489	7831
A_{18}	0.998098	0.995321	0.996708	7908
Accuracy	0.989287	0.989287	0.989287	0.989287
Macro avg	0.989429	0.989336	0.989352	150,000
Weighted avg	0.989362	0.989287	0.989294	150,000

Algorithm 1 1D-CNN-BiGRU Model

```
▷ Define hyperparameters
n_timesteps = 50
n_features = 45
n_outputs = 19

▷ Create a Sequential model
m = Sequential()

▷ Add a 1D Convolutional layer
m.add(Conv1D(filters = 64,
             kernel_size = 3,
             activation = "relu",
             input_shape = (n_timesteps,n_features)))

▷ Add a Bidirectional GRU layer
m.add(Bidirectional(GRU(units = 25,
                        activation = "relu",
                        return_sequences = False)))

▷ Add a Dense layer with ReLU activation
m.add(Dense(units = 30, activation = "relu"))

▷ Add the output Dense layer with softmax activation
m.add(Dense(units = n_outputs, activation = "softmax"))

▷ Compile the model with categorical cross-entropy loss and ADAM optimizer
m.compile(loss = "categorical_crossentropy",
          optimizer = "adam",
          metrics = ["accuracy"])
```

Chapter 6

Deep Learning Algorithms for Cryptocurrency Price Prediction: a Comparative Analysis

6.1 Introduction

Unlike an historical model of economy where traditional currencies (managed by governments) are dominant, new ways to exchange and transfer money have appeared, especially owing to the cryptocurrencies. In general, cryptocurrencies can be seen as decentralized encrypted digital coins designed to be traded without an intermediate monetary authority (e.g., a central bank) and used on the Internet [138]. Thanks to their decentralized, secure, and self-managed nature, nearly 9,000 cryptocurrencies have be created until 2023, eventually gaining a significant attention from a wide plethora of investors and having a strong impact on the global financial market. As an example, the cryptocurrencies' market value overcame $1.48 trillion in December 2023 (with its highest market cap, about $2.82 trillion, in November 2021) [139]. Among the existing cryptocurrencies, Bitcoin (BTC) is so far one of the most popular and represents, as a Peer-to-Peer (P2P) electronic currency exploiting core cryptographic mechanisms, the first cryptocurrency (introduced in 2008 [140]).

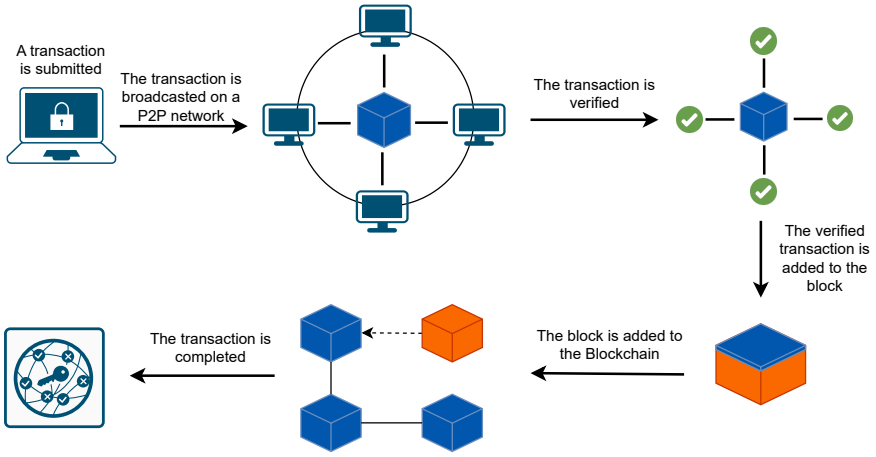

Figure 6.1: High-level representation of a generic blockchain-based transaction flow.

Another technology strictly related to several cryptocurrencies (and required in order to govern them) is the Distributed Ledger Technology (DLT) [141], based on the well-known paradigm embodied by blockchain. In fact, blockchain was originally designed to be exploited as an efficient (and trustable) way to record financial transactions. As shown in Figure 6.1, blockchain exploits cryptographic hashing techniques and a distributed network to force the persistence of all digital transactions as immutable, in the end ensuring (i) reliable transactions tracing, (ii) security maximization, and (iii) removal of reliance on third-party intermediaries [142].

More in detail, the benefits of blockchain and DLT—which transform data management from centralized to decentralized—support a wide range of applications, including banking, agriculture, intelligent transportation, energy, and supply chain management [143]. As an example, in [144] the authors introduce a blockchain-based framework designed for managing patients' medical records and the corresponding medical supply chain, in particular exploiting an Hyperledger Fabric network to ensure node confidentiality. Similarly, a blockchain-based Web application for generating medical certificates is proposed in [145], allowing users to manage their healthcare information through IoT devices, and operating on an Ethereum (ETH)

blockchain to prevent fraud in medical records. To this end, patients are provided credentials by a hospital authority and their data are verified on the blockchain before medical services are provided. Focusing on Intelligent Transportation Systems (ITSs), in [146] an architecture securing data sharing within ITSs, using a combination of edge computing and DLT, is presented. This system, built on an ETH network and processing data closer to their sources, allows to reduce the volume of data to be sent to a central cloud.

As mentioned before, DLTs offer significant benefits, including high transparency, reduced supply chain costs, and enhanced safety, by tracking products from the origin to the customers [147]. As an example, in [148,149] the authors provide an effective strategy for updating livestock agriculture by combining IoT and blockchain technologies. In detail, a cloud-based Smart Livestock Farming (SLF)-oriented management system, composed by IoT wearable sensors to track livestock health in real time, is proposed, together with a blockchain-powered platform (i) providing secure and transparent electronic record-keeping and (ii) removing the necessity for middlemen, and (iii) boosting trust between the involved parties.

As can be understood from the variability in the aforementioned market cap, cryptocurrency-based financial markets follow arbitrary and uncertain trends: this can only attract and make investors (of every nature) eager to find models to detect and predict financial market's trends. On a more technical side, this is due to the fact that the value of a cryptocurrency heavily depends on several factors, identifiable as both *internal* and *external*: demand and supply represent the most important *internal factors* affecting the price of cryptocurrencies (and, in general, of every kind of standard currency); political decisions and the degree of attractiveness of the cryptocurrency market, as well as the states of other markets (such as gold, regular stock, oil, etc.), are in general envisioned as *external factors* [150]. In the end, this highlights—one more time—how, on the basis of the price fluctuations in the cryptocurrency market, accurate price prediction is crucial to reduce the investment risk.

From a more statistical point of view, predicting the price of a cryptocurrency can be considered as a time series prediction, with forecasts being based on historical events over a proper time interval. Thus, given that time series prediction is cur-

rently used in a variety of applications in several fields—e.g., signal processing [151], weather forecasting [152], disease prediction [153], computer networks [154], and business planning [155]—the adoption of DL algorithms for prediction purposes can be considered a natural fit for time series. This is further motivated by the fact that, in recent years, DL algorithms have been widely used to extract features from financial time series and make predictions, since they (i) can be trained and operate with large amounts of multidimensional data and (ii) can learn non-linear dependencies between variables [156]. Indeed, financial time series are well-known as being non-linear, non-stationary, and depending on multiple variables, in the end featuring intricate trends that should be taken into account by an accurate (and complicated) forecasting model. This is what makes forecasting financial time series challenging.

Besides DL, traditional and ML algorithms are also commonly used to forecast time series. More in detail, *traditional approaches* (e.g., Automatic Regression Integrated Moving Average (ARIMA) and Auto-Regressive Moving Average (ARMA) [157]) use statistical methods to obtain patterns in time series, but they are affected by some limitations. As an example, considering a linear relationship between the variables makes them ineffective in predicting and analyzing time series with non-linear data, in the end reducing the accuracy of long-term forecasts with low accuracy and making them unsuitable for multidimensional big data [158].

In this study, a comparative performance analysis and evaluation of different DL algorithms in predicting the price of three well-known cryptocurrencies (namely: BTC; ETH; and Ripple (XRP)) is proposed. In detail, several "single" DL algorithms (namely: MLP, SimpleRNN, LSTM, BiLSTM, GRU, BiGRU, CNN, Transformer) as well as concatenated DL mechanisms (namely: CNN-RNN and CNN-GRU) are considered. Their prediction accuracy is investigated in terms of RMSE, MAE, MAPE, and R^2. The performance is analyzed considered various values of time lag, corresponding to the number of past observations considered to predict the price at the next time epoch. The Adaptive Moment Estimation (Adam) optimizer [159] is applied to improve the estimation accuracy of the considered DL methods for both short- and long-term cryptocurrencies' prices predictions.

6.2 Literature Review

Looking at the adoption of AI-based techniques in the context of cryptocurrencies, in [160] different AI mechanisms are used to improve the privacy and the effectiveness of the Physical Unclonable Function (PUF) electronic cash system. In detail, the authors in [160] focus on this PUF system due to the fact that it makes use of physical unduplicatable functions in order to facilitate authentication and encryption procedures in the case different entities and multiple trusted third-parties should be involved. Similarly, in [161] the authors consider various ANNs-based mechanisms aiming at the next day's BTC price prediction, training and validating their approach through the use of historical daily data (related to the time interval 2013-2017): a Back Propagation Neural Network (BPNN) is shown to return the best prediction performance. Furthermore, in [162] LSTM is evaluated and compared with different ARIMA-based hybrid models, showing that LSTM outperforms, in terms of the BTC price's prediction accuracy, the other methods. A similar evaluation (with similar outcomes) is proposed in [163], in which different Generalized Regression Neural Network (GRNN) models are compared with LSTM aiming at predicting the price of BTC, Digital Cash, and XRP, on the basis of a dataset related to the time interval July 2010-October 2018.

In [164], a BiLSTM is used on a BTC daily price dataset (containing data referring to the time interval January 2012-September 2020) in order to predict BTC prices: the final results show that the BiLSTM model provides more accurate predictions than linear regression-based models. In a similar way, the authors in [165] use a daily price dataset (referring to the time interval January 2014-October 2017) to evaluate the impact of the number of LSTM features on the BTC price prediction, considering single and multiple features. In the end, by taking into account multiple features, LSTM is shown to provide more accurate predictions at a significantly reduced error rate.

In [166], LSTM, GRU and BiLSTM models are adopted to predict the prices of BTC, ETH, and Litecoin, on the basis of their market capitalization, obtaining that BiLSTM returns the most precise forecasts for all the considered cryptocurrencies—

in detail, with reference to RMSE and MAPE, considering data in the time interval January 2021–January 2023. Similarly, a Multi-Layer GRU model forecasting the prices of BTC, ETH, and Dogecoin, is introduced in [167]. In detail, this model consists of three hidden layers (each containing 200, 100, and 50 neurons, respectively): according to the presented results, the Multi-Layer GRU returns (on average) an RMSE 23 times lower than that of the LSTM model and nearly 4 times lower than that of the GRU model.

A comparison on the efficiencies of LSTM and ARIMA for short-term BTC price prediction (using data in the time interval December 2020–December 2021) is investigated in [168], obtaining that LSTM significantly enhances RMSE and MAE by 83.84% and 84.84%, respectively. Instead, encoder-decoder (also denoted as AEs) predictive models for financial markets price prediction are proposed in [169]. In detail, the performance of AE-GRU and AE-LSTM is analyzed in forecasting stock prices, stock indexes, and cryptocurrencies—in particular, BTC pricing stretching in the time interval 2014–2022. Based on the achieved results, AE-GRU outperforms AE-LSTM in lowering MAPE by 50%.

The use of RNNs and MLP for predicting short- and long-terms BTC prices, exploiting a 7-year dataset collected in the time interval August 2010–October 2017 at 2-day intervals, is discussed in [170], returning that MLP outperforms RNNs in the long-term BTC price prediction.

In [171], a Transformer model is employed to predict stock prices on the Dhaka Stock Exchange, in particular using Time2Vec encoding [172] to manage time series data (namely, both historical daily and weekly data). The experimental results indicate that the Transformer model returns promising predictions, achieving a lower RMSE with respect to ARIMA. Finally, in [173] the authors compare various DL models (including LSTM, RNN, CNN, and Transformer) for predicting major stock market indices (namely: CSI 300, S&P 500, Hang Seng Index, Nikkei 225). The obtained results show that Transformer outperforms the other considered DL models, providing the lowest errors in terms of MSE, MAE, and MAPE.

Table 6.1: Summary of the dataset information.

Observations interval [hour]	Training data size [record]	Test data size [record]	Features [num]
4	4000	1000	4

6.3 Evaluation

The mentioned DL models are evaluated in a comparative way. Their performance is evaluated in terms of prediction accuracy of BTC, ETH, and XRP cryptocurrencies' price, considering different lag sizes for short-term and long-term forecasting.

6.3.1 Dataset and Proposed Model

In order to conduct this analysis, three datasets, each containing 5,000 samples resulting from three cryptocurrencies data obtained from the twelvedata cryptocurrency trading platform [174], have been created. In particular, the data have been obtained exploiting the RESTful APIs provided by twelvedata, which allow to retrieve historical price data on the basis of proper query parameters using a predefined interval.[1] The ETH and XRP price information refer to the time interval August 2020-November 2022, while the BTC price information refer to the time interval November 2020-March 2023, with a 4-hour observation interval in each dataset. More in detail, as shown in Table 6.1, each dataset has been divided into a training set, consisting of 4,000 records, and a test set, composed by 1,000 records. Both sets contain the following 4 features: (i) the price at the start of each interval, denoted as `Open Price`; (ii) the highest price in each interval, denoted as `High Price`; (iii) the lowest price in each interval, denoted as `Low Price`; and (iv) the closing price at each interval, denoted as `Close Price`.

The DL models (namely: MLP, SimpleRNN, LSTM , GRU, BiGRU, BiLSTM) share the same network architecture composed of three layers, with 50, 30, and 20 neurons, respectively. Instead, the CNN model is composed of two 1D-convolution windows (with a size equal to 2), each layer having 64 and 32 feature detectors, re-

[1] Additional information on twelvedata APIs can be found at https://twelvedata.com/docs.

spectively, while a max pooling layer with size equal to 2 is used between the convolutional layers. Then, the Rectified Linear Unit (ReLU) activation function [81, 175,176] (defined as $x^+ \triangleq \max(0,x)$, where x is the input of a neuron) is applied to input and hidden layers of the models. Finally, with regard to the proposed Transformer model, it includes (i) an embedding layer with 128 units to project input data into a suitable dimension, and (ii) a Transformer block with a multi-head attention mechanism using 4 attention heads and an embedding dimension equal to 128. Then, the attention mechanism is followed by dropout regularization, residual connections, and normalization layer. Finally, the block also contains a FFN with two dense layers (128 units each), using ReLU for the first layer and projecting back to the embedding dimension, while dropout is applied after the FFN to regularize the output.

Furthermore, the Scaled Exponential Linear Unit (SELU) activation function, defined as

$$f(\alpha,x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

with $\alpha \simeq 1.6733$ and $\lambda \simeq 1.0507$ [177], is used for the output layer of all the previous algorithms, while the Adam optimizer (detailed in Subsection 6.3.1) is used to minimize the prediction errors of the DL models. Finally, as listed in the detailed architectures of the proposed models in Table 6.2, Table 6.3 and Table 6.4, on the basis of experimental tuning, a learning rate equal to 0.001 and a batch size equal to 32 have been identified and tuned to train the NNs, as shown in Table 6.5. Then, in order to prevent overfitting during the training phase, dropout layers [178], L2 regularization [179], and early stopping [180] techniques have been applied.

Table 6.2: Network architectures of the evaluated DL models.

MLP	**RNN**	**LSTM**
Dense (layer_size = 50, activation = relu)	SimpleRNN (layer_size = 50, activation = relu)	LSTM (layer_size = 50, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 30, activation = relu)	SimpleRNN (layer_size = 30, activation = relu)	LSTM (layer_size = 30, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
Dense (layer_size = 20, activation = relu)	SimpleRNN (layer_size = 20, activation = relu)	LSTM (layer_size = 20, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
Dense (layer_size = *s*, activation = selu, kernel_regularizer = l2)	Dense (layer_size = *s*, activation = selu, kernel_regularizer = l2)	Dense (layer_size = *s*, activation = selu, kernel_regularizer = l2)

GRU	**BiLSTM**	**BiGRU**
GRU (layer_size = 50, activation = relu)	BiLSTM (layer_size = 25, activation = relu)	BiGRU (layer_size = 25, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
GRU (layer_size = 30, activation = relu)	BiLSTM (layer_size = 25, activation = relu)	BiGRU (layer_size = 15, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
GRU (layer_size = 20, activation = relu)	BiLSTM (layer_size = 10, activation = relu)	BiGRU (layer_size = 10, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
Dense (layer_size = *s*, activation = selu, kernel_regularizer = l2)	Dense (layer_size = *s*, activation = selu, kernel_regularizer = l2)	Dense (layer_size = *s*, activation = selu, kernel_regularizer = l2)

Table 6.3: Network architectures of the evaluated CNN-based DL models.

CNN	**CNN-RNN**	**CNN-GRU**
Conv1D (filters = 64, kernel_size = 2, activation = relu)	Conv1D (filters = 64, kernel_size = 2, activation = relu)	Conv1D (filters = 64, kernel_size = 2, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
Maxpooling1D	Maxpooling1D	Maxpooling1D
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
Conv1D (filters = 32, kernel_size = 2, activation = relu)	Conv1D (filters = 32, kernel_size = 2, activation = relu)	Conv1D (filters = 32, kernel_size = 2, activation = relu)
Dropout (0.2)	Dropout (0.2)	Dropout (0.2)
Dense (layer_size = s, activation = selu, kernel_regularizer = l2)	SimpleRNN (layer_size = 32, activation = relu)	GRU (layer_size = 32, activation = relu)
	Dense (layer_size = s, activation = selu, kernel_regularizer = l2)	Dense (layer_size = s, activation = selu, kernel_regularizer = l2)

Table 6.4: Network architecture of the evaluated Transformer model, detailing its composing layers (left) and their corresponding description (right).

Embedding Layer	
Dense (layer_size = 128)	Projects input data to embedding dimension
Transformer Block	
MultiHead (h = 4, key_dim = 128)	Computes self-attention
Dropout (0.1)	Regularizes the attention output
LayerNormalization()	Normalizes the sum of input and attention output
Dense (layer_size = 128, activation = relu)	First layer in FFN
Dense (layer_size = 128)	Projects back to embedding dimension
Dropout (0.1)	Regularizes the FFN output
LayerNormalization()	Normalizes the sum of input and FFN output
Fully Connected Layers	
Flatten()	Flattens the output for the final dense layers
Dropout (0.3)	Regularizes before the output layer
Dense (layer_size = s, activation = selu, kernel_regularizer = l2)	Output layer with L2 regularization

Table 6.5: Summary of the adopted hyperparameters.

Parameters	Learning rate	Optimizer	Loss function	Batch size
Value	0.001	Adam	MSE	32

From an operational point of view, all the considered DL algorithms perform the following main prediction steps. In the *first* pre-processing step, missing values in each column of the dataset are filled with the closest previous value. *Then*, a MinMaxScaler normalization [181] is applied to fit data in the range $[0,1]$—this takes into account the fact that the data of different columns have different distributions. As a matter of fact, this normalization action accelerates the learning process within the DL models. At this point, the dataset must be framed on the basis of the number of forecasting steps and the lag size, in order to perform time series forecasting. Then, as mentioned before, the dataset is divided into two sets (denoted as training and test), where the first is used to train the DL models, while the latter is used to perform cryptocurrencies' prices predictions. *Finally*, predicted and actual data are converted back to the original scale (using the inverse of the MinMaxScaler), and the difference between predicted and actual values is computed.

Adaptive Moment Estimation (Adam) Optimizer

Adam [159] is a computationally efficient optimization algorithm with low memory requirements that combines the benefits of RMSProp and momentu [182]. On the analytical side, in order to compute the learning rate for each weight, Adam exploits the mean (denoted as m_t) and the uncentered variance (denoted as v_t) of the gradients, corresponding to the first and second moments, respectively, as follows:

$$g_t = \nabla\theta f_t(\theta_{t-1}) \tag{6.1}$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{6.2}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{6.3}$$

where: g_t represents the gradient of the loss function f_t with respect to the parameters θ (weights and biases) at time step t, which are then updated during the training phase

to optimize the model's performance; β_1 and β_2 are hyperparameters of the Adam algorithm, typically set to 0.9 and 0.999, respectively.

Then, in order to counteract biases in the moment estimates, especially during initial time steps, bias-corrected estimates are calculated as follows:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{6.4}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}\,. \tag{6.5}$$

The parameter update rule in the Adam algorithm is the following:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{6.6}$$

where α is the learning rate and ε is a small constant (e.g., 10^{-8}) to prevent any division by zero.

6.3.2 Combination of CNN and RNN

The integration of CNNs with RNNs represents a highly effective approach in the field of the DL, since they are specifically designed for problems involving spatial and temporal inter-dependence—e.g., sequential data processing, such as generating captions for images, analyzing videos, and processing data in a sequential manner. Due to their strengths (especially if combined together), in this study two additional hybrid DL models, involving a combination of 1D-CNN, SimpleRNN, and GRU, are considered.

6.3.3 Loss Functions as Evaluation Metrics

In order to evaluate the prediction accuracy of a DL algorithm, a comparison between predicted and actual values needs to be carried out with proper performance metrics. In order to do this, the loss functions used in this study—MAE, RMSE, MAPE, and

R^2—are defined as follows:

$$\text{RMSE}(y,\hat{y}) \triangleq \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}(y_i-\hat{y}_i)^2} \tag{6.7}$$

$$\text{MAE}(y,\hat{y}) \triangleq \frac{1}{N}\sum_{i=0}^{N-1}|y_i-\hat{y}_i| \tag{6.8}$$

$$\text{MAPE}(y,\hat{y}) \triangleq \frac{1}{N}\sum_{i=0}^{N-1}\frac{|y_i-\hat{y}_i|}{\max(\varepsilon,|y_i|)} \tag{6.9}$$

$$R^2(y,\hat{y}) \triangleq 1-\frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}} = 1-\frac{\sum_{i=0}^{N-1}(y_i-\hat{y}_i)^2}{\sum_{i=0}^{N-1}(y_i-\bar{y})^2} \tag{6.10}$$

where: N is the total number of samples in the test set; $y=(y_0,\ldots,y_{N-1})$ is the actual (target) values vector; $\hat{y}=(\hat{y}_0,\ldots,\hat{y}_{N-1})$ is the vector of predicted values; and $\bar{y}$ is the mean of the actual values. With regard to R^2, when $\text{SS}_{\text{res}} > \text{SS}_{\text{tot}}$, then R^2 becomes negative: this indicates that the model's predictions are less accurate than a simple horizontal line representing the mean of the observed values [183]. Nevertheless, given that values scales' dependency might represent a significant drawback, it is important to highlight that data scaling would help in mitigating this undesired problem, ensuring that the metrics provide a fair assessment regardless of the magnitudes of the values in the dataset. This is the reason why, as detailed in Subsection 6.3.1, a MinMaxScaler normalization has been applied to fit data in the range $[0,1]$. Then, while RMSE, MAE, and MAPE are known to be valuable for understanding error magnitude and relative error, R^2 was also included because it offers a scale-invariant measure of how well the model captures the variance in the data. The combined use of all these evaluation metrics provides a comprehensive and balanced evaluation of the model's performance.

6.3.4 Experimental Results

On the basis of the evaluation metrics, presents a comparison and discussion of the experimental results (in terms of cryptocurrencies' prices predictions) obtained exploiting the different DL algorithms presented in the previous sections (namely, MLP,

SimpleRNN, LSTM, GRU, BiLSTM, BiGRU, CNN, CNN-RNN, CNN-GRU, Transformer), for both short- and long-term forecasting.

6.3.5 Accuracy

More in detail, Figure 6.2 and Figure 6.3 represent the RMSE of the considered DL models for the BTC price prediction, considering a 3-step ahead short-term prediction horizon ($s = 3$) and various values of time lag (denoted as $\mathscr{W}$). In detail, $\mathscr{W}$ refers to the number of previous time steps (or observations) considered as input for predicting the cryptocurrencies values in next time steps. For the sake of clarity, with $\mathscr{W} = 5$ and $s = 3$, and considering that the reference dataset has a 4-hour data collection interval (as detailed in Subsection 6.3.1), the next 12-hour currency prices' prediction is based on the information from the past 20 hours. According to the obtained results, Transformer provides (in general) the lowest RMSE and MAE, while MLP and CNN-RNN return the highest RMSE and MAE across different lags. Moreover, as shown in Figure 6.4, Transformer can return a small MAPE considering large time lags (e.g., $\mathscr{W} = \{5, 10, 15, 20, 25\}$). The results of R^2 as the final evaluation metric are shown in Figure 6.5, where Transformer provides the highest value.

Similarly, the experimental performance results for the ETH price prediction, in terms of RMSE and MAE, considering a 3-step ahead short-term prediction horizon ($s = 3$) and various values of time lag $\mathscr{W}$, are shown in Figure 6.6 and Figure 6.7, respectively. In general, on the basis of the experimental results, Transformer returns the lowest error rates if compared to the other DL methods for various time lags $\mathscr{W}$. The same conclusion arises from the results shown in Figure 6.8, with Transformer returning the lowest MAPE for $\mathscr{W} = \{5, 15, 20, 25, 30\}$, while BiLSTM performing better for $\mathscr{W} = \{10\}$. The results of R^2 as the final evaluation metric are shown in Figure 6.9, where Transformer provides the highest value.

Finally, the experimental performance results for the XRP price prediction (in terms of RMSE and MAE) considering a 3-step ahead short-term prediction horizon ($s = 3$) and various values of time lag $\mathscr{W}$ are shown in Figure 6.10 and Figure 6.11, respectively. On the basis of the achieved results, Transformer seems to outperform the other considered DL algorithms. In addition, when examining the re-

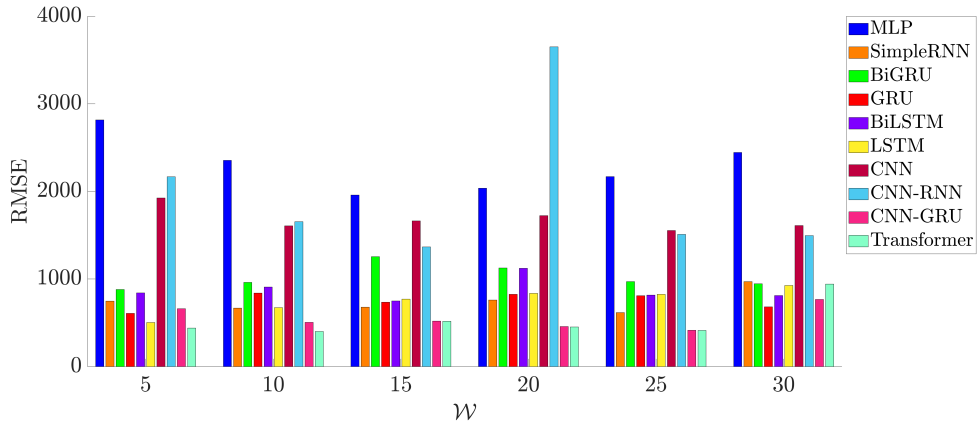

Figure 6.2: Impact, in terms of RMSE, of the value of the time lag $\mathscr{W}$ on the BTC price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

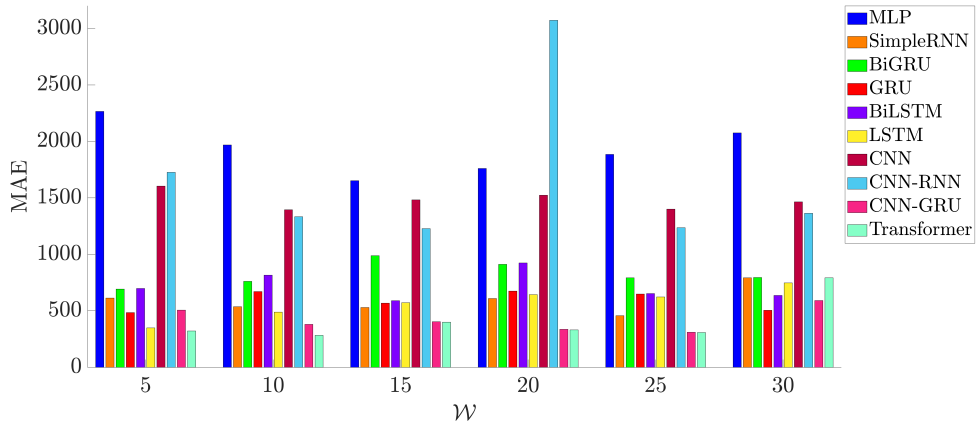

Figure 6.3: Impact, in terms of MAE, of the value of the time lag $\mathscr{W}$ on the BTC price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

sults shown in Figure 6.12, it can be observed that Transformer achieves the lowest MAPE for $\mathscr{W} \in \{5, 15, 20\}$, whereas CNN-GRU and BiLSTM outperform Trans-

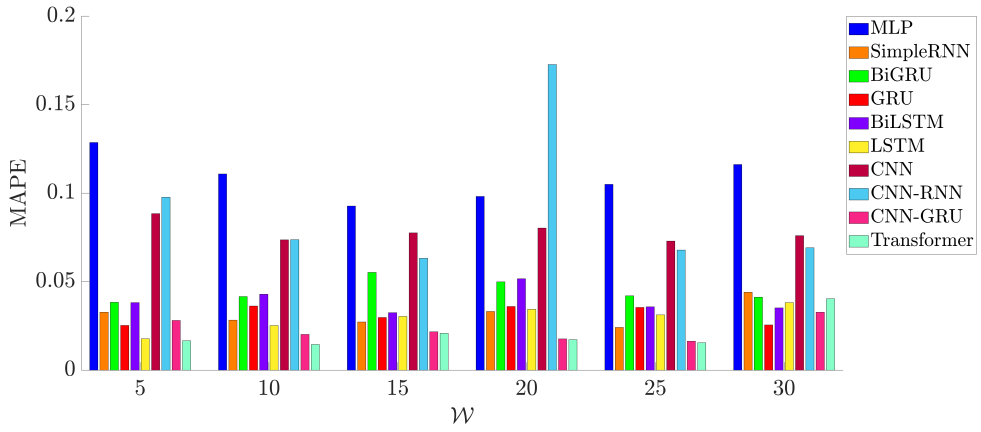

Figure 6.4: Impact, in terms of MAPE, of the value of the time lag $\mathscr{W}$ on the BTC price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

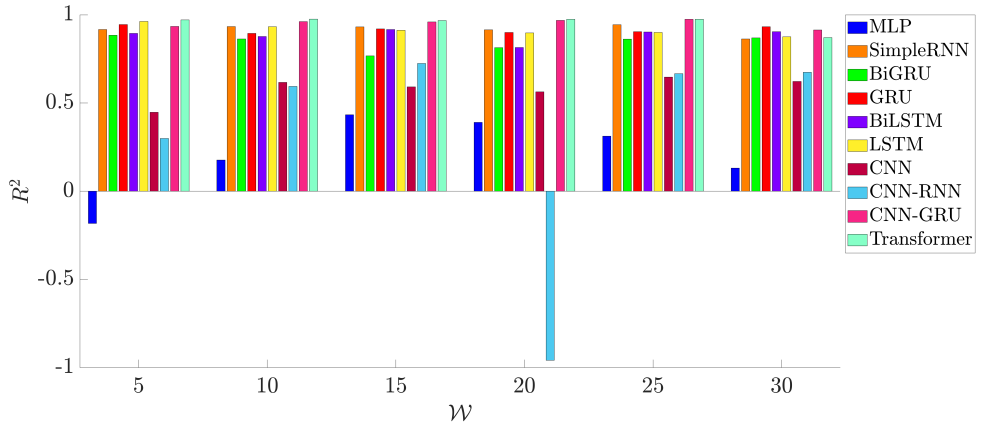

Figure 6.5: Impact, in terms of R^2, of the value of the time lag $\mathscr{W}$ on the BTC price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

former for $\mathscr{W} \in \{25, 30\}$ and $\mathscr{W} = 10$, respectively. As shown in Figure 6.9, the Transformer generally outperforms the other models, except when $\mathscr{W} \in \{10, 25, 30\}$,

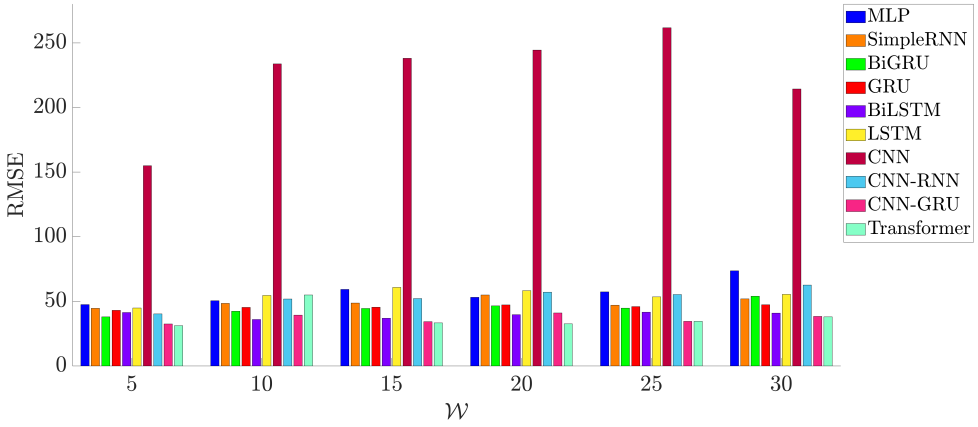

Figure 6.6: Impact, in terms of RMSE, of the value of the time lag $\mathcal{W}$ on the ETH price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

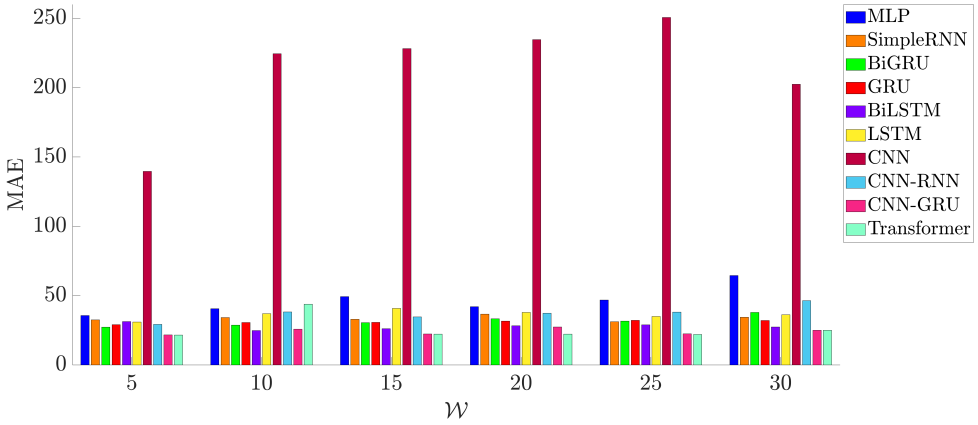

Figure 6.7: Impact, in terms of MAE, of the value of the time lag $\mathcal{W}$ on the ETH price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

where BiLSTM and CNN-GRU achieve higher R^2 values.

For the sake of clarity, a summary of the best-performing DL algorithms, in terms

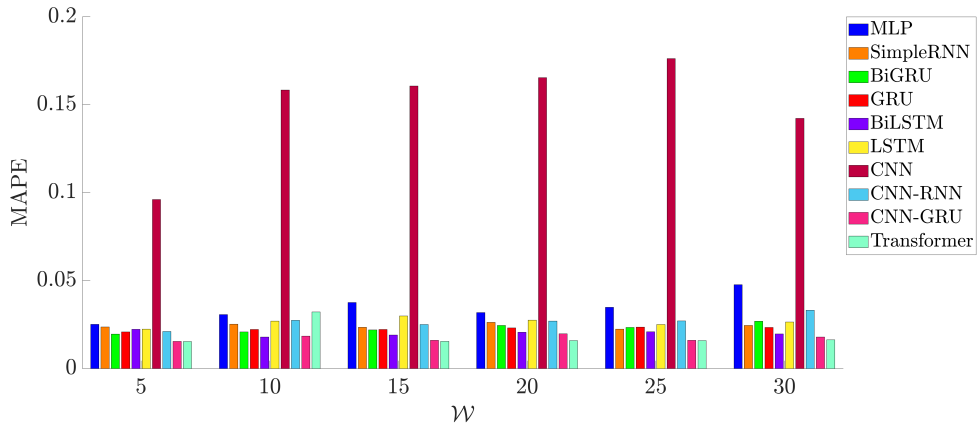

Figure 6.8: Impact, in terms of MAPE, of the value of the time lag $\mathcal{W}$ on the ETH price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

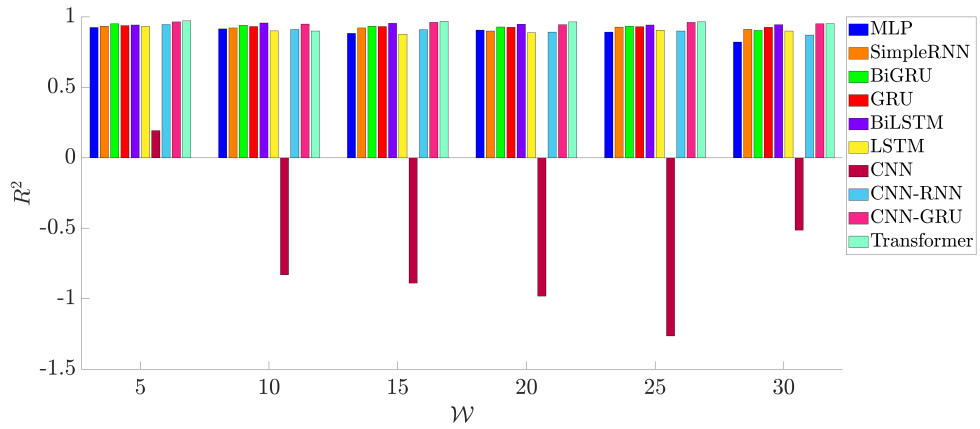

Figure 6.9: Impact, in terms of R^2, of the value of the time lag $\mathcal{W}$ on the ETH price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

of RMSE, is shown in Table 6.6, which highlights how Transformer consistently outperforms the other models in the majority of scenarios, confirming its robustness in

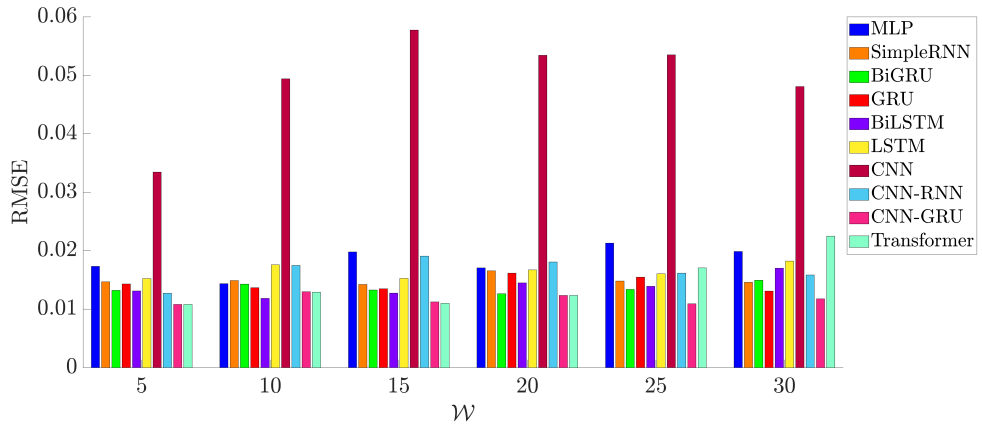

Figure 6.10: Impact, in terms of RMSE, of the value of the time lag $\mathcal{W}$ on the XRP price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

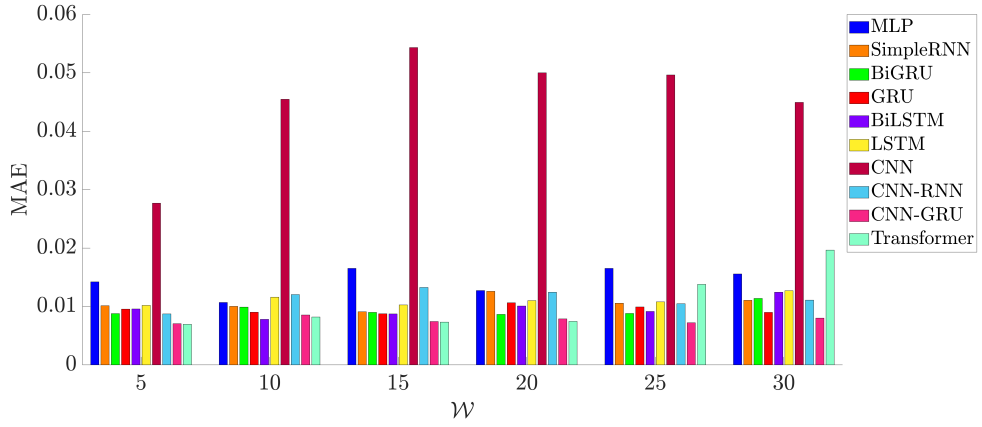

Figure 6.11: Impact, in terms of MAE, of the value of the time lag $\mathcal{W}$ on the XRP price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

capturing temporal dependencies for short-term predictions. To better highlight the performance and generalization capability of the considered models in predicting

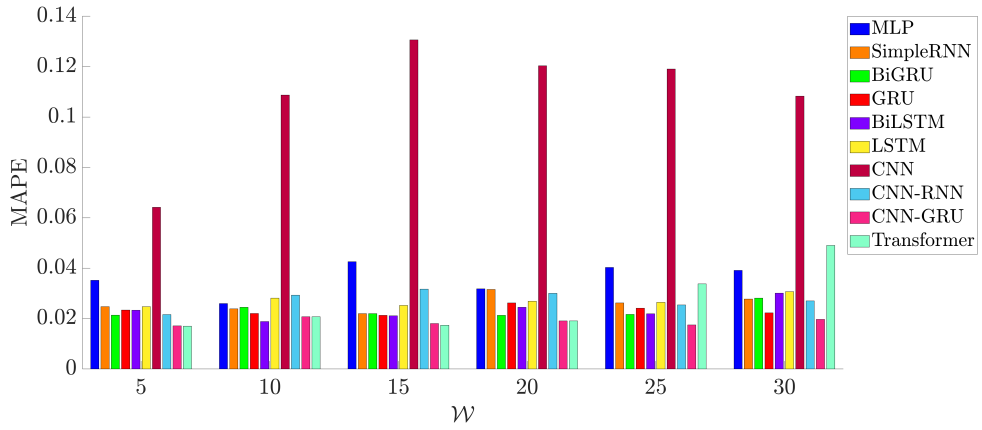

Figure 6.12: Impact, in terms of MAPE, of the value of the time lag $\mathcal{W}$ on the XRP price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

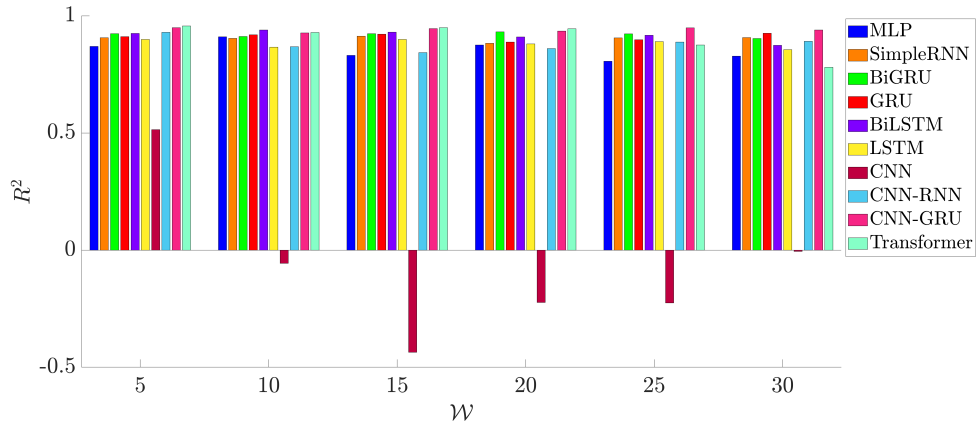

Figure 6.13: Impact, in terms of R^2, of the value of the time lag $\mathcal{W}$ on the XRP price prediction with various DL algorithms, considering a 3-step ahead prediction horizon ($s = 3$).

BTC, ETH, and XRP prices, the training and validation losses are shown in Figure 6.14, Figure 6.15, Figure 6.16, and Figure 6.17.

Table 6.6: Best-performing DL algorithms, according to RMSE, for the cryptocurrencies' prices prediction, considering a 3-step ahead prediction horizon ($s = 3$) and various values of $\mathscr{W}$.

Cryptocurrency	$\mathscr{W} = 5$	$\mathscr{W} = 10$	$\mathscr{W} = 15$	$\mathscr{W} = 20$	$\mathscr{W} = 25$	$\mathscr{W} = 30$
BTC	**Transformer**	**Transformer**	**Transformer**	**Transformer**	**Transformer**	GRU
ETH	**Transformer**	BiLSTM	**Transformer**	**Transformer**	**Transformer**	**Transformer**
XRP	**Transformer**	BiLSTM	**Transformer**	**Transformer**	CNN-GRU	CNN-GRU

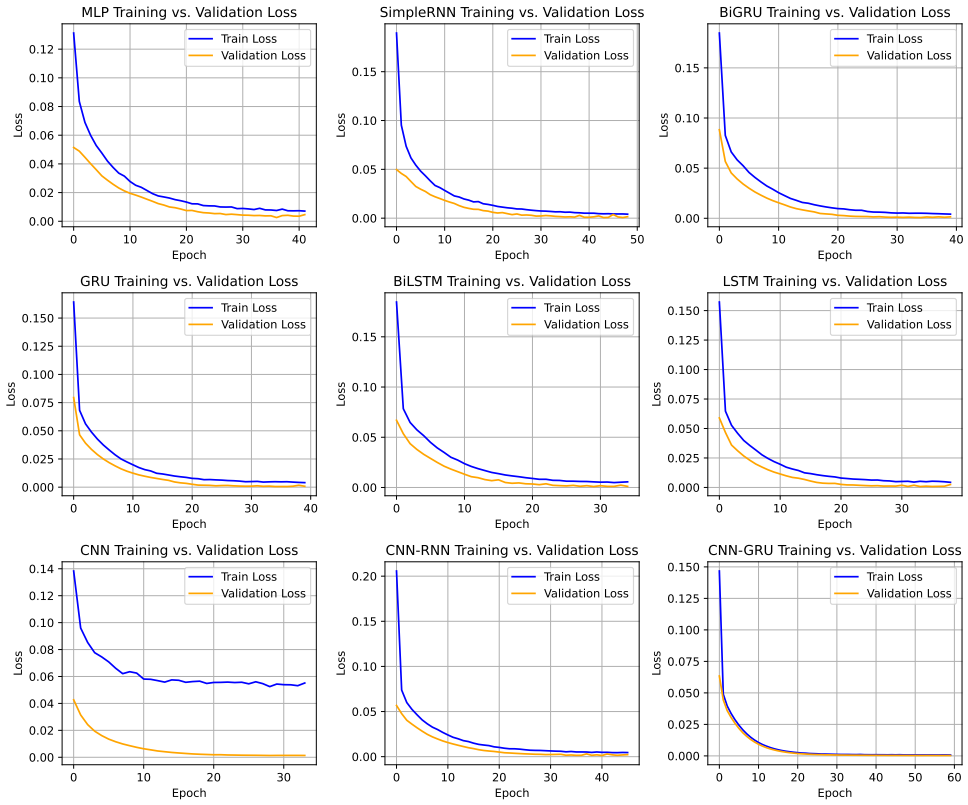

Figure 6.14: Training and validation loss for BTC using the proposed DL models.

Then, in order to evaluate the accuracy in the case of long-term cryptocurrencies prices forecasting, the prediction horizon has been extended to 10-, 15-, and 20-step ahead ($s \in \{10, 15, 20\}$), considering a fixed time lag $\mathscr{W} = 25$. In particular, after

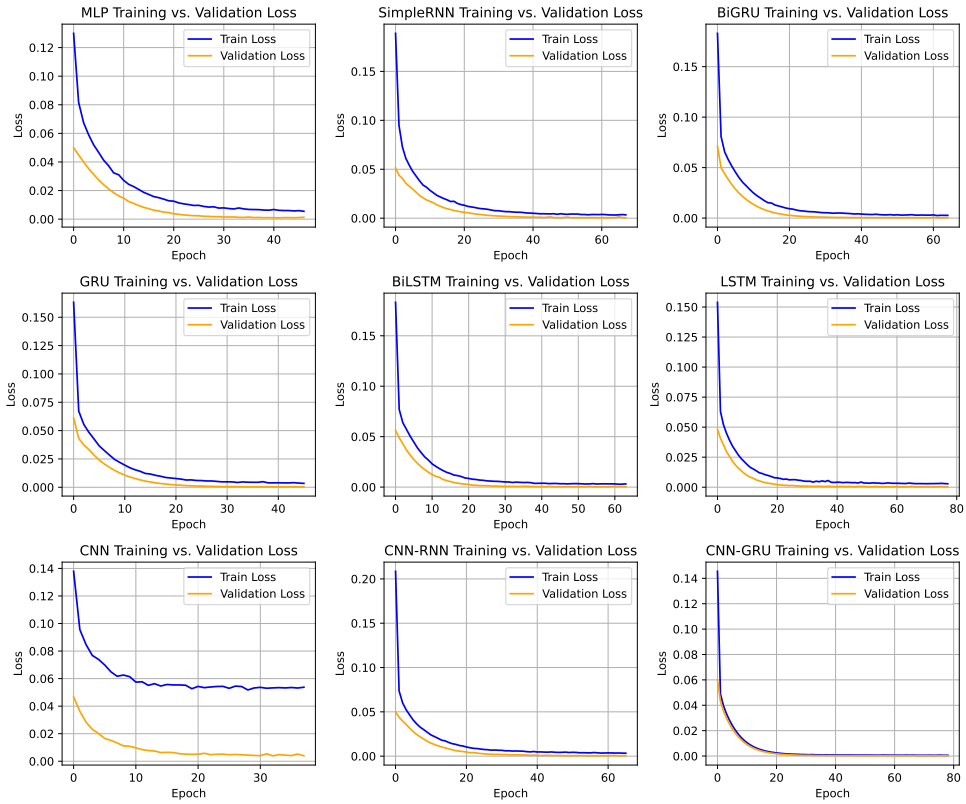

Figure 6.15: Training and validation loss for ETH using the proposed DL models.

experimental tuning activities, $\mathscr{W} = 25$ has been selected since it represents a good trade-off between low values of the error metrics (MSE, RMSE, MAE, MAPE) and high value of R^2, thus indicating accurate and reliable predictions.

At this point, recalling that the collection time interval in the dataset is equal to 4 hours, the considered values of s correspond to cryptocurrencies' price predictions for the next 40, 60, and 80 hours, respectively. The experimental findings achieved with this long-term prediction assessment in terms of RMSE for BTC, ETH, and XRP, are presented in Figure 6.18, Figure 6.22, and Figure 6.26, respectively. On the basis of these results and as described in Table 6.7, Transformer is more accurate (if com-

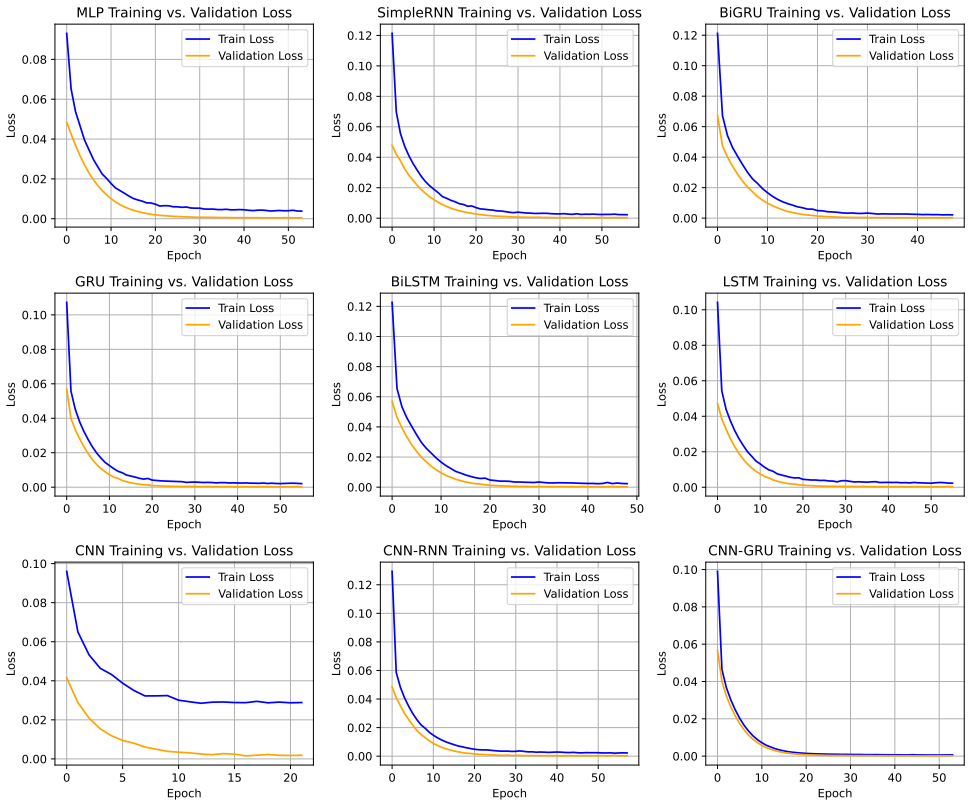

Figure 6.16: Training and validation loss for XRP using the proposed DL models.

pared to the other evaluated DL models) for long-term price prediction. Furthermore, the results of the comparison, in terms of MAE, of BTC, ETH, and XPR are shown in Figure 6.19, Figure 6.23, and Figure 6.27, respectively. More exactly, Transformer returns the lowest MAE rate in all of the long-term predictions for ETH and BTC, as well as trials with XRP ($s \in \{10, 20\}$). The evaluations in respect to the MAPE are shown in Figure 6.20, Figure 6.24, and Figure 6.28, respectively, where the superiority of Transformer in clear. Finally, R^2 is the last evaluated metric that illustrated for BTC, ETH and XRP in Figure 6.21, Figure 6.25, and Figure 6.29, respectively. On the basis of the collected data, similar to the previous results, Transformer outperforms

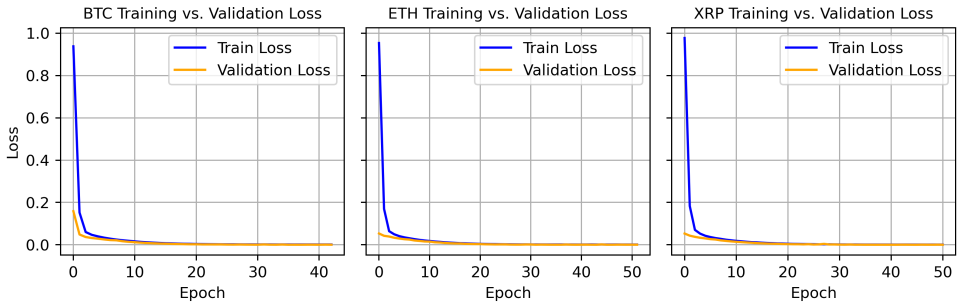

Figure 6.17: Training and validation loss for BTC, ETH, and XRP using the proposed Transformer model.

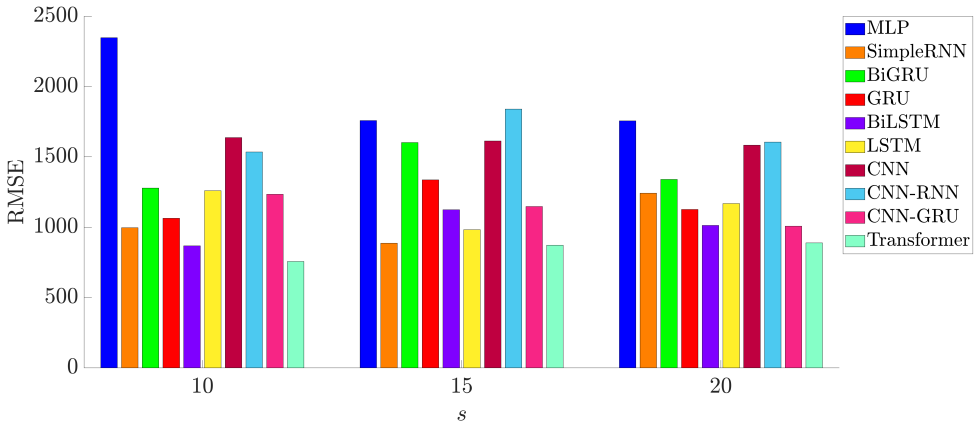

Figure 6.18: Impact, in terms of RMSE, of the value of the prediction horizon s on the BTC price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

the other models in terms of R^2.

Finally, for the sake of clarity, real prices and those predicted through the considered DL algorithms, are shown for BTC, ETH, and XRP in Figure 6.30, Figure 6.31, and Figure 6.32, respectively.

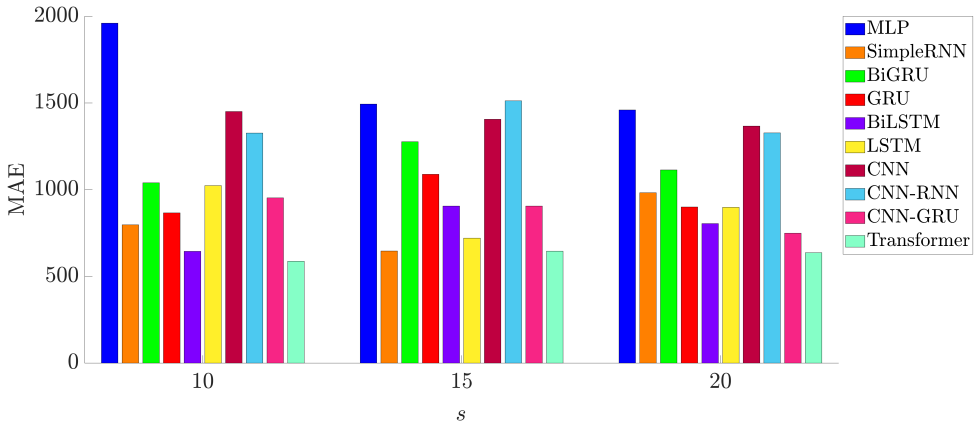

Figure 6.19: Impact, in terms of MAE, of the value of the prediction horizon s on the BTC price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

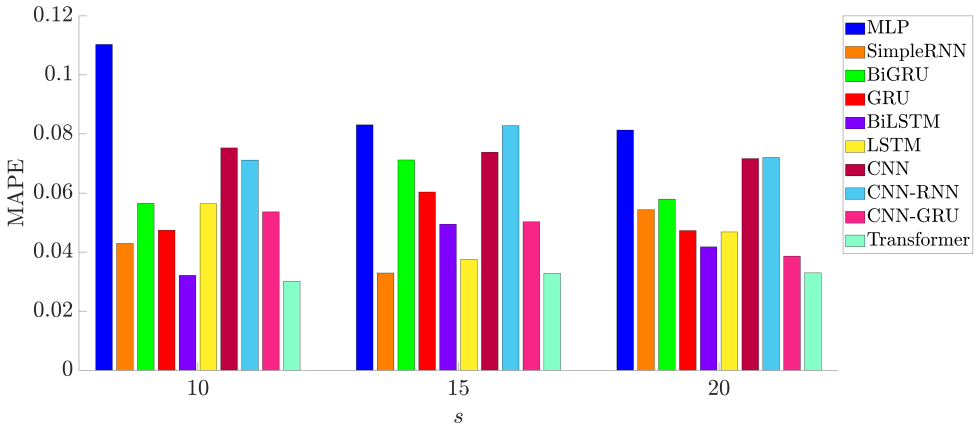

Figure 6.20: Impact, in terms of MAPE, of the value of the prediction horizon s on the BTC price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

6.3.6 Computational Complexity

In order to investigate the computational complexity of the considered DL algorithms, the number of MACC operations, the number of parameters, and the FLASH memory size returned by each DL model, have been considered as performance metrics

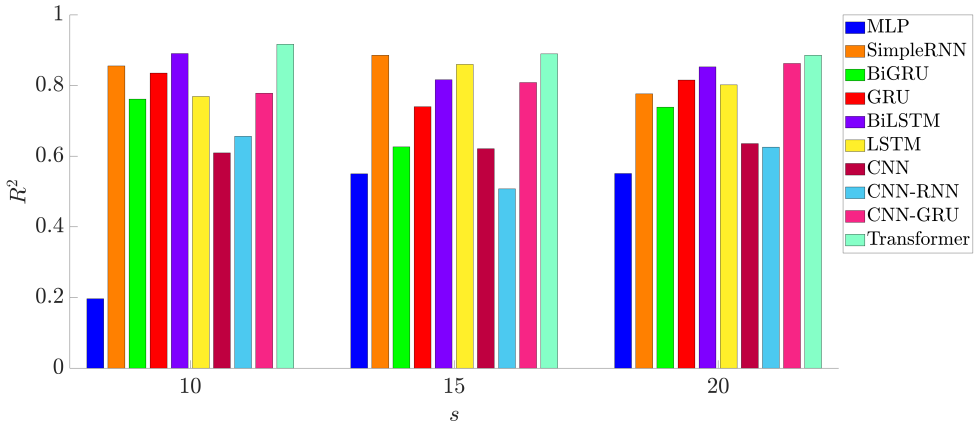

Figure 6.21: Impact, in terms of R^2, of the value of the prediction horizon s on the BTC price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

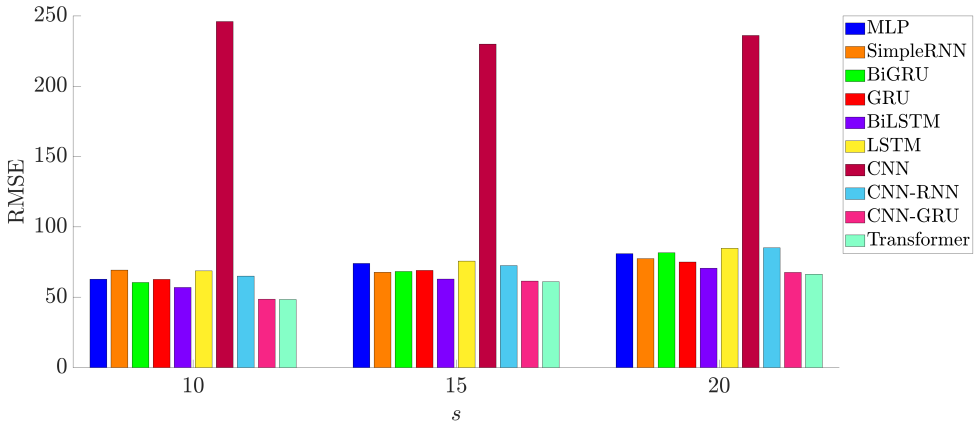

Figure 6.22: Impact, in terms of RMSE, of the value of the prediction horizon s on the ETH price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

exploiting the ST Edge AI Developer Cloud [74] tool. As a result, the obtained performance is shown in Table 6.8, where it can be observed that CNN-RNN returns the lowest number of MACC operations and parameters in comparison to the other discussed RNNs, while SimpleRNN offers the smallest FLASH memory size.

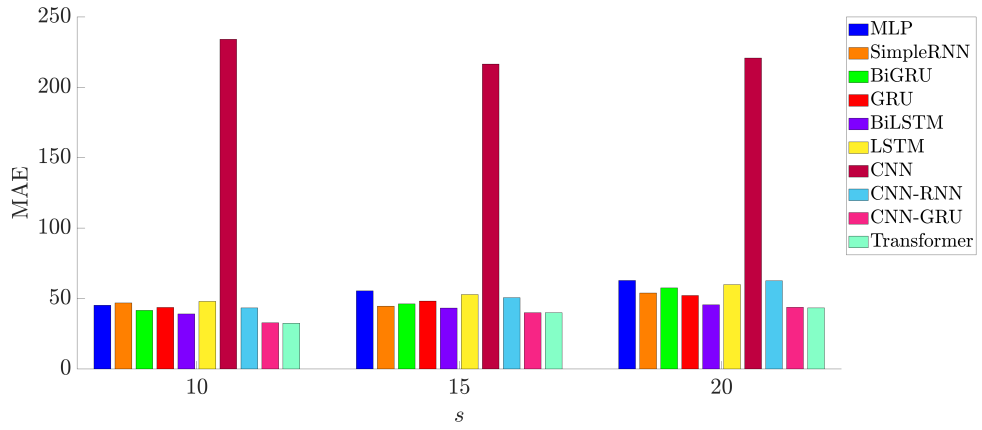

Figure 6.23: Impact, in terms of MAE, of the value of the prediction horizon s on the ETH price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

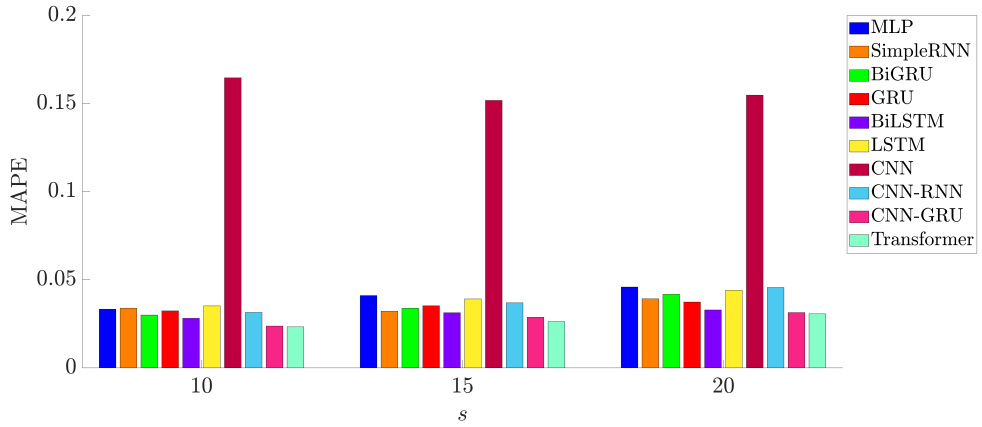

Figure 6.24: Impact, in terms of MAPE, of the value of the prediction horizon s on the ETH price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

6.3.7 Joint Accuracy-Complexity Trade-Off

Finally, in order to evaluate the overall computational-accuracy performance trade-off of the considered DL algorithms in predicting BTC, ETH, and XRP prices, it is crucial to assess a performance metric that considers both the computing complexity

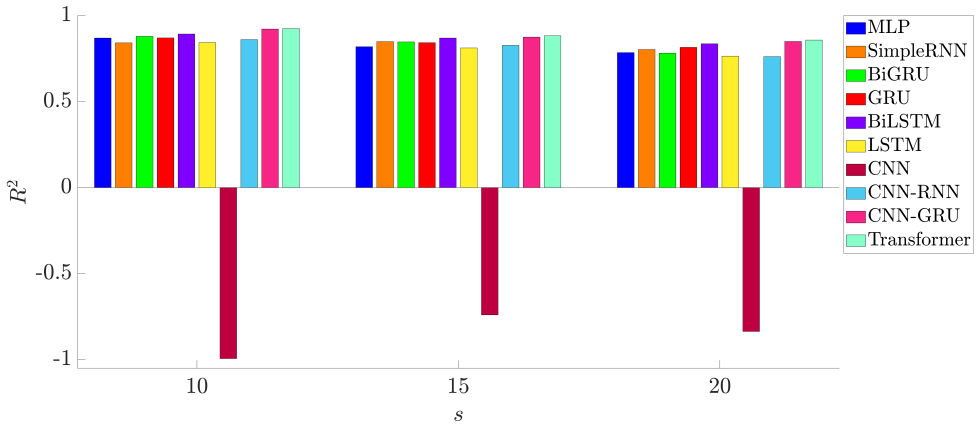

Figure 6.25: Impact, in terms of R^2, of the value of the prediction horizon s on the ETH price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

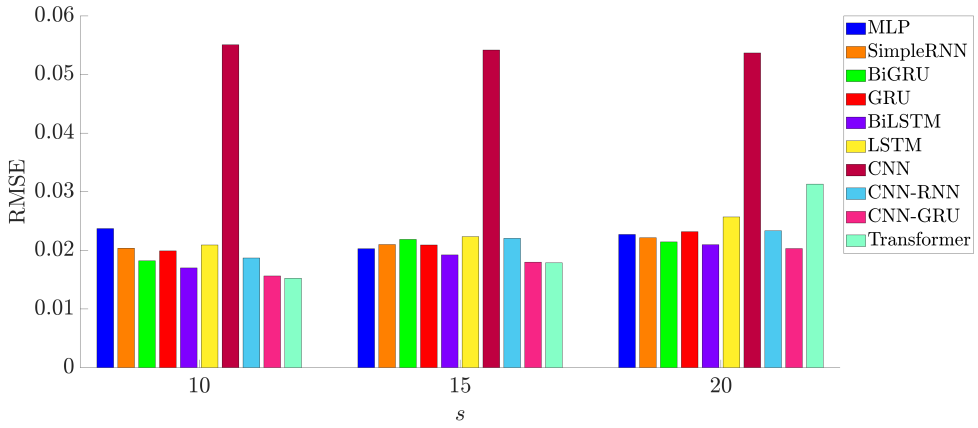

Figure 6.26: Impact, in terms of RMSE, of the value of the prediction horizon s on the XRP price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

and the accuracy. To this end, the following metric is defined:

$$\xi \triangleq \text{MACCs Number} * \text{Average MAPE}. \tag{6.11}$$

Then, the average MAPE has been calculated considering various time lags $\mathscr{W} \in \{5, 10, 15, 20, 25, 30\}$ and a prediction horizon $s = 3$ for the different DL models of

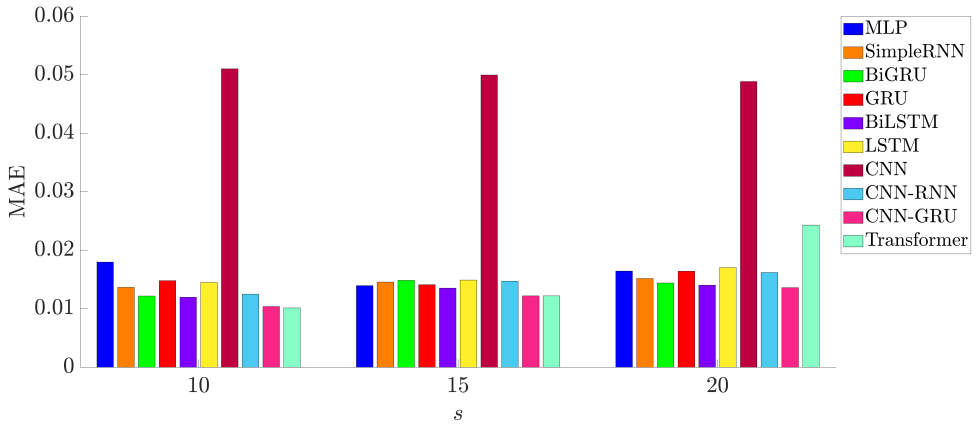

Figure 6.27: Impact, in terms of MAE, of the value of the prediction horizon s on the XRP price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

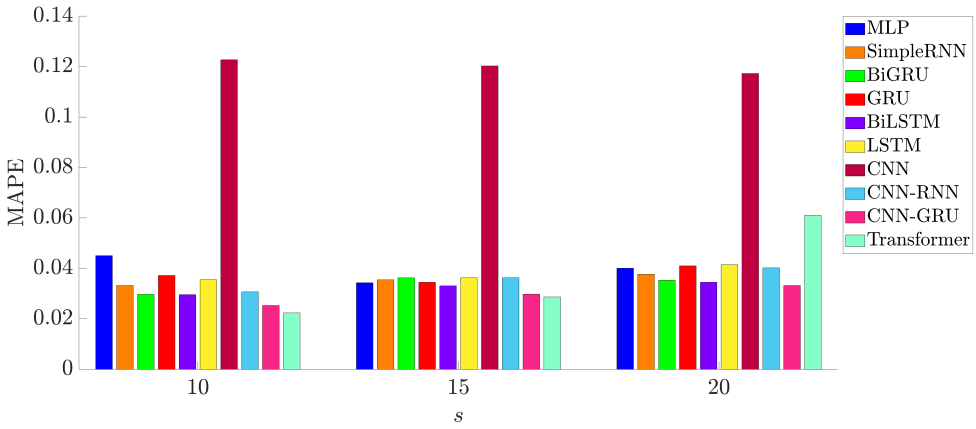

Figure 6.28: Impact, in terms of MAPE, of the value of the prediction horizon s on the XRP price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

interest. According to the results shown in Table 6.9, CNN-GRU provides the lowest value of ξ when focusing on the BTC and ETH price prediction, while CNN-RNN returns the lowest value of ξ for XPR price prediction.

These results indicate that the time lag substantially impacts the prediction accu-

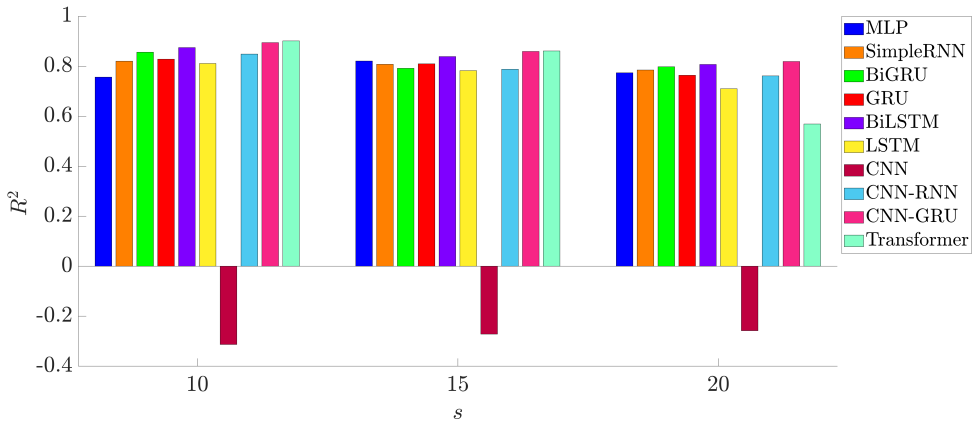

Figure 6.29: Impact, in terms of R^2, of the value of the prediction horizon s on the XRP price prediction with various DL algorithms, considering a time lag $\mathscr{W} = 25$.

Table 6.7: Best-performing DL algorithms, according to RMSE, for the cryptocurrencies' prices prediction, considering various long-term step ahead prediction horizons $s \in \{10, 15, 20\}$ and a fixed time lag $\mathscr{W} = 25$.

Cryptocurrency	$s = 10$	$s = 15$	$s = 20$
BTC	**Transformer**	**Transformer**	**Transformer**
ETH	**Transformer**	**Transformer**	**Transformer**
XRP	**Transformer**	**Transformer**	CNN-GRU

Table 6.8: Complexity comparison of the considered DL models.

Model	SimpleRNN	LSTM	BiLSTM	GRU
MACCs number	155,099	622,599	432,599	462,599
Parameters number	**6,263**	24,863	17,263	18,963
FLASH size [KiB]	**37**	116	89	93
Model	BiGRU	CNN-RNN	CNN-GRU	Transformer
MACCs number	320,099	**83,879**	129,287	1,151,360
Parameters number	13,263	6,883	11,139	299,907
FLASH size [KiB]	74	46	69	970

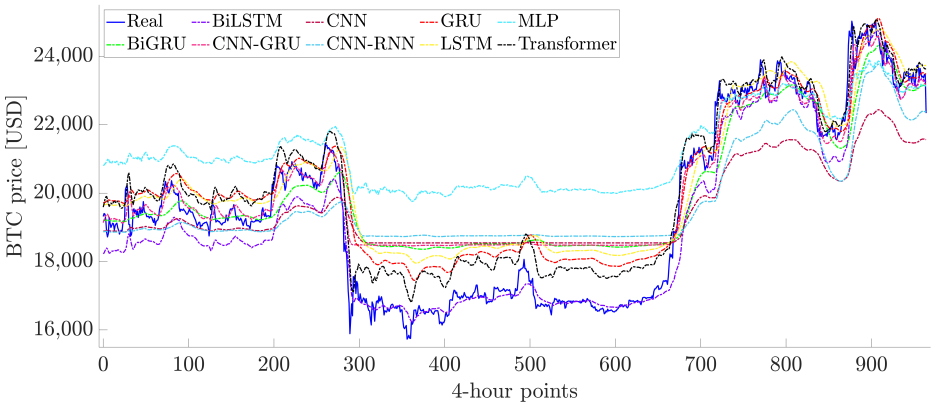

Figure 6.30: BTC price long-term prediction results obtained by the considered DL models, with a time lag $\mathscr{W} = 25$, and considering a 10-step ahead prediction horizon ($s = 10$).

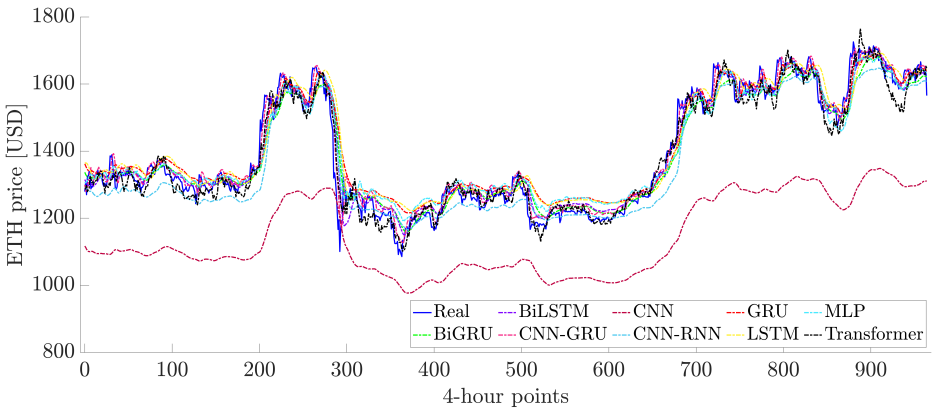

Figure 6.31: ETH price long-term prediction results obtained by the considered DL models, with a time lag $\mathscr{W} = 25$, and considering a 10-step ahead prediction horizon ($s = 10$).

racy: in fact, shorter values of the time lag capture immediate effects and short-term patterns, whereas longer time lags capture stable trends and cyclic behaviors. Moreover, the selected time lag sizes were customized to the differentiate volatility and

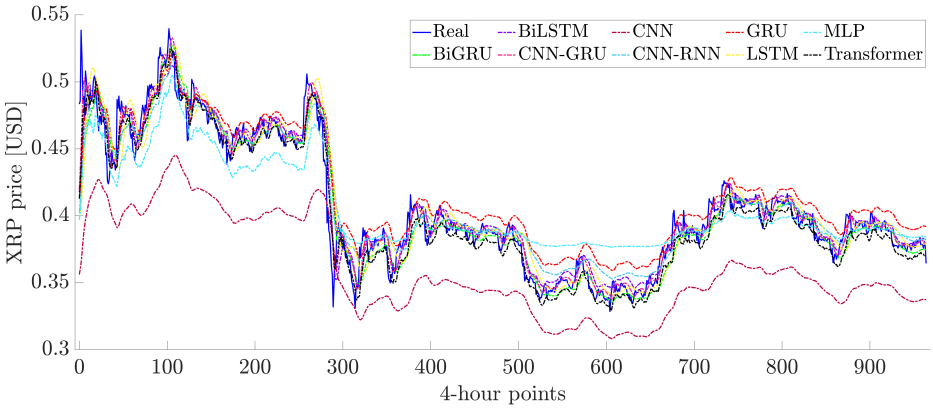

Figure 6.32: XRP price long-term prediction results obtained by the considered DL models, with a time lag $\mathscr{W} = 25$, and considering a 10-step ahead prediction horizon ($s = 10$).

Table 6.9: Joint accuracy-complexity ξ of the considered DL models.

Model	SimpleRNN	LSTM	BiLSTM	GRU
BTC	4,898.29	18,371.39	17,027.93	14,509.03
ETH	3,761.29	16,428.58	8,695.51	10,435.69
XRP	4,032.3	16,803.34	10,063.94	10,724.07
Model	BiGRU	CNN-RNN	CNN-GRU	Transformer
BTC	14,309.22	7,608.38	**2,945.79**	23,976.89
ETH	7,323.34	2,248.59	**2,235.88**	21,324.44
XRP	7,399.53	**2,305.26**	2,412.08	30,091.47

momentum patterns of various cryptocurrencies, thus improving the models' capacity to derive insights from past data. Then, by fine-tuning these time lags, the prediction precision was enhanced, providing traders and investors with more reliable forecasts to make well-informed decisions in practical situations.

6.3.8 Research Limitations

Finally, for the sake of completeness, in the following a few potential limitations of the proposed approach (which can thus lead to further improvements and research

directions) are highlighted and discussed.

- *Determining the appropriate time lag*: identifying the most suitable time lag for prediction requires substantial experimentation. In fact, the selection process is empirical and not straightforward, involving multiple comparisons in order to determine the appropriate value for accurate forecasts and understand the relationship between prediction accuracy and varying lag sizes.
- *Model generalization across cryptocurrencies*: developing a model able to predict values for all the existing cryptocurrencies with the lowest error is challenging, given the fact that each cryptocurrency has unique features and market behaviors. This makes the (ideal) goal of constructing a generally applicable forecasting model exceedingly difficult.
- *Limited historical data*: the availability of short historical data for new cryptocurrencies can complicate the creation of credible forecasting models, since accurate models generally depend on significant historical data in order to be able to capture trends and patterns properly.
- *Market volatility and unpredictability*: the tremendous volatility and unpredictability of the cryptocurrency market can severely impair prediction accuracy. This is due to sudden market movements, regulatory developments, and other external factors, that can lead to considerable failures in predictions.
- *Data quality*: the precision of predictions is contingent on the data quality. Therefore, it is crucial to ensure that the data collection process is robust and efficient.
- *Overfitting*: DL models are susceptible to overfitting, which makes it essential to carefully tune the model and apply appropriate regularization techniques.

6.4 Conclusions

This study presents and discusses an experimental comparative analysis of various DL algorithms, including MLP, SimpleRNN, LSTM, BiLSTM, GRU, BiGRU, CNN,

CNN-RNN, CNN-GRU, and Transformer, for short- and long-term price prediction of three well-known cryptocurrencies: BTC, ETH, and XRP. The obtained results highlight that, for both *short-term* and *long-term* predictions, the Transformer model outperforms the other considered DL algorithms in all the cases, returning the lowest prediction errors (including RMSE, MAE, and MAPE) and achieving the highest R^2 values across various experiments. Furthermore, on the basis of experimental findings, CNN-RNN exhibits the lowest complexity level, measured in terms of the number of MACC operations. On the other hand, SimpleRNN has the lowest number of parameters and requires the smallest amount of FLASH memory (compared to the other DL models). Finally, by investigating the joint accuracy-complexity through the metric ξ introduced in (6.11), our results show that CNN-GRU provides the lowest (i.e., the best) value of ξ when predicting future BTC and ETH prices, while CNN-RNN yields the best performance (i.e., the lowest value of ξ) when the prediction focuses on XRP. In fact, although the Transformer model returns the highest prediction accuracy, it is also the most computationally expensive and is not the best considering the metric ξ. The complexity of the Transformer may be a limitation depending on the available resources on the chosen processing platform. Future research could focus on integrating sentiment analysis-based approaches and the influence of macroeconomic factors into predictive models to enhance the accuracy of cryptocurrency price predictions.

Chapter 7

Conclusions

In this Thesis dissertation, AI has been employed to model and analyze complex data, extracting meaningful patterns and features, and has been outstanding in solving both linear and nonlinear problems. Many times, AI algorithms outperform traditional methods in terms of accuracy, efficiency, and scalability. Currently, an extensive range of AI algorithms is utilized across various domains, including NLP, healthcare, and environmental monitoring. Concurrently, microcontrollers and IoT devices have been enhanced, enabling the deployment of AI on resource-limited IoT systems. These enable IoT devices, widespread in everyday life, to collect, analyze, and process data locally with enhanced intelligence.

Local data processing has several advantages. By lowering latency dramatically, it enables real-time decision-making and speedy responses to dynamic situations. This strategy increases privacy and security by eliminating reliance on the transfer of data from third-party suppliers, lowering vulnerabilities, and boosting data integrity. This capability for real-time makes the devices of IoT more responsive and efficient in their application to environmental monitoring, human activity recognition for healthcare and fitness applications, or object detection in video streams for surveillance and automation.

This Thesis further indicates that there is a lot of opportunity to improve AI, more precisely ML and DL models, for operations on resource-constrained devices.

The core contribution of this Thesis lies in exploring how to design and optimize ML/DL models for real-world deployments on constrained hardware. Applications span from air quality prediction and healthcare to cryptocurrency predictions and human activity identification, illustrating the great applicability of embedded AI. This study seeks to build a case that the real-world deployments on edge devices can be facilitated by the approaches of BO, QAT, and PTQ in a quest for model accuracy, computing efficiency, and deployability trade-offs. These studies illustrate how embedded AI is able to transform intelligent, efficient, and secure IoT systems. For the sake of completeness and clarity, the best-performing models, in terms of joint trade-off between prediction accuracy and computational complexity, which provide high prediction accuracy while maintaining efficient computational complexity, are summarized in Table 7.1. In fact, these models were selected on the basis of (i) their suitability for deployment on resource-constrained tiny IoT devices and (ii) their performance within the specific applications discussed in this Thesis. The findings indicate that GRU, CNN, and their combinations offer a suitable compromise between prediction accuracy and computational complexity across diverse applications. However, for audio-based applications, specifically Parkinson's disease detection and tiny audio denoising, these models struggled to effectively capture the temporal and spectral features intrinsic to audio data.

In more detail, the purpose of these studies that are presented in this Thesis was to undertake a full inquiry regarding the integration of DL and ML models in IoT devices with limited capabilities and delve into the methodologies that can help us produce robust, efficient, and lite deployable models. These findings show that we cannot discover a strategy that performs the best in all studies with varied natures of data, and each ML/DL model has its own advantages and drawbacks in performing prediction and classification tasks. Moreover, these studies cover many ways of preprocessing for inputs, from cryptocurrency features to the data from wearable sensors that track human activity to the audio data and the data from environmental sensors that highlights the importance of the preprocessing phase in predictions efficiency. The preprocessing procedure and techniques that are required are dependent on the nature of the raw data that we want to process.

Table 7.1: Table summarizing the selected DL models optimized for joint accuracy and complexity across the different application scenarios discussed in this Thesis.

Topic	Best Model (Joint Accuracy-Complexity)
Air quality prediction (Chapter 2)	CNN-BiGRU & GRU
Parkinson Disease detection (Chapter 3)	Q1D-CNN
Tiny audio denoising (Chapter 4)	LMU
Sport activities classification (Chapter 5)	CNN-BiGRU
Cryptocurrency price prediction (Chapter 6)	CNN-GRU

Bibliography

[1] Kerem Altun and Billur Barshan. Daily and Sports Activities Data Set, 2019. IEEE Dataport, doi:10.21227/at1v-6f84. doi:10.21227/at1v-6f84.

[2] Leonardo Babun, Kyle Denney, Z Berkay Celik, Patrick McDaniel, and A Selcuk Uluagac. A survey on iot platforms: Communication, security, and privacy perspectives. *Computer Networks*, 192:108040, 2021.

[3] Syed Saba Raoof and MA Saleem Durai. A comprehensive review on smart health care: applications, paradigms, and challenges with case studies. *Contrast Media & Molecular Imaging*, 2022(1):4822235, 2022.

[4] Moammar Dayoub, Saida Shnaigat, Radi A Tarawneh, Azzam N Al-Yacoub, Faisal Al-Barakeh, and Khaled Al-Najjar. Enhancing animal production through smart agriculture: Possibilities, hurdles, resolutions, and advantages. *Ruminants*, 4(1):22–46, 2024.

[5] Jose Sanchez Gracias, Gregory S Parnell, Eric Specking, Edward A Pohl, and Randy Buchanan. Smart cities—a structured literature review. *Smart Cities*, 6(4):1719–1743, 2023.

[6] Michael J McGrath, Cliodhna Ní Scanaill, Michael J McGrath, and Cliodhna Ní Scanaill. Key sensor technology components: hardware and software overview. *Sensor Technologies: Healthcare, Wellness, and Environmental Applications*, pages 51–77, 2013.

[7] Deepti Sehrawat and Nasib Singh Gill. Smart sensors: Analysis of different types of iot sensors. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 523–528. IEEE, 2019.

[8] STMicroelectronics. STM32CubeMX, 2024. `https://www.st.com/en/development-tools/stm32cubemx.html`. Accessed on March 12, 2024.

[9] Abhi Raj and Dan Steingart. Power sources for the internet of things. *Journal of The Electrochemical Society*, 165(8):B3130, 2018.

[10] Praveen Kumar Malik, Rohit Sharma, Rajesh Singh, Anita Gehlot, Suresh Chandra Satapathy, Waleed S Alnumay, Danilo Pelusi, Uttam Ghosh, and Janmenjoy Nayak. Industrial internet of things and its applications in industry 4.0: State of the art. *Computer Communications*, 166:125–139, 2021.

[11] Massimo Merenda, Carlo Porcaro, and Demetrio Iero. Edge machine learning for ai-enabled iot devices: A review. *Sensors*, 20(9):2533, 2020.

[12] Viktor Prutyanov, Nikita Melentev, Daniil Lopatkin, Alexander Menshchikov, and Andrey Somov. Developing iot devices empowered by artificial intelligence: Experimental study. In *2019 Global IoT Summit (GIoTS)*, pages 1–6. IEEE, 2019.

[13] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. `doi:10.1109/72.279181`.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997. `doi:10.1162/neco.1997.9.8.1735`.

[15] Junyoung Chung et al. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv preprint arXiv:1412.3555*, 1(1):1–9, 2014. `doi:10.48550/arXiv.1412.3555`.

[16] Pedro Lara-Benítez et al. Temporal Convolutional Networks Applied to Energy-Related Time Series Forecasting. *Applied Sciences*, 10(7), 2020. doi:10.3390/app10072322. doi:10.3390/app10072322.

[17] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv preprint arXiv:1803.01271*, 2018. doi:10.48550/arXiv.1803.01271. doi:10.48550/arXiv.1803.01271.

[18] Xianglong Luo et al. A Deep Learning Prediction Model for Structural Deformation Based on Temporal Convolutional Networks. *Computational Intelligence and Neuroscience*, 2021:1–12, 2021. doi:10.1155/2021/8829639. doi:10.1155/2021/8829639.

[19] Jiaojiao Zhu, Liancheng Su, and Yingwei Li. Wind power forecasting based on new hybrid model with TCN residual modification. *Energy and AI*, 10:100199, 2022. doi:10.1016/j.egyai.2022.100199. doi:10.1016/j.egyai.2022.100199.

[20] Aaron Voelker, Ivana Kajić, and Chris Eliasmith. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/952285b9b7e7a1be5aa7849f32ffff05-Paper.pdf.

[21] Ashish Vaswani et al. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 1(1):1–15, 2017. doi:10.48550/arXiv.1706.03762.

[22] Maria Rita Palattella, Stefano Scanzio, and Sinem Coleri Ergen. *Ad-Hoc, Mobile, and Wireless Networks: 18th International Conference on Ad-Hoc Networks and Wireless, ADHOC-NOW 2019, Luxembourg, Luxembourg, October 1–3, 2019, Proceedings*. Springer, Berlin, Germany, 2019. doi:10.1007/978-3-030-31831-4.

[23] Caosen Xu et al. A Financial Time-Series Prediction Model based on Multiplex Attention and Linear Transformer Structure. *Applied Sciences*, 13(8):5175, 2023. doi:10.3390/app13085175.

[24] Sabeen Ahmed et al. Transformers in Time-Series Analysis: A Tutorial. *Circuits, Systems, and Signal Processing*, 42(12):7433–7466, 2023. doi:10.1007/s00034-023-02454-8.

[25] Sushant Singh and Ausif Mahmood. The NLP Cookbook: Modern Recipes for Transformer Based Deep Learning Architectures. *IEEE Access*, 9:68675–68702, 2021. doi:10.1109/ACCESS.2021.3077350.

[26] Bert Brunekreef and Stephen T Holgate. Air pollution and health. *The Lancet*, 360(9341):1233–1242, 2002. doi:10.1016/S0140-6736(02)11274-8. doi:10.1016/S0140-6736(02)11274-8.

[27] Morton Lippmann. Particulate matter (pm) air pollution and health: regulatory and policy implications. *Air Quality, Atmosphere & Health*, 5:237–241, 2012. doi:10.1007/s11869-011-0136-5. doi:10.1007/s11869-011-0136-5.

[28] Sanghyuk BBae and Yun-Chul Hong. Health effects of particulate matter. *Journal of the Korean Medical Association*, 61(12):749–755, 2018. doi:10.5124/jkma.2018.61.12.749. doi:10.5124/jkma.2018.61.12.749.

[29] Aaron J Cohen et al. The global burden of disease due to outdoor air pollution. *Journal of Toxicology and Environmental Health, Part A*, 68(13-14):1301–1307, 2005. doi:10.1080/15287390590936166. doi:10.1080/15287390590936166.

[30] EF Darley and JT Middleton. Problems of Air Pollution in Plant Pathology. *Annual Review of Phytopathology*, 4:103–118, 1966. doi:10.1146/annurev.py.04.090166.000535. doi:10.1146/annurev.py.04.090166.000535.

[31] Daniele Sofia et al. Mitigation strategies for reducing air pollution. *Environmental Science and Pollution Research*, 27(16):19226–19235, 2020. doi:10.1007/s11356-020-08647-x. doi:10.1007/s11356-020-08647-x.

[32] Aleksy Kwilinski, Oleksii Lyulyov, and Tetyana Pimonenko. Reducing transport sector CO2 emissions patterns: Environmental technologies and renewable energy. *Journal of Open Innovation: Technology, Market, and Complexity*, 10(1):100217, 2024. doi:10.1016/j.joitmc.2024.100217. doi:10.1016/j.joitmc.2024.100217.

[33] Shivangi Saxena Somvanshi et al. *Delhi Air Pollution Modeling Using Remote Sensing Technique*, pages 1–27. Springer International Publishing, 2019. doi:10.1007/978-3-319-58538-3_174-1.

[34] Yong Chen et al. Multifactor Spatio-Temporal Correlation Model Based on a Combination of Convolutional Neural Network and Long Short-Term Memory Neural Network for Wind Speed Forecasting. *Energy Conversion and Management*, 185:783–799, 2019. doi:10.1016/j.enconman.2019.02.018.

[35] Jun Ma et al. Improving Air Quality Prediction Accuracy at Larger Temporal Resolutions using Deep Learning and Transfer Learning Techniques. *Atmospheric Environment*, 214:116885, 2019. doi:10.1016/j.atmosenv.2019.116885.

[36] Shorouq Al-Eidi et al. Comparative Analysis Study for Air Quality Prediction in Smart Cities Using Regression Techniques. *IEEE Access*, 11:115140–115149, 2023. doi:10.1109/ACCESS.2023.3323447. doi:10.1109/ACCESS.2023.3323447.

[37] Jun Zhang and Kim-Fung Man. Time series prediction using rnn in multi-dimension embedding phase space. In *1998 IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1868–1873, San Diego, CA,

USA, 1998. doi:10.1109/ICSMC.1998.728168. doi:10.1109/ICSMC.1998.728168.

[38] Yuxiu Hua et al. Deep Learning with Long Short-Term Memory for Time Series Prediction. *IEEE Communications Magazine*, 57(6):114–119, 2019. doi:10.1109/MCOM.2019.1800155. doi:10.1109/MCOM.2019.1800155.

[39] Naiju Zhai, Peifu Yao, and Xiaofeng Zhou. Multivariate Time Series Forecast in Industrial Process Based on XGBoost and GRU. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 1397–1400, Chongqing, China, 2020. doi:10.1109/ITAIC49862.2020.9338878. doi:10.1109/ITAIC49862.2020.9338878.

[40] Farzaneh Mohammadi et al. Prediction of atmospheric PM2. 5 level by machine learning techniques in Isfahan, Iran. *Scientific Reports*, 14(2109), 2024. doi:10.1038/s41598-024-52617-z. doi:10.1038/s41598-024-52617-z.

[41] Kankamon Thaweephol and Nuwee Wiwatwattana. Long Short-Term Memory Deep Neural Network Model for PM2.5 Forecasting in the Bangkok Urban Area. In *2019 17th International Conference on ICT and Knowledge Engineering (ICT&KE)*, pages 1–6, Bangkok, Thailand, 2019. doi:10.1109/ICTKE47035.2019.8966854. doi:10.1109/ICTKE47035.2019.8966854.

[42] Fahad Radhi Alharbi and Denes Csala. A Seasonal Autoregressive Integrated Moving Average with Exogenous Factors (SARIMAX) Forecasting Model-Based Time Series Approach. *Inventions*, 7(4), 2022. doi:10.3390/inventions7040094. doi:10.3390/inventions7040094.

[43] Yue-Shan Chang et al. An LSTM-based aggregated model for air pollution forecasting. *Atmospheric Pollution Research*, 11(8):1451–1463, 2020. doi:10.1016/j.apr.2020.05.015. doi:10.1016/j.apr.2020.05.015.

[44] A. Usha Ruby et al. Forecasting PM2.5 Concentration Using Gradient-Boosted Regression Tree with CNN Learning Model. *Optical Memory and Neural Networks*, 33(1):86–96, 2024. doi:10.3103/s1060992x24010107. doi:10.3103/s1060992x24010107.

[45] Aorong Luo et al. Application of accurate online support vector regression in atmospheric SO2 concentration prediction. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 6274–6279, Shenyang, China, 2018. doi:10.1109/CCDC.2018.8408231. doi:10.1109/CCDC.2018.8408231.

[46] Sang Won Choi and Brian H. S. Kim. Applying PCA to Deep Learning Forecasting Models for Predicting PM2.5. *Sustainability*, 13(7), 2021. doi:10.3390/su13073726. doi:10.3390/su13073726.

[47] Markus Ringnér. What is principal component analysis? *Nature Biotechnology*, 26(3):303–304, 2008. doi:10.1038/nbt0308-303. doi:10.1038/nbt0308-303.

[48] Chen Ding et al. A hybrid CNN-LSTM model for predicting PM2.5 in Beijing based on spatiotemporal correlation. *Environmental and Ecological Statistics*, 2021. doi:10.1007/s10651-021-00501-8. doi:10.1007/s10651-021-00501-8.

[49] Qing Tao et al. Air Pollution Forecasting Using a Deep Learning Model Based on 1D Convnets and Bidirectional GRU. *IEEE Access*, 7:76690–76698, 2019. doi:10.1109/ACCESS.2019.2921578. doi:10.1109/ACCESS.2019.2921578.

[50] Yong-been Kim et al. Comparison of PM2.5 prediction performance of the three deep learning models: A case study of Seoul, Daejeon, and Bu-

san. *Journal of Industrial and Engineering Chemistry*, 120:159–169, 2023. doi:10.1016/j.jiec.2022.12.022. doi:10.1016/j.jiec.2022.12.022.

[51] Beytullah Eren, İpek Aksangür, and Caner Erden. Predicting next hour fine particulate matter (PM2.5) in the Istanbul Metropolitan City using deep learning algorithms with time windowing strategy. *Urban Climate*, 48:101418, 2023. doi:10.1016/j.uclim.2023.101418. doi:10.1016/j.uclim.2023.101418.

[52] Patricio Perez et al. Forecasting of hourly pm2. 5 in south-west zone in santiago de chile. *Aerosol and Air Quality Research*, 18(10):2666–2679, 2018.

[53] Ho Chang-Hoi et al. Development of a PM2.5 Prediction Model using a Recurrent Neural Network Algorithm for the Seoul Metropolitan Area, Republic of Korea. *Atmospheric Environment*, 245:118021, 2021. doi:10.1016/j.atmosenv.2020.118021.

[54] Endah Kristiani et al. Short-Term Prediction of PM2.5 Using LSTM Deep Learning Methods. *Sustainability*, 14(4):2068, 2022. doi:10.3390/su14042068.

[55] Yinan Chen, Shichong Yang, and Qiyuan Wang. Prediction of PM2.5 Concentration in Guangzhou based on LSTM Neural Network. In *2021 2nd International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI)*, pages 8–12, Shenyang, China, 2021. doi:10.1109/ICHCI54629.2021.00009.

[56] Temesegan Walelign Ayele and Rutvik Mehta. Air Pollution Monitoring and Prediction using IoT. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1741–1745, Coimbatore, India, 2018. doi:10.1109/ICICCT.2018.8473272.

[57] Mingmin Zhang, Dihua Wu, and Rongna Xue. Hourly Prediction of PM2.5 Concentration in Beijing based on Bi-LSTM Neural Network. *Multime-*

dia Tools and Applications, 80(16):24455–24468, 2021. doi:10.1007/s11042-021-10852-w.

[58] Xinxing Zhou et al. Air Pollutant Concentration Prediction Based on GRU Method. *Journal of Physics: Conference Series*, 1168(3):032058, 2019. doi:10.1088/1742-6596/1168/3/032058.

[59] Mingying Zhu and Jie Xie. Investigation of nearby monitoring station for hourly PM2.5 forecasting using parallel multi-input 1D-CNN-biLSTM. *Expert Systems with Applications*, 211:118707, 2023. doi:10.1016/j.eswa.2022.118707. doi:10.1016/j.eswa.2022.118707.

[60] Arko Datta et al. Efficient Air Quality Index Prediction on Resource-Constrained Devices using TinyML: Design, Implementation, and Evaluation. In *25th International Conference on Distributed Computing and Networking*, pages 304–309, Chennai, India, 2024. doi:10.1145/3631461.3631956. doi:10.1145/3631461.3631956.

[61] I Nyoman Kusuma Wardana, Suhaib A. Fahmy, and Julian W. Gardner. TinyML Models for a Low-Cost Air Quality Monitoring Device. *IEEE Sensors Letters*, 7(11):1–4, 2023. doi:10.1109/LSENS.2023.3315249. doi:10.1109/LSENS.2023.3315249.

[62] Armin Mazinani et al. Air Quality Estimation with Embedded AI-Based Prediction Algorithms. In *2023 International Conference on Information Technology Research and Innovation (ICITRI)*, pages 87–92, Jakarta, Indonesia, 2023. doi:10.1109/ICITRI59340.2023.10249864. doi:10.1109/ICITRI59340.2023.10249864.

[63] Luca Davoli, Laura Belli, and Gianluigi Ferrari. Air quality dataset from an indoor airport travelers transit area. *Data in Brief*, 52:109821, 2024. doi:10.1016/j.dib.2023.109821. doi:10.1016/j.dib.2023.109821.

[64] Jacob Benesty et al. *Pearson Correlation Coefficient*, pages 1–4. Springer, 2009. doi:10.1007/978-3-642-00296-0_5. doi:10.1007/978-3-642-00296-0_5.

[65] scikit learn. MinMaxScaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. Accessed on October 11, 2024.

[66] Jia Wu et al. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019. doi:10.11989/JEST.1674-862X.80904120. doi:10.11989/JEST.1674-862X.80904120.

[67] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[68] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[69] Danilo Pau, Andrea Pisani, and Antonio Candelieri. Towards full forward on-tiny-device learning: A guided search for a randomly initialized neural network. *Algorithms*, 17(1):22, 2024.

[70] Jeroen van Hoof and Joaquin Vanschoren. Hyperboost: Hyperparameter optimization by gradient boosting surrogate models. *arXiv preprint arXiv:2101.02289*, 2021.

[71] Luca Davoli, Laura Belli, and Gianluigi Ferrari. Indoor Air Quality Monitoring @ Brindisi Airport, 2023. doi:10.17632/BV2HVM4PMZ. doi:10.17632/BV2HVM4PMZ.

[72] STMicroelectronics. Discovery kit with STM32F469NI MCU. https://www.st.com/en/evaluation-tools/32f469idiscovery.html. Accessed on October 11, 2024.

[73] STMicroelectronics. STM32 F469I-DISCOVERY KIT, 2024. https://www.st.com/en/evaluation-tools/32f469idiscovery.html. Accessed on March 12, 2024.

[74] STMicroelectronics. ST Edge AI Developer Cloud, 2024. https://stm32ai-cs.st.com/home. Accessed on August 29, 2024.

[75] Google AI Edge. LiteRT – Post-training quantization. https://ai.google.dev/edge/litert/models/post_training_quantization. Accessed on October 11, 2024.

[76] UCI ML Repository. https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data. Accessed on July 9, 2023.

[77] KNNImputer Python library. https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html. Accessed on July 9, 2023.

[78] Keras. https://keras.io/. Accessed on July 9, 2023.

[79] TensorFlow. https://www.tensorflow.org/. Accessed on July 9, 2023.

[80] scikit-learn. https://scikit-learn.org/. Accessed on July 9, 2023.

[81] Jenny Cifuentes et al. Air Temperature Forecasting Using Machine Learning Techniques: A Review. *Energies*, 13(16):1–28, 2020. doi:10.3390/en13164215.

[82] Xia Hu et al. Model Complexity of Deep Learning: A Survey. *Knowledge and Information Systems*, 63(10), 2021. doi:10.1007/s10115-021-01605-0.

[83] STMicroelectronics. B-L475E-IOT01A – STM32L4 Discovery kit IoT node. https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html. Accessed on July 9, 2023.

[84] Mingsong Lv and Enyu Xu. Deep Learning on Energy Harvesting IoT Devices: Survey and Future Challenges. *IEEE Access*, 10:124999–125014, 2022. doi:10.1109/ACCESS.2022.3225092.

[85] Ioannis C. Lampropoulos et al. Worldwide trends in mortality related to Parkinson's disease in the period of 1994-2019: Analysis of vital registration data from the WHO Mortality Database. *Frontiers in Neurology*, 13:956440, 2022. doi:10.3389/fneur.2022.956440. doi:10.3389/fneur.2022.956440.

[86] E. R. Dorsey et al. Projected Number of People with Parkinson Disease in the Most Populous Nations, 2005 through 2030. *Neurology*, 68(5):384–386, 2007. doi:10.1212/01.wnl.0000247740.47667.03. doi:10.1212/01.wnl.0000247740.47667.03.

[87] Margaret T. M. Prenger et al. Social Symptoms of Parkinson's Disease. *Parkinson's Disease*, 2020:1–10, 12 2020. doi:10.1155/2020/8846544. doi:10.1155/2020/8846544.

[88] R. Balestrino and A.H.V. Schapira. Parkinson Disease. *European Journal of Neurology*, 27(1):27–42, 2020. doi:10.1111/ene.14108. doi:10.1111/ene.14108.

[89] Honglei Chen and Beate Ritz. The Search for Environmental Causes of Parkinson's Disease: Moving Forward. *Journal of Parkinson's Disease*, 8(s1):S9–S17, 12 2018. doi:10.3233/jpd-181493. doi:10.3233/jpd-181493.

[90] Alberto J. Espay et al. Technology in Parkinson's disease: Challenges and opportunities. *Movement Disorders*, 31(9):1272–1282, 2016. doi:10.1002/mds.26642. doi:10.1002/mds.26642.

[91] Federico Parisi et al. Inertial BSN-Based Characterization and Automatic UPDRS Evaluation of the Gait Task of Parkinsonians. *IEEE Transactions on Affective Computing*, 7(3):258–271, 2016. doi:10.1109/TAFFC.2016.2549533. doi:10.1109/TAFFC.2016.2549533.

[92] Shivangi, Anubhav Johri, and Ashish Tripathi. Parkinson Disease Detection Using Deep Neural Networks. In *2019 Twelfth International Conference on Contemporary Computing (IC3)*, pages 1–4, Noida, India, 2019. doi:10.1109/IC3.2019.8844941. doi:10.1109/IC3.2019.8844941.

[93] Aditi Govindu and Sushila Palwe. Early Detection of Parkinson's Disease using Machine Learning. *Procedia Computer Science*, 218:249–261, 2023. doi:10.1016/j.procs.2023.01.007. doi:10.1016/j.procs.2023.01.007.

[94] Federico Parisi et al. Body-Sensor-Network-Based Kinematic Characterization and Comparative Outlook of UPDRS Scoring in Leg Agility, Sit-to-Stand, and Gait Tasks in Parkinson's Disease. *IEEE Journal of Biomedical and Health Informatics*, 19(6):1777–1793, 2015. doi:10.1109/JBHI.2015.2472640. doi:10.1109/JBHI.2015.2472640.

[95] Atiqur Rahman, Aurangzeb Khan, and Arsalan Ali Raza. Parkinson's Disease Detection Based on Signal Processing Algorithms and Machine Learning. *CRPASE: Transactions of Electrical, Electronic and Computer Engineering*, 6(3):141–145, 2020.

[96] Changqin Quan et al. End-to-End Deep Learning Approach for Parkinson's Disease Detection from Speech Signals. *Biocybernetics and Biomedical Engineering*, 42(2):556–574, 2022. doi:10.1016/j.bbe.2022.04.002. doi:10.1016/j.bbe.2022.04.002.

[97] Kevin Saltos et al. Detecting Parkinson's Disease with Convolutional Neural Networks: Voice Analysis and Deep Learning. In *Information and Communication Technologies*, pages 324–336, 2023. doi:10.1007/978-3-031-45438-7_22. doi:10.1007/978-3-031-45438-7_22.

[98] Changqin Quan, Kang Ren, and Zhiwei Luo. A Deep Learning Based Method for Parkinson's Disease Detection Using Dynamic Features of Speech. *IEEE*

Access, 9:10239–10252, 2021. doi:10.1109/ACCESS.2021.3051432. doi:10.1109/ACCESS.2021.3051432.

[99] Giovanni Dimauro and Francesco Girardi. Italian Parkinson's Voice and Speech, 2019. doi:10.21227/AW6B-TG17. doi:10.21227/AW6B-TG17.

[100] Giovanni Dimauro et al. Assessment of Speech Intelligibility in Parkinson's Disease Using a Speech-To-Text System. *IEEE Access*, 5:22199–22208, 2017. doi:10.1109/ACCESS.2017.2762475. doi:10.1109/ACCESS.2017.2762475.

[101] Zrar Kh. Abdul and Abdulbasit K. Al-Talabani. Mel Frequency Cepstral Coefficient and its Applications: A Review. *IEEE Access*, 10:122136–122158, 2022. doi:10.1109/ACCESS.2022.3223444. doi:10.1109/ACCESS.2022.3223444.

[102] N. V. Chawla et al. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357, 6 2002. doi:10.1613/jair.953. doi:10.1613/jair.953.

[103] Benoit Jacob et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *arXiv preprint arXiv:1712.05877*, 2017. doi:10.48550/arXiv.1712.05877. doi:10.48550/arXiv.1712.05877.

[104] David M. W. Powers. Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness and Correlation. *arXiv preprint arXiv:2010.16061*, 2020. doi:10.48550/arXiv.2010.16061. doi:10.48550/arXiv.2010.16061.

[105] Ghalib H Alshammri et al. IoT-Based Voice-Controlled Smart Homes with Source Separation Based on Deep Learning. *Journal of Sensors*, 2023:1–18, 2023. doi:10.1155/2023/1911385. doi:10.1155/2023/1911385.

[106] Shinji Watanabe et al. Introduction to the Issue on Far-Field Speech Processing in the Era of Deep Learning: Speech Enhancement, Separation, and Recognition. *IEEE Journal on Selected Topics in Signal Processing*, 13(4):785–786,

2019. doi:10.1109/JSTSP.2019.2925640. doi:10.1109/JSTSP.2019.2925640.

[107] S Boll. A Spectral Subtraction Algorithm for Suppression of Acoustic Noise in Speech. In *1979 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 200–203, Washington, DC, USA, 1979. doi:10.1109/ICASSP.1979.1170696. doi:10.1109/ICASSP.1979.1170696.

[108] Pascal Scalart et al. Speech Enhancement based on a Priori Signal to Noise Estimation. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 629–632, Atlanta, GA, USA, 1996. doi:10.1109/ICASSP.1996.543199. doi:10.1109/ICASSP.1996.543199.

[109] Yariv Ephraim and Harry L Van Trees. A Signal Subspace Approach for Speech Enhancement. *IEEE Transactions on Speech and Audio Processing*, 3(4):251–266, 1995. doi:10.1109/89.397090. doi:10.1109/89.397090.

[110] Guillaume Carbajal et al. Joint NN-Supported Multichannel Reduction of Acoustic Echo, Reverberation and Noise. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2158–2173, 2020. doi:10.1109/TASLP.2020.3008974. doi:10.1109/TASLP.2020.3008974.

[111] Chengshi Zheng et al. Sixty Years of Frequency-Domain Monaural Speech Enhancement: From Traditional to Deep Learning Methods. *Trends in Hearing*, 27:1–52, 2023. doi:10.1177/23312165231209913. doi:10.1177/23312165231209913.

[112] Vijay Janapa Reddi et al. MLPerf Inference Benchmark. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 446–459, Valencia, Spain, 2020. doi:10.1109/ISCA45697.2020.00045. doi:10.1109/ISCA45697.2020.00045.

[113] Hyeong-Seok Choi et al. Real-Time Denoising and Dereverberation wtih Tiny Recurrent U-Net. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5789–5793, Toronto, ON, Canada, 2021. doi:10.1109/ICASSP39728.2021.9414852. doi:10.1109/ICASSP39728.2021.9414852.

[114] Vincent Lostanlen et al. Per-Channel Energy Normalization: Why and How. *IEEE Signal Processing Letters*, 26(1):39–43, 2019. doi:10.1109/LSP.2018.2878620. doi:10.1109/LSP.2018.2878620.

[115] Emad M Grais and Mark D Plumbley. Single Channel Audio Source Separation using Convolutional Denoising Autoencoders. In *2017 IEEE global conference on signal and information processing (GlobalSIP)*, pages 1265–1269, Montreal, QC, Canada, 2017. doi:10.1109/GlobalSIP.2017.8309164. doi:10.1109/GlobalSIP.2017.8309164.

[116] Lovedeep Gondara. Medical Image Denoising Using Convolutional Denoising Autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, Barcelona, Spain, 2016. doi:10.1109/icdmw.2016.0041. doi:10.1109/icdmw.2016.0041.

[117] Deeksha, Sandeep Yadav, and K. Sreelakshmi. Implementation of Audio De-noising using Convolutional Neural Networks on Edge Device. In *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, pages 1–5, Bangalore, India, 2021. doi:10.1109/CSITSS54238.2021.9683183. doi:10.1109/CSITSS54238.2021.9683183.

[118] Jonathan Le Roux et al. SDR – Half-baked or Well Done? In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 626–630, Brighton, UK, 2019. doi:10.1109/ICASSP.2019.8683855. doi:10.1109/ICASSP.2019.8683855.

[119] Igor Fedorov et al. TinyLSTMs: Efficient Neural Speech Enhancement for Hearing Aids. In *Proc. Interspeech 2020*, pages 4054–4058, 2020. doi:10.21437/Interspeech.2020-1864. doi:10.21437/Interspeech.2020-1864.

[120] STMicroelectronics. NUCLEO-G474RE – STM32 Nucleo-64 development board with STM32G474RE MCU. https://www.st.com/en/evaluation-tools/nucleo-g474re.html. Accessed on October 21, 2023.

[121] Shiqiang Zhu et al. Intelligent Computing: The Latest Advances, Challenges, and Future. *Intelligent Computing*, 2, Jan 2023. doi:10.34133/icomputing.0006. doi:10.34133/icomputing.0006.

[122] Charmi Jobanputra, Jatna Bavishi, and Nishant Doshi. Human Activity Recognition: A Survey. *Procedia Computer Science*, 155:698–703, 2019. doi:10.1016/j.procs.2019.08.100. doi:10.1016/j.procs.2019.08.100.

[123] Florenc Demrozi et al. Human Activity Recognition Using Inertial, Physiological and Environmental Sensors: A Comprehensive Survey. *IEEE Access*, 8:210816–210836, 2020. doi:10.1109/ACCESS.2020.3037715. doi:10.1109/ACCESS.2020.3037715.

[124] Jun Qi et al. An Overview of Data Fusion Techniques for Internet of Things Enabled Physical Activity Recognition and Measure. *Information Fusion*, 55:269–280, 2020. doi:10.1016/j.inffus.2019.09.002. doi:10.1016/j.inffus.2019.09.002.

[125] Kaixuan Chen et al. Deep Learning for Sensor-Based Human Activity Recognition: Overview, Challenges, and Opportunities. *ACM Comput. Surv.*, 54(4), May 2021. doi:10.1145/3447744. doi:10.1145/3447744.

[126] Saurabh Gupta. Deep Learning Based Human Activity Recognition (HAR) Using Wearable Sensor Data. *International Journal of Information Man-*

agement Data Insights, 1(2):100046, 2021. 10.1016/j.jjimei.2021.100046. doi:10.1016/j.jjimei.2021.100046.

[127] Iqbal H. Sarker. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3), Mar 2021. doi:10.1007/s42979-021-00592-x. doi:10.1007/s42979-021-00592-x.

[128] Pedro Lara-Benítez, Manuel Carranza-García, and José C. Riquelme. An Experimental Review on Deep Learning Architectures for Time Series Forecasting. *International Journal of Neural Systems*, 31(03):2130001, Feb 2021. doi:10.1142/s0129065721300011. doi:10.1142/s0129065721300011.

[129] Frédéric Li et al. Deep Transfer Learning for Time Series Data Based on Sensor Modality Classification. *Sensors*, 20(15):4271, Jul 2020. doi:10.3390/s20154271. doi:10.3390/s20154271.

[130] Ankita Jain and Vivek Kanhangad. Human Activity Classification in Smartphones Using Accelerometer and Gyroscope Sensors. *IEEE Sensors Journal*, 18(3):1169–1177, 2018. doi:10.1109/JSEN.2017.2782492. doi:10.1109/JSEN.2017.2782492.

[131] Francisco Ordóñez and Daniel Roggen. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors*, 16(1):115, Jan 2016. doi:10.3390/s16010115. doi:10.3390/s16010115.

[132] Shilong Yu and Long Qin. Human Activity Recognition with Smartphone Inertial Sensors Using Bidir-LSTM Networks. In *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 219–224, Huhhot, China, 2018. doi:10.1109/ICMCCE.2018.00052. doi:10.1109/ICMCCE.2018.00052.

[133] Mohib Ullah et al. Stacked Lstm Network for Human Activity Recognition Using Smartphone Data. In *2019 8th European Workshop on Visual Information Processing (EUVIP)*, pages 175–180, Rome, Italy, 2019. doi:10.1109/EUVIP47703.2019.8946180. doi:10.1109/EUVIP47703.2019.8946180.

[134] Chamani Shiranthika et al. Human Activity Recognition Using CNN & LSTM. In *2020 5th International Conference on Information Technology Research (ICITR)*, pages 1–6, Moratuwa, Sri Lanka, 2020. doi:10.1109/ICITR51448.2020.9310792. doi:10.1109/ICITR51448.2020.9310792.

[135] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, London, England, nov 2016.

[136] StandardScaler Python library. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed on October 21, 2023.

[137] HamidReza Naseri and Vahid Mehrdad. Novel CNN with Investigation on Accuracy by Modifying Stride, Padding, Kernel Size and Filter Numbers. *Multimedia Tools and Applications*, 82(15):23673–23691, Feb 2023. doi:10.1007/s11042-023-14603-x. doi:10.1007/s11042-023-14603-x.

[138] Rajat Rathore et al. Real-World Model for Bitcoin Price Prediction. *Information Processing & Management*, 59(4):102968, 2022. doi:10.1016/j.ipm.2022.102968.

[139] CoinMarketCap. Global Cryptocurrency Charts – Total Market Cap, 2024. https://coinmarketcap.com/charts/. Accessed on August 29, 2024.

[140] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. https://bitcoin.org/bitcoin.pdf. Accessed on August 29, 2024.

[141] Sheetal Singh et al. Securing Blockchain Transactions Using Quantum Teleportation and Quantum Digital Signature. *Neural Processing Letters*, 55(4):3827–3842, 2020. doi:10.1007/s11063-020-10272-1.

[142] Yijun Zou et al. Focus on Blockchain: A Comprehensive Survey on Academic and Application. *IEEE Access*, 8:187182–187201, 2020. doi:10.1109/ACCESS.2020.3030491.

[143] Parikshit Joshi et al. Blockchain Technology for Sustainable Development: A Systematic Literature Review. *Journal of Global Operations and Strategic Sourcing*, 16(3):683–717, 2023. doi:10.1108/JGOSS-06-2022-0054.

[144] Alessandra Rizzardi et al. IoT-driven Blockchain to Manage the Healthcare Supply Chain and Protect Medical Records. *Future Generation Computer Systems*, 161:415–431, 2024. doi:10.1016/j.future.2024.07.039.

[145] Pratima Sharma et al. EHDHE: Enhancing Security of Healthcare Documents in IoT-enabled Digital Healthcare Ecosystems using Blockchain. *Information Sciences*, 629:703–718, 2023. doi:10.1016/j.ins.2023.01.148.

[146] Douglas LL Moura, Andre LL Aquino, and Antonio AF Loureiro. An Edge Computing and Distributed Ledger Technology Architecture for Secure and Efficient Transportation. *Ad Hoc Networks*, 164:103633, 2024. doi:10.1016/j.adhoc.2024.103633.

[147] Tanishq Soni et al. Tracing Food Products in Supply Chain Using DLT Enabled Blockchain Technology. In *2023 International Conference on Electrical, Electronics, Communication and Computers (ELEXCOM)*, pages 1–6, Roorkee, India, 2023. IEEE. doi:10.1109/ELEXCOM58812.2023.10370526.

[148] Mohammed Alshehri. Blockchain-assisted Internet of Things Framework in Smart Livestock Farming. *Internet of Things*, 22:100739, 2023. doi:10.1016/j.iot.2023.100739.

[149] Anum Nawaz et al. Hyperledger Fabric in Precision Agriculture: A Study on Data Integrity and Availability. In *2024 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–8, Girona, Spain, 2024. IEEE. doi:10.1109/CITS61189.2024.10608019.

[150] Obryan Poyser. Exploring the Determinants of Bitcoin's Price: An Application of Bayesian Structural Time Series. *arXiv preprint arXiv:1706.01437*, 1(1):1–47, 2017. doi:10.48550/arXiv.1706.01437.

[151] Hongling Xu et al. Two-Stage Prediction of Machinery Fault Trend Based on Deep Learning for Time Series Analysis. *Digital Signal Processing*, 117:103150, 2021. doi:10.1016/j.dsp.2021.103150.

[152] Zahra Karevan and Johan AK Suykens. Transductive LSTM for Time-Series Prediction: An Application to Weather Forecasting. *Neural Networks*, 125:1–9, 2020. doi:10.1016/j.neunet.2019.12.030.

[153] Talha Burak Alakus and Ibrahim Turkoglu. Comparison of Deep Learning Approaches to Predict COVID-19 Infection. *Chaos, Solitons & Fractals*, 140:110120, 2020. doi:10.1016/j.chaos.2020.110120.

[154] Tania Cerquitelli et al. Machine Learning Empowered Computer Networks. *Computer Networks*, 230:109807, 2023. doi:10.1016/j.comnet.2023.109807.

[155] Ibrahim M Awad, Ghada K Al-Jerashi, and Zaid Ahmad Alabaddi. Determinants of Private Domestic Investment in Palestine: Time Series Analysis. *Journal of Business and Socio-economic Development*, 1(1):71–86, 2021. doi:10.1108/JBSED-04-2021-0038.

[156] Hongju Yan and Hongbing Ouyang. Financial Time Series Prediction Based on Deep Learning. *Wireless Personal Communications*, 102(2):683–700, 2018. doi:10.1007/s11277-017-5086-2.

[157] Terence C Mills. *Time Series Techniques for Economists*. Cambridge University Press, Cambridge, UK, 1990. ISBN 978-0-521-34339-8.

[158] Mehdi Khashei and Mehdi Bijari. A Novel Hybridization of Artificial Neural Networks and ARIMA Models for Time Series Forecasting. *Applied Soft Computing*, 11(2):2664–2675, 2011. doi:10.1016/j.asoc.2010.10.015.

[159] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 1(9):1–15, 2014. doi:10.48550/arXiv.1412.6980.

[160] Georgios Fragkos et al. Artificially Intelligent Electronic Money. *IEEE Consumer Electronics Magazine*, 10(4):81–89, 2020. doi:10.1109/MCE.2020.3024512.

[161] Arief Radityo, Qorib Munajat, and Indra Budi. Prediction of Bitcoin Exchange Rate to American Dollar Using Artificial Neural Network Methods. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 433–438, Bali, Indonesia, 2018. IEEE. doi:10.1109/ICACSIS.2017.8355070.

[162] D Olvera-Juarez and E Huerta-Manzanilla. Forecasting Bitcoin Pricing with Hybrid models: A Review of the Literature. *International Journal of Advanced Engineering Research and Science*, 6(9):161–164, 2019. doi:10.22161/ijaers.69.18.

[163] Salim Lahmiri and Stelios Bekiros. Cryptocurrency Forecasting with Deep Learning Chaotic Neural Networks. *Chaos, Solitons & Fractals*, 118:35–40, 2019. doi:10.1016/j.chaos.2018.11.014.

[164] P. Nithyakani et al. Prediction of Bitcoin Price Using Bi-LSTM Network. In *2021 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5, Coimbatore, India, 2021. IEEE. doi:10.1109/ICCCI50826.2021.9402427.

[165] Tonghui Li. Prediction of Bitcoin Price Based on LSTM. In *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, pages 19–23, Guangzhou, China, 2022. IEEE. doi:10.1109/MLISE57402.2022.00012.

[166] Phumudzo Lloyd Seabe, Claude Rodrigue Bambe Moutsinga, and Edson Pindza. Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach. *Fractal and Fractional*, 7(2):203, 2023. doi:10.3390/fractalfract7020203.

[167] Gyana Ranjan Patra and Mihir Narayan Mohanty. Price Prediction of Cryptocurrency using a Multi-Layer Gated Recurrent Unit Network with Multi Features. *Computational Economics*, 62(4):1525–1544, 2023. doi:10.1007/s10614-022-10310-1.

[168] Navmeen Latif et al. Comparative Performance of LSTM and ARIMA for the Short-Term Prediction of Bitcoin Prices. *Australasian Accounting, Business and Finance Journal*, 17(1):256–276, 2023. doi:10.14453/aabfj.v17i1.15.

[169] Joy Dip Das et al. Encoder–Decoder Based LSTM and GRU Architectures for Stocks and Cryptocurrency Prediction. *Journal of Risk and Financial Management*, 17(5):200, 2024. doi:10.3390/jrfm17050200.

[170] Rahmat Albariqi and Edi Winarko. Prediction of Bitcoin Price Change using Neural Networks. In *2020 International Conference on Smart Technology and Applications (ICoSTA)*, pages 1–4, Surabaya, Indonesia, 2020. IEEE. doi:10.1109/ICoSTA48221.2020.1570610936.

[171] Tashreef Muhammad et al. Transformer-based deep learning model for stock price prediction: A case study on bangladesh stock market. *International Journal of Computational Intelligence and Applications*, 22(03):2350013, 2023. doi:10.1142/S146902682350013X.

[172] Seyed Mehran Kazemi et al. Time2Vec: Learning a Vector Representation of Time. *arXiv preprint arXiv:1907.05321*, 1(1):1–16, 2019. doi:10.48550/arXiv.1907.05321.

[173] Chaojie Wang et al. Stock market index prediction using deep transformer model. *Expert Systems with Applications*, 208:118128, 2022. doi:10.1016/j.eswa.2022.118128.

[174] Twelve Data Pte. Ltd. Twelvedata platform, 2024. https://www.twelvedata.com. Accessed on August 29, 2024.

[175] Gaia Codeluppi, Luca Davoli, and Gianluigi Ferrari. Forecasting Air Temperature on Edge Devices with Embedded AI. *Sensors*, 21(12):1–29, 2021. doi:10.3390/s21123973.

[176] Gaia Codeluppi et al. AI at the Edge: a Smart Gateway for Greenhouse Air Temperature Forecasting. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 348–353, Trento, Italy, 2020. IEEE. doi:10.1109/MetroAgriFor50201.2020.9277553.

[177] Günter Klambauer et al. Self-Normalizing Neural Networks. *arXiv preprint arXiv:1706.02515*, 1(5):1–102, 2017. doi:10.48550/arXiv.1706.02515.

[178] Keras. Dropout Layer, 2024. https://keras.io/api/layers/regularization_layers/dropout/. Accessed on August 29, 2024.

[179] Keras. Layer Weight Regularizers, 2024. https://keras.io/api/layers/regularizers/. Accessed on August 29, 2024.

[180] Keras. EarlyStopping, 2024. https://keras.io/api/callbacks/early_stopping/. Accessed on August 29, 2024.

[181] scikit-learn developers. MinMaxScaler, 2024. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. Accessed on August 29, 2024.

[182] Sebastian Ruder. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747*, 1(2):1–14, 2016. doi:10.48550/arXiv.1609.04747.

[183] Davide Chicco, Matthijs J Warrens, and Giuseppe Jurman. The Coefficient of Determination R-squared is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation. *Peerj computer science*, 7:e623, 2021. doi:10.7717/peerj-cs.623.

Acknowledgments

UNIONE EUROPEA
Fondo Sociale Europeo
Ministero dell'Università e della Ricerca
PON RICERCA E INNOVAZIONE 2014 - 2020
REACT EU
UNIVERSITÀ DI PARMA

La borsa di dottorato è stata cofinanziata con risorse del
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020, risorse FSE REACT-EU
Azione IV.4 “Dottorati e contratti di ricerca su tematiche dell’innovazione”
e Azione IV.5 “Dottorati su tematiche Green”

UNIONE EUROPEA
Fondo Sociale Europeo

Ministero dell'Università e della Ricerca

PON RICERCA E INNOVAZIONE 2014 - 2020

REACT EU

UNIVERSITÀ DI PARMA

The doctoral scholarship has been co-funded with financial resources of the
Programma Operativo Nazionale Ricerca e Innovazione 2014-2020, risorse FSE REACT-EU
Azione IV.4 "Dottorati e contratti di ricerca su tematiche dell'innovazione"
e Azione IV.5 "Dottorati su tematiche Green"