



Universidad Autónoma de Querétaro

Facultad de Ingeniería

Licenciatura en Ingeniería Física

**DEEP LEARNING-BASED MODELLING FOR TIME SERIES**  
TESIS

Que como parte de los requisitos para obtener el grado de  
Ingeniero Físico

Presenta:

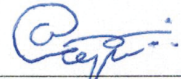
**Josue Becerra Rico**

Dirigido por:

**Dr. Marco Antonio Aceves Fernández**

SINODALES


Dr. Marco Antonio Aceves Fernández  
Presidente

  
Firma

Dr. Jesús Carlos Pedraza Ortega  
Secretario

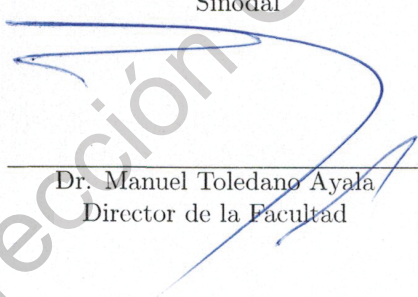
  
Firma

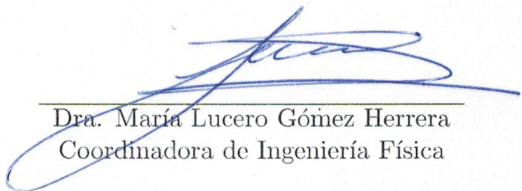
Dra. María Lucero Gómez Herrera  
Vocal

  
Firma

Dr. Alberto Hernández Almada  
Sinodal

  
Firma

  
Dr. Manuel Toledano Ayala  
Director de la Facultad

  
Dra. María Lucero Gómez Herrera  
Coordinadora de Ingeniería Física

Campus Aeropuerto  
Querétaro, QRO  
México.  
Septiembre 2019

Dirección General de Bibliotecas UAQ

© 2019 - Josue Becerra Rico

All rights reserved.

Dirección General de Bibliotecas UAQ

Dirección General de Bibliotecas UAQ



*This thesis is dedicated to my mother Kizzy, my family and my girlfriend Caro who always supported me to achieve my goals and helped me to become who I am.*

Dirección General de Bibliotecas UAQ

# Acknowledgments

I would like to thank the Autonomous University of Queretaro for providing the virtual library to complete this investigation. I am also grateful to the members of my committee for their patience and support in overcoming numerous obstacles I have been facing through my research. I would like to thank my classmates for their cooperation and of course friendship and support during the last 5 years.

Nevertheless, I am also grateful to the PhD Marco Aceves Fernandez for his patience, unconditional support and guidance provided to me in this project.

Dirección General de Bibliotecas UAQ

# Abstract

Developments in deep learning for time-series problems have shown promising for modeling static data. In this work a comparison between two algorithms will be presented to model non-linear time series with Deep Learning techniques, using a Recurrent Neural Network (RNN) which is a structure for learning long and short-term dependencies "Long Short Term Memory (LSTM) and the Gated Recurrent Unit (GRU) network (that is a variation of the RNN and Long Short-Term Memory networks) compared to traditional Long Short-Term Memory (LSTM), requires fewer parameters and less computation time in processing.

In this contribution GRU and LSTM are used to tackle PM10 particles, due to their non-linear behavior, a forecasting model can be generated by the use of artificial intelligence techniques.

**Keywords:** Artificial Intelligence (AI), Artificial Neural Networks(ANN); Recurrent Neural Networks(RNN); Deep Neural Networks; Time Series; Air Pollution Modelling; PM10 Particles.

Dirección General de Bibliotecas UAQ

# Resumen

Los desarrollos en aprendizaje profundo para problemas de series de tiempo se han mostrado prometedores para modelar datos estáticos. En este trabajo se presentará la comparación entre dos algoritmos de redes neuronales recurrentes (RNN) capaces de modelar series de tiempo no lineales con técnicas de aprendizaje profundo, utilizando el algoritmo Long Short-Term Memory (LSTM) que es una estructura para el aprendizaje de dependencias a corto y largo plazo. En contraste con Gated Recurrent Unit (GRU) (que es una variación de las redes RNN y Long Short-Term Memory) en comparación con la Long Short-Term Memory (LSTM) tradicional, requiere menos parámetros y menos tiempo de cálculo en el procesamiento.

En este trabajo, las estructuras GRU y LSTM se utilizan para abordar partículas PM10, debido a su comportamiento no lineal, se puede generar un modelo de predicción con el uso de técnicas de inteligencia artificial.

**Palabras clave:** Inteligencia Artificial (AI); Redes Neuronales Artificiales (ANN); Redes Neuronales Recurrentes (RNN); Redes Neuronales Profundas; Series de Tiempo; Modelamiento de Contaminación Ambiental; Partículas PM10.



Dirección General de Bibliotecas UAQ

# Contents

<b>Acknowledgments</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Resumen</b>	<b>xi</b>
<b>Contents</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Formulation . . . . .	1
1.3 Objectives . . . . .	2
1.3.1 Specific Objectives . . . . .	2
1.4 Hypothesis . . . . .	2
1.5 Thesis Structure . . . . .	2
<b>2 Literature Survey</b>	<b>5</b>
2.1 Airborne Pollution . . . . .	5
2.2 Air Quality and Weather Monitoring . . . . .	6
2.3 Artificial Neural Networks . . . . .	7
2.3.1 Artificial Intelligence, Machine Learning and Deep Learning . . . . .	8
2.4 Types of Machine Learning . . . . .	8
2.4.1 Supervised Learning . . . . .	8
2.4.2 Unsupervised Learning . . . . .	9
2.4.3 Self-supervised Learning . . . . .	9
2.4.4 Reinforcement Learning . . . . .	10
2.5 Neural Network Architectures . . . . .	10
2.5.1 Recurrent Neural Networks . . . . .	10
2.5.2 Long Short Term Memory . . . . .	11
2.5.3 Gated Recurrent Unit . . . . .	13
2.6 Machine Learning Applications to Physics . . . . .	14

2.6.1	Schrödinger Equation . . . . .	14
2.6.2	Image Analysis for Morphological Galaxy Classification . . . . .	16
2.6.3	Molecular Dynamics . . . . .	17
2.6.4	Boltzmann Machines and Thermodynamics . . . . .	18
<b>3</b>	<b>Methodology</b>	<b>21</b>
3.1	Requirements . . . . .	21
3.1.1	System Requirements . . . . .	21
3.1.2	Python Libraries . . . . .	21
3.2	Design . . . . .	22
3.2.1	Hardware System . . . . .	22
3.2.2	Software Development Design . . . . .	24
3.3	Data . . . . .	27
3.4	Implementation . . . . .	28
<b>4</b>	<b>Results and Discussion</b>	<b>31</b>
4.1	Results . . . . .	31
4.1.1	Long Short Term Memory . . . . .	31
4.1.2	Gated Recurrent Unit . . . . .	35
4.1.3	Error Estimation . . . . .	38
4.2	Discussion . . . . .	42
4.3	Significance/Impact . . . . .	42
4.4	Future Work . . . . .	42
<b>5</b>	<b>Conclusion</b>	<b>43</b>
	<b>References</b>	<b>47</b>
.1	Appendix . . . . .	48
.1.1	Hadamard Product . . . . .	48
.1.2	Activation Functions . . . . .	48

# List of Figures

2.1	Contribution of PM10 pollution emitters in Mexico City (data obtained from <i>Red Automática de Monitoreo Ambiental</i> [SEDEMA, 2018]) . . . . .	5
2.2	RAMA monitoring sites ([Aguirre-Salado et al., 2017]) . . . . .	6
2.3	Artificial Neuron [Engelbrecht, 2007] . . . . .	7
2.4	Artificial Intelligence, Machine Learning and Deep Learning [Engelbrecht, 2007] . . . . .	8
2.5	(a) Using only curiosity-driven exploration, the agent makes significant progress in Level-1. (b) The gained knowledge helps the agent explore subsequent levels much faster than when starting from scratch. [Pathak et al., 2017] . . . . .	10
2.6	Recurrent Neural Network model [Hochreiter and Schmidhuber, 1997]. . . . .	11
2.7	interpretation of a recurrent neural as a conventional neural network [Hochreiter and Schmidhuber, 1997]	
2.8	The repeating module in an LSTM contains four interacting layers [Zhou et al., 2016]. . . . .	12
2.9	Gated Recurrent Unit Model [Zhou et al., 2016]. . . . .	13
2.10	Neural network used to find ground state wave functions [Luo and Clark, 2019]. . . . .	15
2.11	Spin-up (top) and spin-down (bottom) single particle orbital(s.p.o) for $\psi_{S0}$ (left) and $\psi_{SN}$ (right) with 256 hidden neurons on 4x4 Hubbard model at $U/t=8$ , $n=0.875$ , where the s.p.o are evaluated at the (1d-reshaped) spin-configuration shown. For the s.p.o, row is orbital index and the column is position index [Luo and Clark, 2019].	15
2.12	Weights between the input layer and hidden layer for $\psi_{SN}$ with 256 hidden neurons for the spin-up (left) and spin-down (right) neural networks on 4x4 Hubbard model at $U/t=8$ , $n=0.875$ . Hidden neurons are ordered by their bias and shown are neurons 1-32 (top), 96-128 (middle) and 224-256 (bottom). [Luo and Clark, 2019]. . . . .	16
2.13	Elements of the classification method proposed by [De La Calleja and Fuentes, 2004]. . . . .	17
2.14	Comparison of the bulk Si phonon spectrum calculated with DFTB (blue), and SW (black) and with the ML on-the-fly (MLOTF) approach (red), computed with the finite displacement Parlinski-Li-Kawazoe method using a standard $\sigma_{err} = 0.055 \times eV/\text{\AA}$ (dotted lines) and a high-accuracy $\sigma_{err} = 5 \times 10^{-4} eV/\text{\AA}$ value (solid lines) for the ML data noise parameter. The ML database was constructed from a 300 K MD trajectory [Payne et al., 1992]. . . . .	18

2.15	Restricted Boltzmann machine. The visible nodes (blue) are connected to the hidden nodes (red) with a symmetric matrix of weight $W$ . The external fields in the Hamiltonian are represented by new edges with weights $b$ and $c$ connecting the visible and hidden nodes, respectively, with ancillary nodes (purple and orange) with value clamped to one [Torlai and Melko, 2016]. . . . .	19
2.16	KL divergence as a function of training step (a) and probability distributions (b) for a $d=1$ Ising model with $N=6$ spins. We show the comparison between the exact probability distribution (red) and the approximate distribution produced by the Boltzmann machine after 10 (green) and 500 (blue) training steps for all of the $2^6$ states $\sigma$ [Torlai and Melko, 2016]. . . . .	19
2.17	Scaling of the specific heat $C_V$ with the number of hidden nodes $n_H$ . In (a) we show scaling at different temperatures $T$ , when the system is ordered (blue and cyan), disordered (red and pink), and critical (green). In (b) we show the scaling at criticality for different system sizes $L$ . Dotted lines represent the exact value computed on the spin configurations of the training data set. [Torlai and Melko, 2016]. . . . .	20
3.1	Comparison between CPU and GPU multicore, the figure shows, how a GPU have much more cores than the CPU, thereby it can compute more process at the same time. [Zeno, 2014] . . . . .	22
3.2	Two core GPU architecture diagram, where Host is the CPU. [Zeno, 2014] . . . . .	23
3.3	Class diagram. . . . .	24
3.4	Methodology flowchart. . . . .	26
3.5	Geographic location of the monitoring stations where the data was obtained ([SEDEMA, 2018]).	27
3.6	Graphical User Interface. . . . .	28
3.7	Graphical User Interface showing the results of predicting 120 hours of $PM_{10}\mu/m^3$ particle pollution concentration in the air for the site "MGH" after March 12th, 2016.	29
4.1	LSTM prediction of 12 hours of $PM_{10}$ for AJM site. . . . .	32
4.2	LSTM prediction of 12 hours of $PM_{10}$ for XAL site. . . . .	32
4.3	LSTM prediction of 24 hours of $PM_{10}$ for BJU site. . . . .	32
4.4	LSTM prediction of 24 hours of $PM_{10}$ for XAL site. . . . .	33
4.5	LSTM prediction of 48 hours of $PM_{10}$ for AJM site. . . . .	33
4.6	LSTM prediction of 48 hours of $PM_{10}$ for MGH site. . . . .	33
4.7	LSTM prediction of 120 hours of $PM_{10}$ for AJM site. . . . .	34
4.8	LSTM prediction of 120 hours of $PM_{10}$ for BJU site. . . . .	34
4.9	GRU prediction of 12 hours of $PM_{10}$ for AJM site. . . . .	35
4.10	GRU prediction of 12 hours of $PM_{10}$ for MGH site. . . . .	36
4.11	GRU prediction of 24 hours of $PM_{10}$ for BJU site. . . . .	36
4.12	GRU prediction of 24 hours of $PM_{10}$ for MGH site . . . . .	36
4.13	GRU prediction of 48 hours of $PM_{10}$ for XAL site. . . . .	37
4.14	GRU prediction of 48 hours of $PM_{10}$ for AJM site. . . . .	37
4.15	GRU prediction of 120 hours of $PM_{10}$ for AJM site. . . . .	37
4.16	GRU prediction of 120 hours of $PM_{10}$ for XAL site. . . . .	38
4.17	Boxplot of RMSE for LSTM model predictions by site. . . . .	39
4.18	Boxplot of GRU for LSTM model predictions by site. . . . .	39
4.19	Boxplot of RMSE for LSTM predictions by prediction target length. . . . .	40

4.20	Boxplot of RMSE for GRU predictions by prediction target length. . . . .	40
4.21	Boxplot of RMSE for LSTM predictions by month. . . . .	41
4.22	Boxplot of GRU for LSTM predictions by month. . . . .	41

Dirección General de Bibliotecas UAQ

Dirección General de Bibliotecas UAQ



# List of Tables

3.1	GPU Specifications . . . . .	23
3.2	CPU Specifications . . . . .	23
3.3	Batches used to train the model. . . . .	25
3.4	Raw data statistics used to train the model. . . . .	27
4.1	LSTM Root Mean Square Error in $PM_{10} \mu/m^3$ . . . . .	38
4.2	GRU Root Mean Square Error in $PM_{10} \mu/m^3$ . . . . .	38

Dirección General de Bibliotecas UAQ

Dirección General de Bibliotecas UAQ

---

# Introduction

## 1.1 Motivation

During the physics engineering bachelor, several physical and mathematical models and theories proposed through the history of science were studied. Those models that have transcended through the time are robust and reliable. Using them allows people to predict and understand behaviors of physical phenomena.

Now we have powerful computers that allow us to implement more complicated prediction algorithms and apply them to solve physics, mathematics, economical and environmental problems.

Airborne pollution is an environmental problem that is increasing due to the transport vehicles, industry among others. PM10 particles are one of the biggest issues that need to be studied, they aggravate asthma and respiratory and cardiovascular diseases. Chronic exposure to high concentrations may cause an increase in the risk of morbidity and mortality [SEDEMA, 2018].

It is known that Artificial Intelligence (AI) and Machine Learning has been implemented for many applications, such as image analysis and voice recognition [Långkvist et al., 2014] but it has not been fully investigated for environmental applications.

The motivations are to innovate with this model capable to forecast time series of particle pollution, therefore make an environmental impact with this research; and the challenge that entails to work with deep learning algorithms.

## 1.2 Problem Formulation

Forecasting of airborne pollution is one topic of research due to the negative effects in the respiratory system caused by the PM10 particles. The effects of total suspended particles (TSP) on human health are not limited to the respiratory system, but also can lead to cardiovascular diseases. The effects can be immediate or felt after several days of exposure to these pollutants [Aceves-Fernandez et al., 2015]. By creating a robust model that is able to forecast air pollution in an urban area, it would be easier to decrease the concentration of certain particles after analyze their behavior.

## 1.3 Objectives

The general objective of this project is to generate a model to forecast air pollution of PM10 particles by using machine learning techniques.

### 1.3.1 Specific Objectives

- Acquire data to generate the model.
- Pre-process the raw data with techniques such as filtering, dealing with missing values or outliers, among others.
- Generate a robust, reliable and readjustable model to forecast PM10 particles using deep learning.
- Compare different optimizers for the neural network and use the one that fits better the cost function.
- Train the model with different length of batches and compare which of them is more accurate when the prediction is made.
- Study the results of the model, and compare them with the real data.
- Determine the accuracy of the model using the proposed methodology.

## 1.4 Hypothesis

It is feasible to accurately model and forecast PM10 airborne particles using artificial neural networks. To achieve this, several statistic and deep learning techniques must be applied to deal with the non-linearity behaviour of the system. The model shall output predictions according to the provided training data, the most training it receives, the best performance it will show, due to the learning properties of the recurrent neural networks and their variations, decreasing the gradient vanishing point.

## 1.5 Thesis Structure

The thesis is organized as follows:

- Chapter 1 is about the aim of this project introducing the general overview, the objectives, hypothesis and motivation.
- Chapter 2 is about artificial intelligence topics, neural networks background, some of the deep learning architectures and their applications are explain there and an overview of the airborne pollution emphasizing on the PM10 particles, machine learning applied to physics problems.
- Chapter 3 is about the methodology that was followed within the project, the hardware and software used and the design of the the neural network model step by step.

- Chapter 4 shows the results obtained by using the mentioned methodology, significance and future work.
- Chapter 5 is a summary of work done and what was achieved, issues detected within the model and possible reasons.

Dirección General de Bibliotecas UAQ

Dirección General de Bibliotecas UAQ

---

# Literature Survey

## 2.1 Airborne Pollution

$PM_{10}$  particle concentration sequence modeling has shown good results using statistic and machine learning algorithms, see [Skrzypski et al., 2009, Kukkonen et al., 2003]. Forecasting of airborne pollution is one topic of research due to the negative effects in the respiratory system caused by the  $PM_{10}$  particles. The effects of total suspended particles (TSP) on human health are not limited to the respiratory system, but also can lead to cardiovascular diseases. The effects can be immediate or felt after several days of exposure to these pollutants.[Aceves-Fernandez et al., 2015]

The daily activity of a city generates a huge amount of substances that modify the natural composition of the air. Burning fossil fuels to generate energy and transportation produces tons of pollutants which are emitted into the atmosphere, as shown on figure 2.1.

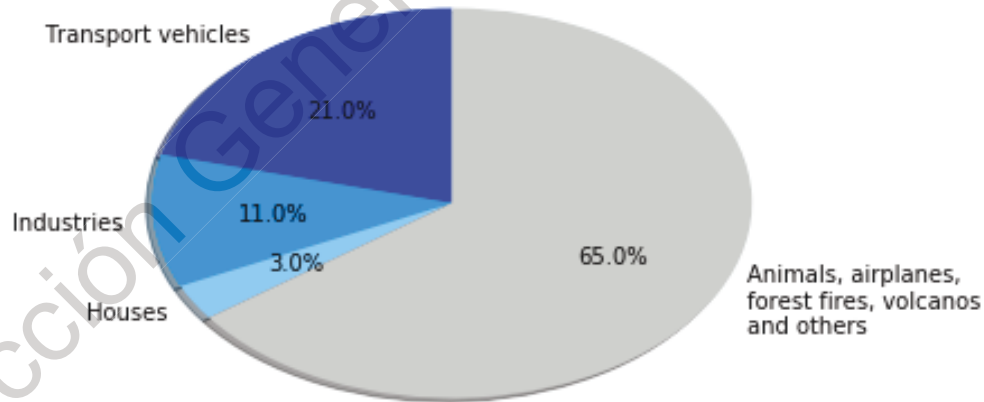


Figure 2.1: Contribution of  $PM_{10}$  pollution emitters in Mexico City (data obtained from *Red Automática de Monitoreo Ambiental* [SEDEMA, 2018])

Airborne pollution is doubtless a topic of concern worldwide. It has been widely documented the



risk in morbidity and mortality due to high concentration of these pollutants [Aceves-Fernandez et al., 2015].

The toxicity of the particles is determined by their physical and chemical characteristics. For instance, size, which is measured in aerodynamic diameter. This is an important parameter to characterize its behavior, among others. This parameter is of paramount importance due to its capacity of penetration and retention in the human respiratory system; also, size may determine its residence time in the atmosphere, hence, the concentration to which population may be at risk [NOM, 2005]. Because of their small size, particles on the order of 10 micrometers or less (PM10) can penetrate into the lungs. The task of predicting contaminants is not an easy task as does not present a linear behavior [Martinez et al., 2014]. Provide various analysis and research made in the field of your proposal and the results already published or existing products.

## 2.2 Air Quality and Weather Monitoring

A way to protect the health of the population is through the monitoring, prediction and diffusion of the air quality in the cities. The Atmospheric Monitoring System (SIMAT for its acronym in Spanish) is conformed by four subsystems: RAMA, REDMA, REDMET y REDDA.

The Automatic Environmental Monitoring Network (RAMA for its acronym in Spanish) use equipment to measure sulfur dioxide, carbon monoxide, nitrogen dioxide, ozone, PM10 and PM2.5. It has 29 monitoring stations in Mexico City (figure 2.2) and has a laboratory for maintenance and calibration of monitoring equipment [SEDEMA, 2018].

The Meteorology and Solar Radiation Network (REDMET for its acronym in Spanish) is composed of 19 sites in Mexico Cites with continuous equipment for the measurement of the main surface meteorological variables: temperature, relative humidity, wind direction and speed, solar radiation and barometric pressure [SEDEMA, 2018].

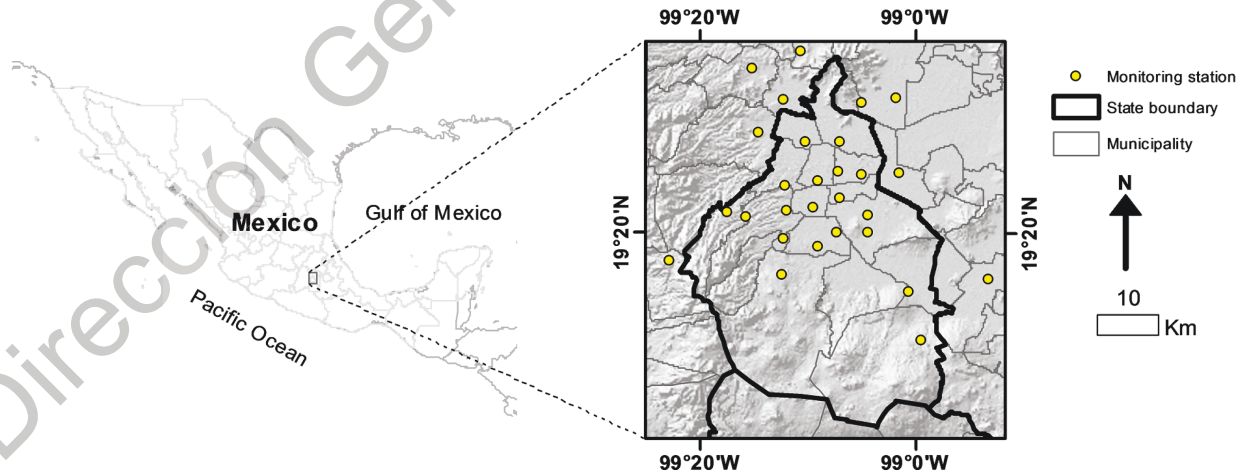


Figure 2.2: RAMA monitoring sites ([Aguirre-Salado et al., 2017])

## 2.3 Artificial Neural Networks

An artificial neuron (AN), implements a nonlinear mapping:

$$f_{AN} : \mathbb{R}^I \rightarrow [0, 1] \quad (2.1)$$

or

$$f_{AN} : \mathbb{R}^I \rightarrow [-1, 1] \quad (2.2)$$

where  $I$  is the number of input signals to the AN. Figure 2.3 presents an illustration of an AN with notational conventions that will be used throughout this text. An AN receives a **vector** of  $I$  input signals, as shown on equation 2.3

$$z = (z_1, z_2, \dots, z_I) \quad (2.3)$$

either from the environment or from other ANs. To each input signal,  $z_i$ , is associated a weight,  $v_i$ , to strengthen or weaken the input signal. The AN computes the net input signal, and uses an activation function  $f_{AN}$  to compute the output signal,  $o$ , given the net input. The strength of the output signal is further influenced by a threshold value,  $\theta$ , also referred to as the *bias*. The net input signal to an AN is usually computed as the weighted sum of all input signals referred as *summation units* (SU),

$$net = \sum_{i=1}^I z_i v_i \quad (2.4)$$

But it can also use *product units* (PU) to perform a higher order combinations of inputs

$$net = \prod_{i=1}^I z_i^{v_i} \quad (2.5)$$

having the advantage of increased information capacity [Engelbrecht, 2007].

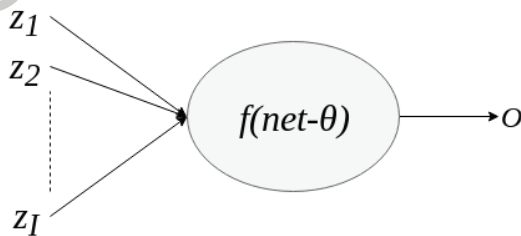


Figure 2.3: Artificial Neuron [Engelbrecht, 2007]

Having several artificial neurons such as the one in Figure 2.3, The outputs  $o$  of some of them to be the input signals  $z_I$  of new artificial neurons to create an *Artificial Neural Network* (ANN) may be defined.

## 2.3.1 Artificial Intelligence, Machine Learning and Deep Learning

### Artificial Intelligence

Artificial Intelligence (AI) is a general field that encompasses machine learning (ML) and deep learning (DL), but that also includes many more approaches that don't involve any learning, such as evolutionary computation (EC), swarm intelligence (SI), artificial immune systems (AIS), and fuzzy systems (FS) [Engelbrecht, 2007].

### Machine Learning

With ML, humans input data as well as the answers expected from the data, and set the rules. These rules can then be applied to new data to produce original answers. A ML system is trained rather than explicitly programmed. It is presented with many examples relevant to a task, and it finds statistical structure in these examples that eventually allows the system to come up with rules for automating the task [Chollet, 2017].

### Deep Learning

A 2-layer neural networks can approximate any continuous function to any degree. Adding layers adds levels of complexity that may be much harder and may require many more neurons to simulate with shallow networks. DL not only refers to the depth of the network, or how many layers the neural net has, but to the level of "learning". In DL, the network does not simply learn to predict an output Y given an input X, but it also understands basic features of the input. In DL, the neural network is able to make abstractions of the features that comprise the input examples, to understand the basic characteristics of the examples, and to make predictions based on those characteristics [Zocca et al., 2017].

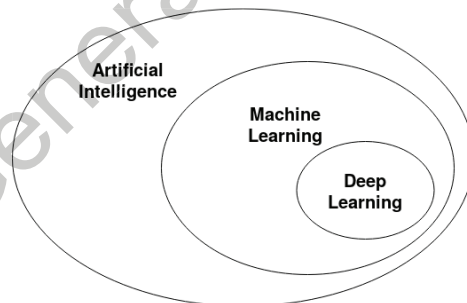


Figure 2.4: Artificial Intelligence, Machine Learning and Deep Learning [Engelbrecht, 2007]

## 2.4 Types of Machine Learning

### 2.4.1 Supervised Learning

The neuron is provided with a data set consisting of input vectors and a target associated with each input vector that is set by the user. This data set is referred to as the training set. The aim of supervised training is then to adjust the weight values such that the error between the real output,

$o = f(\text{net} - \theta)$ , of the neuron and the target output,  $t$ , is minimized [Engelbrecht, 2007].

Supervised Learning has been implemented in several algorithms such as k-nearest neighbors (kNNs) [Zhang, 2016], k-means [Hartigan and Wong, 1979], linear support vector machines (SVMs) [Smola and Schölkopf, 2003], logistic regression [Hosmer and Lemeshow, 2000] and random forests [Breiman, 2001]. They have been used for several applications such as the investigation of Bzdok Et.Al.[Bzdok et al., 2018] where they applied SVMs and kNNs to challenging pattern-recognition problems in biology and medicine.

### 2.4.2 Unsupervised Learning

An important feature of NNs is their ability to learn from their environment. The supervisor presents the NN learner with an input pattern and a desired response. Supervised learning NNs then try to learn the functional mapping between the input and desired response vectors. In contrast to supervised learning, the objective of unsupervised learning is to discover patterns or features in the input data with no help from a learner [Engelbrecht, 2007].

Unsupervised learning has been applied into a complex range of algorithms, such as natural language processing [Berger et al., 1996], where interactions between computers and human (natural) language is studied and analyzed.

### 2.4.3 Self-supervised Learning

Self-supervised learning is supervised learning without human-annotated labels. There are still labels involved (because the learning has to be supervised by something), but they're generated from the input data, typically using an algorithm [Chollet, 2017].

Supervised learning targets the category of methods that generate an intrinsic reward signal based on how hard it is for the agent to predict the consequences of its own actions, i.e. predict the next state given the current state and the executed action. Deepak Pathak et. Al. [Pathak et al., 2017] apply it to discover how to play *Super Mario Bros* without rewards using Curiosity-driven Exploration, shown in figure2.5.

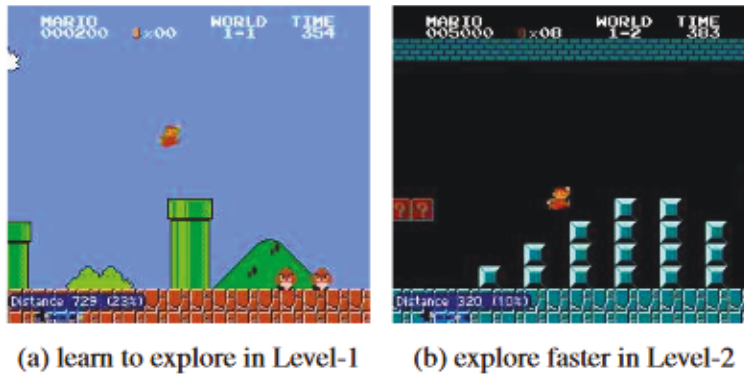


Figure 2.5: (a) Using only curiosity-driven exploration, the agent makes significant progress in Level-1. (b) The gained knowledge helps the agent explore subsequent levels much faster than when starting from scratch. [Pathak et al., 2017]

#### 2.4.4 Reinforcement Learning

In reinforcement learning, the aim is to reward the neuron (or parts of a NN) for good performance, and to penalize the neuron for bad performance.

The agent decides on actions based on the current environmental state, and through feedback in terms of the desirability of the action, learns which action is best associated with which state. The agent learns from interaction with the environment [Engelbrecht, 2007].

Due to the mechanism of reward and penalize, reinforcement learning is applied in control and automated tasks. Lillicrap et. al. [Lillicrap et al., 2016] presented an algorithm capable of solve more than 20 simulated physics tasks, including classic problems such as cartpole swing-up, dexterous manipulation, legged locomotion and car driving.

## 2.5 Neural Network Architectures

### 2.5.1 Recurrent Neural Networks

Several multi-layer neural networks have been developed, one of them are the Recurrent Neural Networks (RNN). Those have feedback connections that allows them to model time series problems.

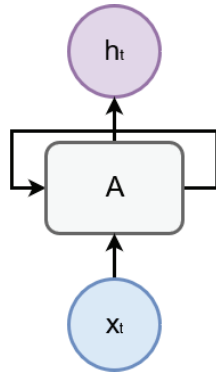


Figure 2.6: Recurrent Neural Network model [Hochreiter and Schmidhuber, 1997].

It can be observed the flowchart of a RNN  $A$  (Figure 2.6), where  $x_t$  are the inputs and  $h_t$  the output signals. This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists, in the past years they have been applied to voice recognition problems, language modeling, and time-series [Långkvist et al., 2014].

We can think of simple RNN as a NN with single layer that is repeated and pass the information to the next NN (Figure 2.7 ).

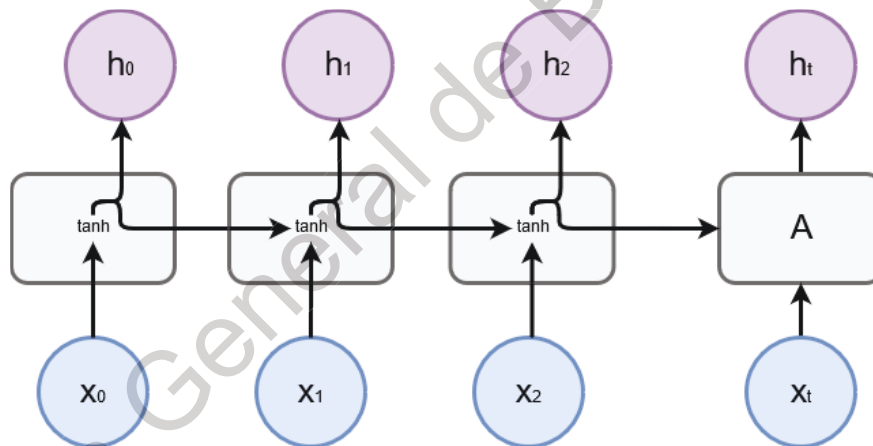


Figure 2.7: interpretation of a recurrent neural as a conventional neural network [Hochreiter and Schmidhuber, 1997]

The RNN uses the hyperbolic tangent activation function (equation 6).

### 2.5.2 Long Short Term Memory

The Long Short Term Memory (LSTM) is a type of RNN. They were introduced in 1997 by Hochreiter and Schmidhuber [Hochreiter and Schmidhuber, 1997] designed to avoid the long term dependency problems. They can remember the learned information for long periods of time without any learning problem.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer, LSTM networks also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way (see figure 2.8)[Hochreiter and Schmidhuber, 1997].

The cell state  $C_{t-1} \rightarrow C_t$  is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. This way is convenient for information to just flow along the chain unchanged. The LSTM networks have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer (5) denoted by  $\sigma$  and a pointwise multiplication operation  $\times$ .

The sigmoid function output 1 represents a complete removal of the data while 0 represents a complete storaction of the data [Hochreiter and Schmidhuber, 1997].

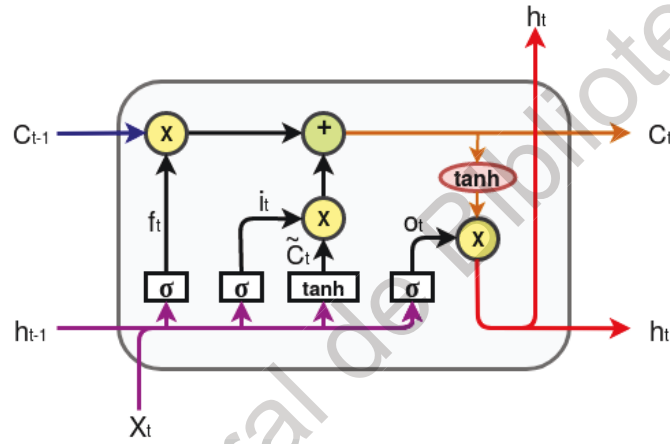


Figure 2.8: The repeating module in an LSTM contains four interacting layers [Zhou et al., 2016].

1. The *forget gate layer* takes the input values  $x_t$  and  $h_{t-1}$ , applies the sigmoid function to determine what to preserve and discard, then is multiplied by  $C_{t-1}$ .

$$f_t = \sigma(W \cdot [h_{t-1}, x_t] + b_f) \quad (2.6)$$

Where  $W$  is a weight and  $b_f$  the bias.

2. The next step is to decide what information is going to store in the cell state. First, a sigmoid layer called the *input gate layer* decides which values are going to be updated. Next, a tanh layer creates a vector of new candidate values,  $\tilde{C}_t$ , that could be added to the state. In the next step, these two steps are combined to create an update to the state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.7)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (2.8)$$



3. The old cell state  $C_{t-1}$  unto the new  $C_t$  is updated

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t \quad (2.9)$$

4. Finally an output is chosen:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.10)$$

$$h_t = o_t * \tanh(C_t) \quad (2.11)$$

### 2.5.3 Gated Recurrent Unit

The Gated Recurrent Unit (GRU) architecture was introduced in 2014 by [Cho et al., 2014]. It is considered a variation of the LSTM, it uses the same functions but they are organized in a different way. Recent studies trend towards simpler recurrent units suggests a need for RNNs with smaller memory footprints and lower training computational load [Heck and Salem, 2017].

To solve the vanishing gradient problem of a standard RNN, GRU uses, so called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. One of the main features of this architecture is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction [Chung et al., 2014].

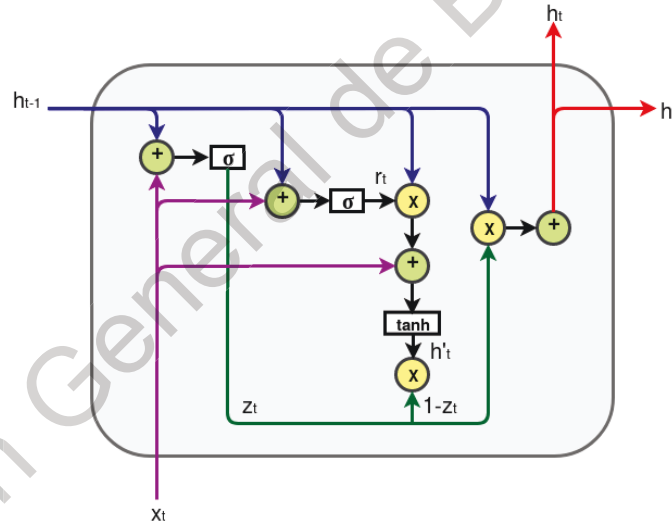


Figure 2.9: Gated Recurrent Unit Model [Zhou et al., 2016].

1. The algorithm starts by calculating the *update gate*  $z_t$  for the time step  $t$ :

$$z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{t-1}) \quad (2.12)$$

When  $x_t$  is plugged into the network unit, it is multiplied by its own weight  $W^{(z)}$ . The same applies for  $h_{t-1}$  which holds the information for the previous  $t - 1$  units and is multiplied by its own weight  $U^{(z)}$ . Both results are added together and a sigmoid activation function is applied to scale the result between 0 and 1.

2. The *reset gate* uses the model to decide how much of the past information to forget, to calculate it, it uses:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \quad (2.13)$$

3. Here a new memory content is introduced, which will use the *reset gate* to store the most important information from the past.

$$h'_t = \tanh(Wx_t + [r_t * Uh_{t-1}]) \quad (2.14)$$

- a) Multiplies the input  $x_t$  by a weight  $W$  and  $h_{t-1}$  by a weight  $U$ .
  - b) Calculate the Hadamard (element-wise) product 1 between the reset gate  $r_t$  and  $Uh_{t-1}$ . That will determine what to remove from the previous time steps.
  - c) Sums the results of the steps a) and b)
  - d) Applies the nonlinear activation function  $\tanh$  (6).
4. Finally it is necessary to calculate the vector  $h_t$  that holds the information of the current unit and passes it down to the network. It determines what to collect from the past steps  $h_{t-1}$  using the following equation 2.15:

$$h_t = [z_t * h_{t-1}] + [(1 - z_t) * ht] \quad (2.15)$$

## 2.6 Machine Learning Applications to Physics

### 2.6.1 Schrödinger Equation

Obtaining an accurate ground state wave function is one of the great challenges in the quantum many-body problem, it was originally approached by [Feynman and Cohen, 1956] and [Luo and Clark, 2019] used neural networks to obtain the wave functions.

The figure 2.10 represents the configuration of 4 electrons on 3 sites. Each input represents the state of the spin, mapped as 1 and -1, the layers are connected with arrows representing the mentioned configuration but only a fraction of them are shown for image clarity. Parameters  $b_j$  and  $w_{i,j}$  are bias and weights to be optimized. The output layer is an  $n_{up-electrons} \times n_{sites}$  matrix which will be the backflow transformation added to the single particle orbitals.

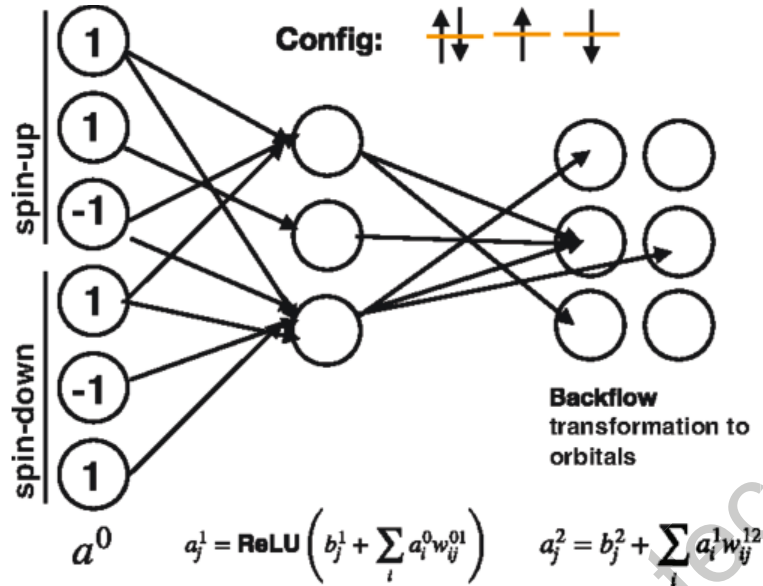


Figure 2.10: Neural network used to find ground state wave functions [Luo and Clark, 2019].

As results Di Luo and Bryan K. Clark obtained the following simulations (figures 2.11 and 2.12) using ReLU units and the architecture specified in figure 2.10.

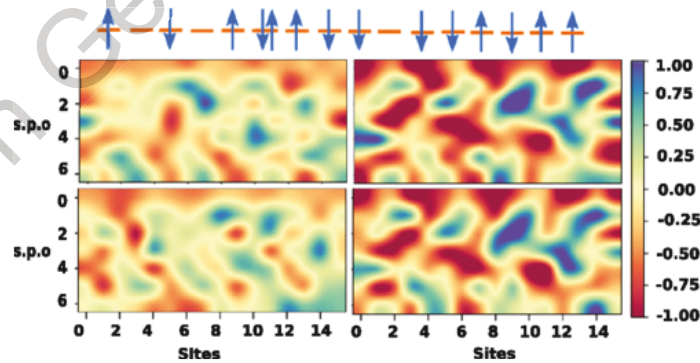


Figure 2.11: Spin-up (top) and spin-down (bottom) single particle orbital(s.p.o) for  $\psi_{S0}$ (left) and  $\psi_{SN}$ (right) with 256 hidden neurons on 4x4 Hubbard model at  $U/t=8$ ,  $n=0.875$ , where the s.p.o are evaluated at the (1d-reshaped) spin-configuration shown. For the s.p.o, row is orbital index and the column is position index [Luo and Clark, 2019].

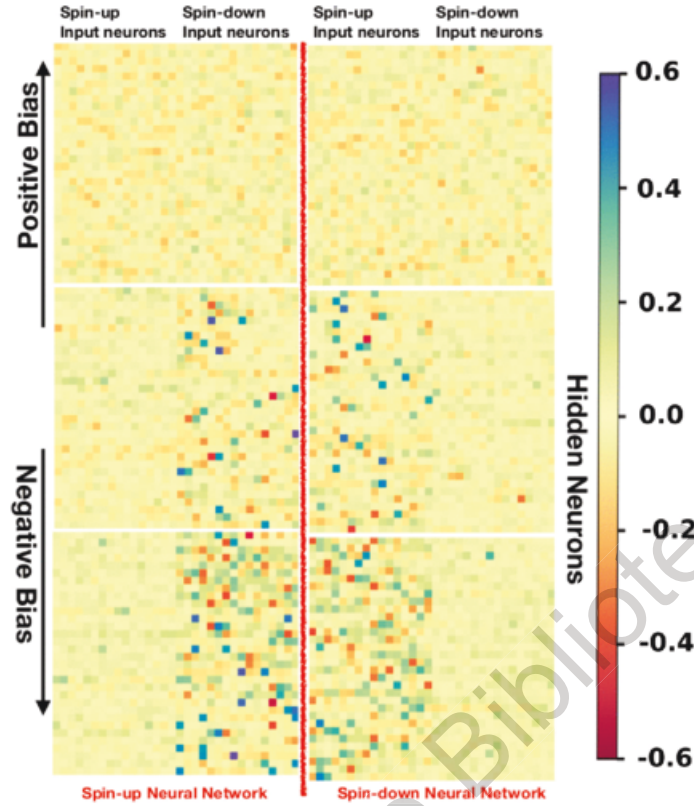


Figure 2.12: Weights between the input layer and hidden layer for  $\psi_{SN}$  with 256 hidden neurons for the spin-up (left) and spin-down (right) neural networks on 4x4 Hubbard model at  $U/t=8$ ,  $n=0.875$ . Hidden neurons are ordered by their bias and shown are neurons 1-32 (top), 96-128 (middle) and 224-256 (bottom). [Luo and Clark, 2019].

## 2.6.2 Image Analysis for Morphological Galaxy Classification

Image analysis and classification are important topics within ML applications, For instance [De La Calleja and Fuenzalida, 2019] present an experimental study of ML and image analysis for performing automated morphological galaxy classification. They used a neural network, and a locally weighted regression method, and implemented homogeneous ensembles of classifiers. The ensemble of neural networks was created using the bagging ensemble method, and manipulation of input features was used to create the ensemble of locally weighed regression.

They developed a method for classifying galaxy images consisting of three parts (see Fig. 1): the Image Analysis Module (IAM), the Data Compression Module (DCM), and the Machine Learning Module (MLM). The method works as follows. First it takes as input the galaxy images, which are then rotated, centred, and cropped in the AIM. Next, in the DCM, principal component analysis is used to reduce the dimensionality of the data and to find a set of features (principal components). The projection of the images onto the principal components gives the input parameters for the MLM. At the end, we will have the classification of the galaxies and the accuracy of each ML algorithm, as well as the accuracy of the ensembles.

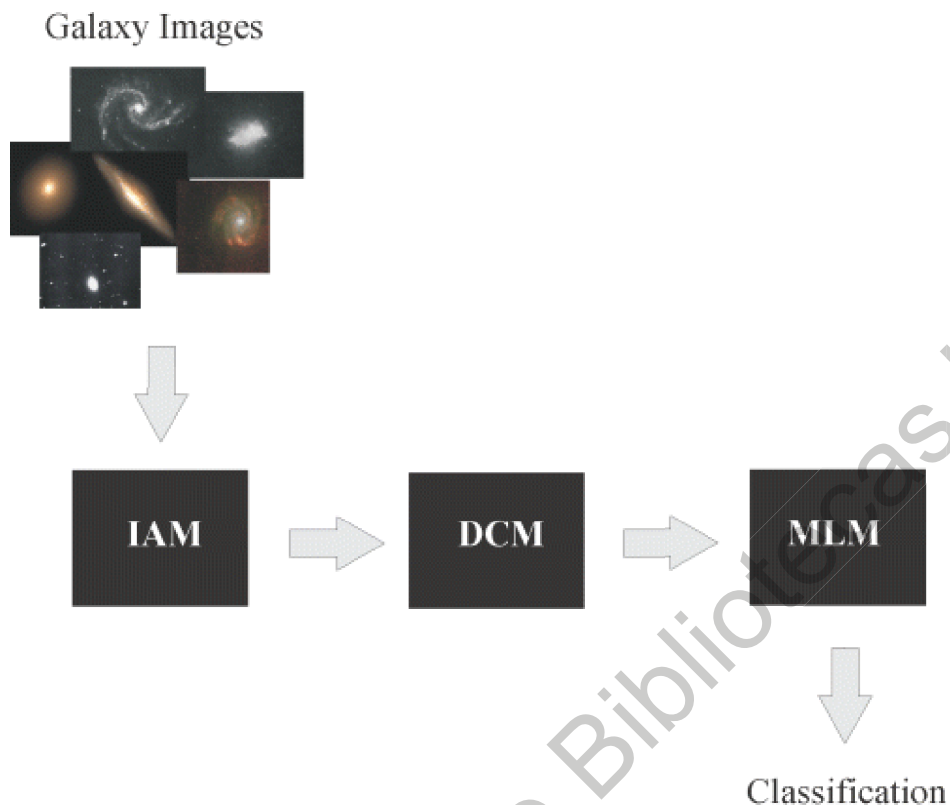


Figure 2.13: Elements of the classification method proposed by [De La Calleja and Fuentes, 2004].

They concluded that locally weighted regression obtains better results than artificial neural networks, and furthermore that it is faster. However, the ensembles of neural networks improve their individual classification more than the ensembles of locally weighted regression. This happens because artificial neural networks is an unstable type of algorithm, and bagging works especially well in this kind of algorithm.

### 2.6.3 Molecular Dynamics

The paper *Molecular Dynamics with On-the-Fly Machine Learning of Quantum-Mechanical Forces* [Li et al., 2015], a molecular dynamics scheme which combines first-principles and ML techniques in a single information-efficient approach, used Density-functional theory (DFT), a computational quantum mechanic method [Hohenberg and Kohn, 1964] as a framework to do this, notably through first-principles molecular dynamics (FPMD) simulations.

They proposed an alternative ML based scheme where a stream of fresh quantum-mechanical (QM) calculations is allowed to augment the ML database during each MD simulation, enabling safe interpolation. The scheme could equally be viewed as an efficient FPMD approach where we seek to compute only the QM information necessary to progress the simulation, while retaining the very broad applicability of FPMD. To minimize the QM workload of the MD simulation, one can start by noticing that ML-predicted atomic forces will suffice as long as the dynamics visits configuration is well represented within the existing database.

Their results are shown in figure 2.14.

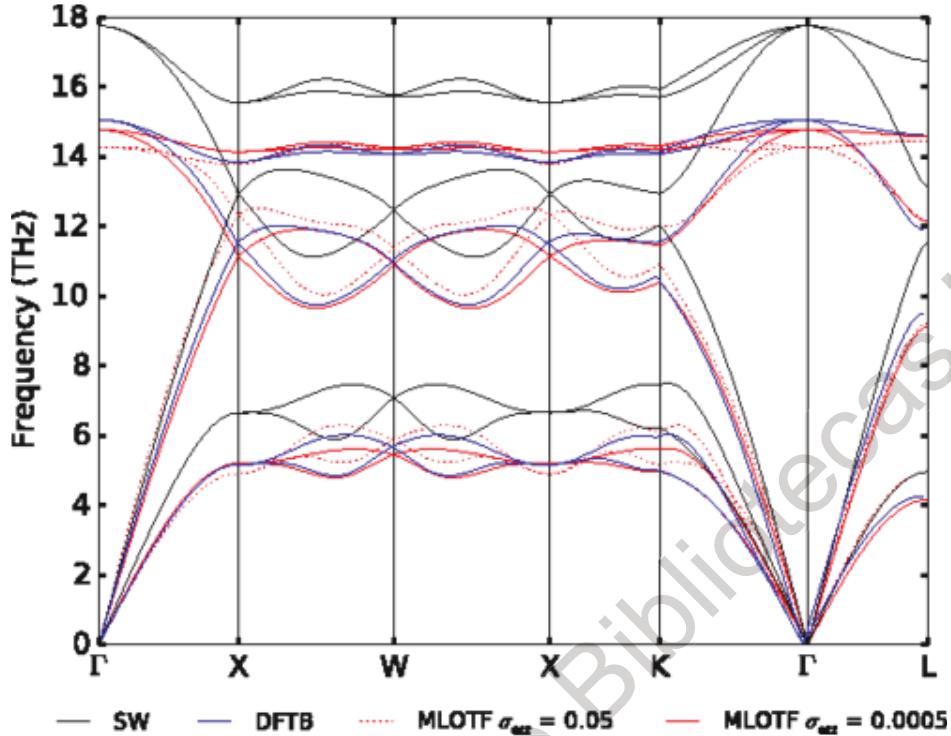


Figure 2.14: Comparison of the bulk Si phonon spectrum calculated with DFTB (blue), and SW (black) and with the ML on-the-fly (MLOTF) approach (red), computed with the finite displacement Parlinski-Li-Kawazoe method using a standard  $\sigma_{err} = 0.055 \times eV/\text{\AA}$  (dotted lines) and a high-accuracy  $\sigma_{err} = 5 \times 10^{-4} eV/\text{\AA}$  value (solid lines) for the ML data noise parameter. The ML database was constructed from a 300 K MD trajectory [Payne et al., 1992].

#### 2.6.4 Boltzmann Machines and Thermodynamics

A Boltzmann machine is a stochastic neural network that has been extensively used in the layers of deep architectures for modern machine learning applications, they are named after the Boltzmann distribution in statistical mechanics, which is used in their sampling function [Ackley et al., 1988]. [Torlai and Melko, 2016] developed a Boltzmann machine that is capable of modeling thermodynamic observables for physical systems in thermal equilibrium. Through unsupervised learning, we train the Boltzmann machine on data sets constructed with spin configurations importance sampled from the partition function of an Ising Hamiltonian at different temperatures using Monte Carlo (MC) methods. The trained Boltzmann machine is then used to generate spin states, for which we compare thermodynamic observables to those computed by direct MC sampling. They demonstrate that the Boltzmann machine can faithfully reproduce the observables of the physical system. Further, we observe that the number of neurons required to obtain accurate results increases as the system is brought close to critically.

The architecture used for the Boltzmann machine is shown in figure 2.15.

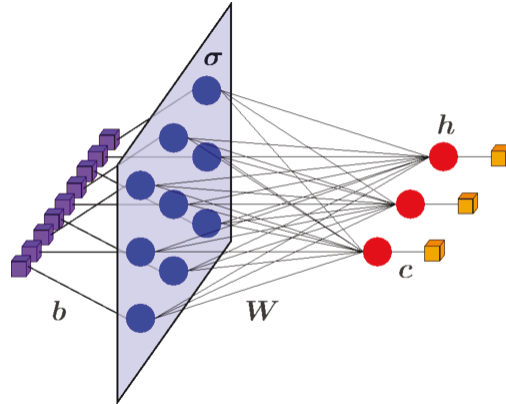


Figure 2.15: Restricted Boltzmann machine. The visible nodes (blue) are connected to the hidden nodes (red) with a symmetric matrix of weight  $W$ . The external fields in the Hamiltonian are represented by new edges with weights  $b$  and  $c$  connecting the visible and hidden nodes, respectively, with ancillary nodes (purple and orange) with value clamped to one [Torlai and Melko, 2016].

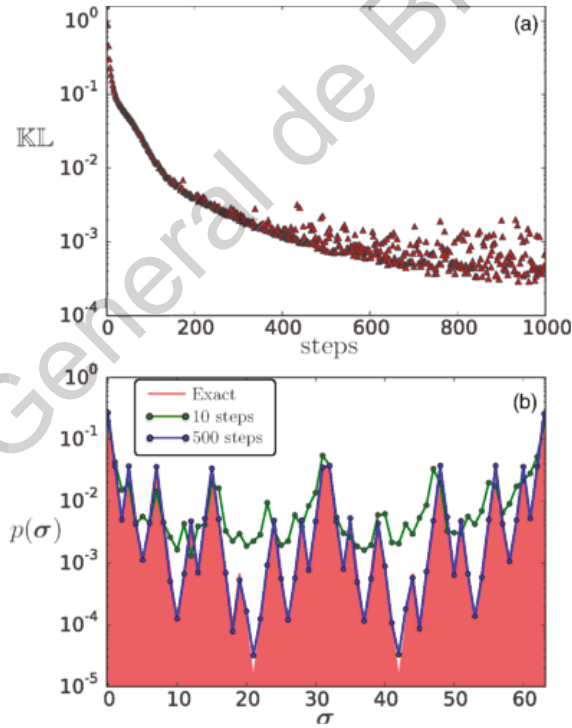


Figure 2.16: KL divergence as a function of training step (a) and probability distributions (b) for a  $d=1$  Ising model with  $N=6$  spins. We show the comparison between the exact probability distribution (red) and the approximate distribution produced by the Boltzmann machine after 10 (green) and 500 (blue) training steps for all of the  $2^6$  states  $\sigma$  [Torlai and Melko, 2016].



The results of the calculated specific heats are shown in figure 2.17.

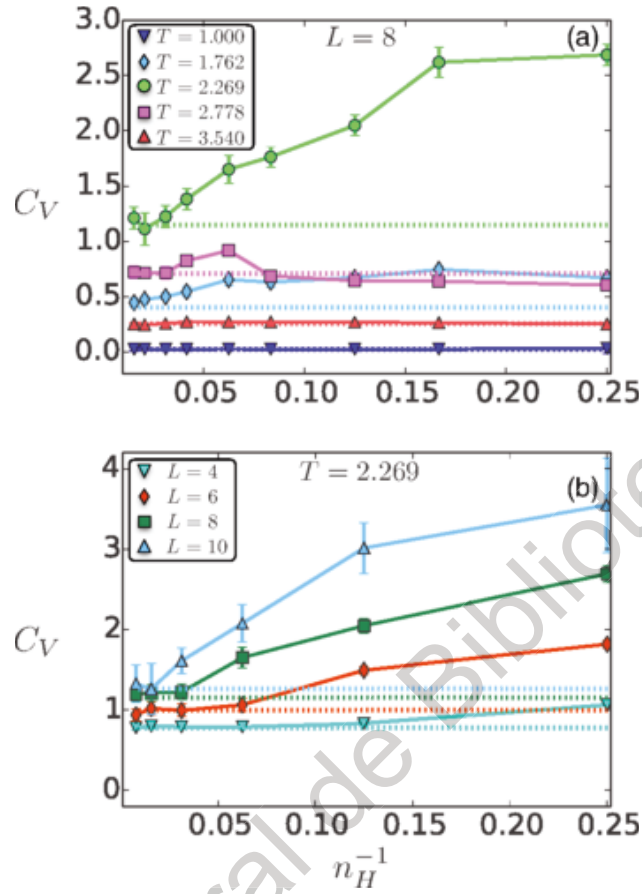


Figure 2.17: Scaling of the specific heat  $C_V$  with the number of hidden nodes  $n_H$ . In (a) we show scaling at different temperatures  $T$ , when the system is ordered (blue and cyan), disordered (red and pink), and critical (green). In (b) we show the scaling at criticality for different system sizes  $L$ . Dotted lines represent the exact value computed on the spin configurations of the training data set. [Torlai and Melko, 2016].



---

# Methodology

## 3.1 Requirements

The model will be trained with previous data of several monitoring stations from RAMA [SEDEMA, 2018] in order to forecast the next steps of data,

### 3.1.1 System Requirements

To generate the neural network model, it is necessary to satisfy the following requirements in order to create a proper environment:

1. The PC must have NVIDIA® GPU card with CUDA® Compute Capability 3.5 or higher in order to satisfy the Tensorflow software requirements.
2. The PC must have installed whether Ubuntu 16.04 or later, MacOS Mojave or later, or Windows 7 or later as operative System.
3. The PC must have installed Python 3.7 or later.
4. The PC must have installed Tensorflow-GPU 1.9 or later.
5. The PC must have at least 8GB of RAM Memory.

### 3.1.2 Python Libraries

The software used to compute this project was Python 3.7.1, the following libraries were used:

1. Tensorflow-gpu
2. Sklearn.
3. Numpy.
4. Pandas.
5. Matplotlib.
6. Seaborn.

## 3.2 Design

### 3.2.1 Hardware System

Machine learning algorithms used to be run within the CPU (Central Processing Unit) in a sequential way, due to the capabilities of the CPU. Then multi-core parallel computing was implemented, in order to run several process at the same time, taking advantage of the architecture of the CPUs.

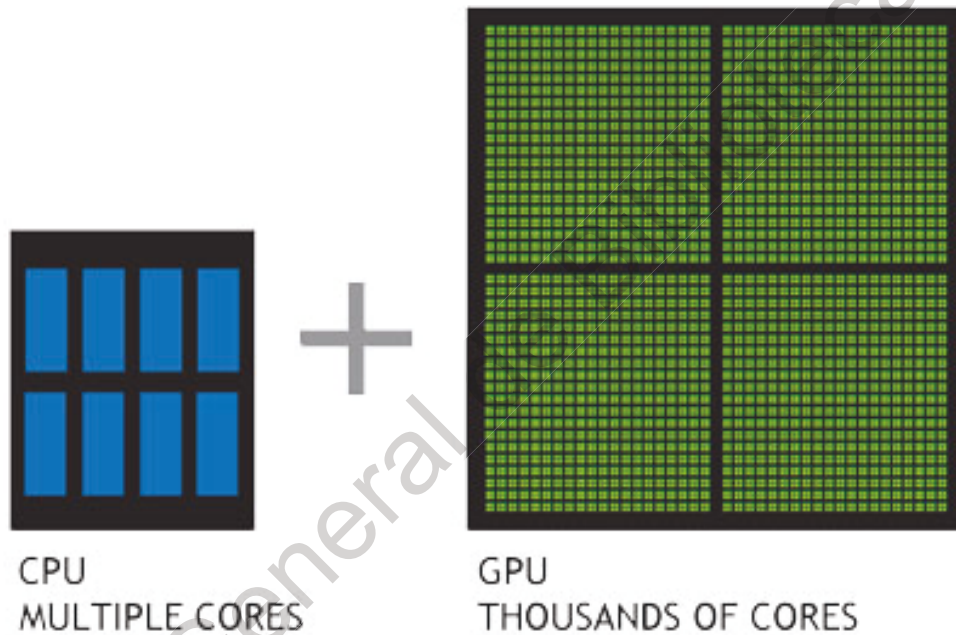


Figure 3.1: Comparison between CPU and GPU multicore, the figure shows, how a GPU have much more cores than the CPU, thereby it can compute more process at the same time. [Zeno, 2014]

Nowdays, machine learning algorithms can be computed using CPU + GPU (Graphical Processing Unit) which makes it much more faster since, GPUs can compute more than a thousand simultaneous process, as shown on figure 3.1. The CPU computes the main process while the process that can be done simultaneously are run within the GPU. Figure 3.2 shows an example of a gpu with 2 cores, whic each of them computes independent process and communicates with the host through the threads. In this project, each GPU core is used as a neuron, reducing the training time by paralelizing the process with the use of the GPU.

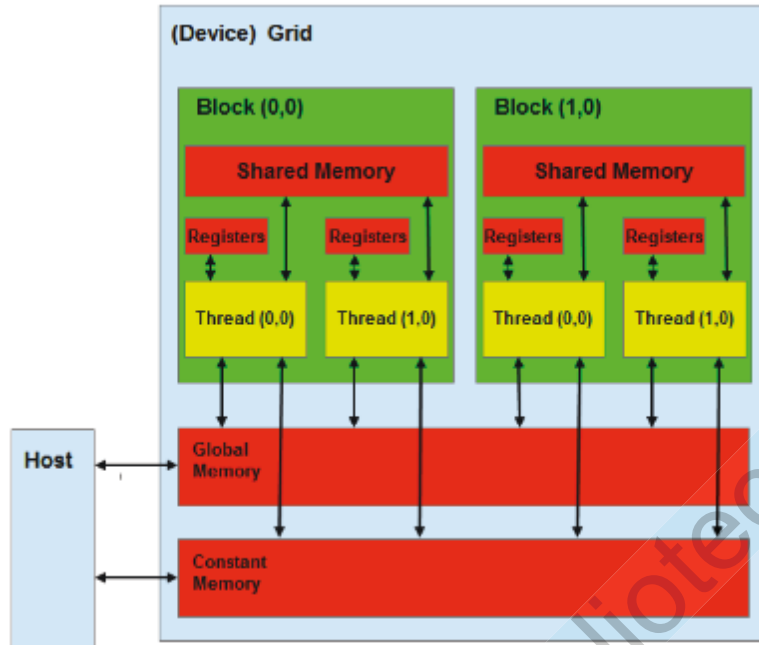


Figure 3.2: Two core GPU architecture diagram, where Host is the CPU. [Zeno, 2014]

For this model the hardware used was a i7-7700HQ processor, a Nvidia GTX-1060 GPU and 16 GB of ram. Full specification are shown in table 3.1 and table 3.2.

Table 3.1: GPU Specifications

Architecture	Pascal
Cores	1280
Buffer	6 GB
Memory Speed	8GB/s
Clock	1708MHz

Table 3.2: CPU Specifications

Cores	4
Threads	8
Clock	3.7 GHz
Cache Level 1	256 Kb
Cache Level 2	1 Mb
Cache Level 3	6 Mb
Chip	14 nm
Max. Tamp.	100 °C
Architecture	x64

### 3.2.2 Software Development Design

#### Class Diagram

The class diagram is represented in figure 3.3.

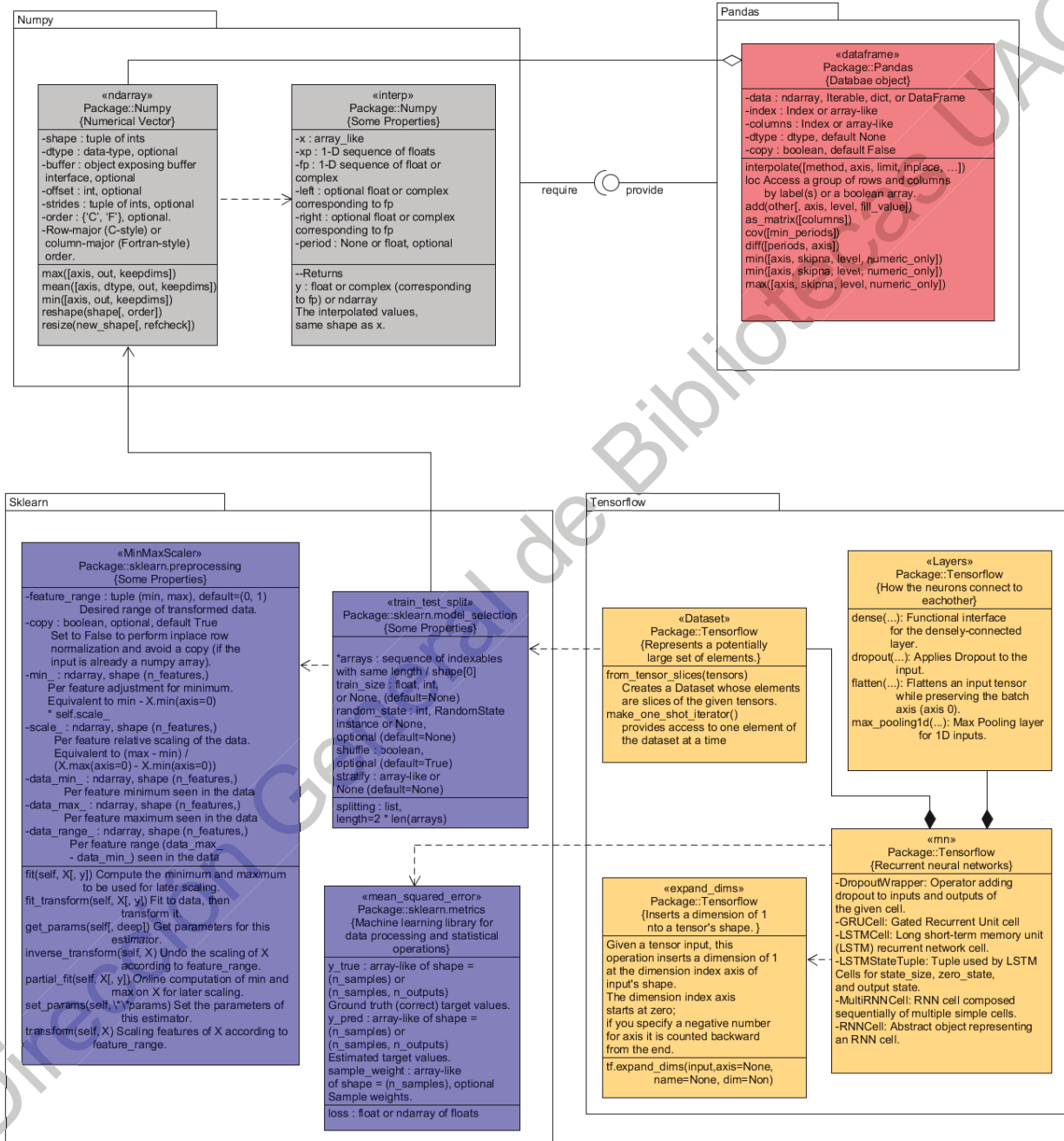


Figure 3.3: Class diagram.

## Model Structure

Below are represented the steps of the used methodology in order to forecast  $PM_{10}$  particle pollution in the target sites (figure 3.4):

1. **Data acquisition:** The first step is to get the required raw data from RAMA [SEDEMA, 2018].
2. **Data Selection:** This part consist in select the data form the target sites and replace the outliers, for this model, the valid range will be from  $0 \mu g/m^3$  to  $150 \mu g/m^3$ .
3. **Fill missing values:** Then, there will be the previous missing values plus the ones that were removed the last step, so a linear regression is used to fill those missing values.
4. **Training and Validation data:** The data sets need to be separated into training and validation sets, since using the same data points for both training and validation may skew the model, giving a false indication of the real accuracy of the model. For this model, the data of a year will be divided into 12 sets (each month) and each of them will be used as a batch of training data.
5. **Format input data:** Before training the model, it is necessary to generate the batches with the proper format that will be used to train this model. The input data must be a vector with dimension of  $1 \times N$  and it must be normalized in a range of  $[0,1]$ . To train the model, four targets will be predicted, training the data for 3000 steps as shown on table 3.3. Where the prediction target is the following data in hours.

Table 3.3: Batches used to train the model.

Prediction Target (hours)	Training Steps
12	3000
24	3000
48	3000
120	3000

6. **Model Construction:** In this section, the model parameters are set. For this work, a GRU and LSTM models were used, the used activation function was a hyperbolic tangent function and a sigmoid function for the gate activation, as show the figures 2.9 and 2.8. The model uses four layers of hidden units with 100 units each. [Zhou et al., 2016] used 100 units for their predictive model, and [Heck and Salem, 2017] decreased the number of units for the shorter sequences to decrease training time using 100 units for the 784-length sequences and 50 units for the 28-length sequences. The output layer will be a fully connected layer of 100 units in all cases, to preserve the input length. The learning rate 0.01, and the batch size 1, due to the configuration of the model, each iteration, a new random batch will be selected, therefore 4000 batches will be used to train the model for each month.
7. **Model Training:** After setting all the parameters, the next step is to train the model.
8. **Model Optimization:** While the model is being trained, an observer that evaluates the loss and adjust the weights of the neurons if needed. For this model, the AdamOptimizer optimizer [Kingma and Ba, 2014] was used.

9. **Value Forecasting:** After the model is trained, a value is forecasted. In this work, 12, 24, 48 and 168 hour forecasting is performed using the methodology described in this section.
10. **Transform output data:** The prediction is made in a  $[0,1]$  scale, so it is necessary to make an inverse transform to the original range  $[0,150]$ , so the prediction is equivalent to the real  $PM_{10}$  particle concentration.
11. **Compare prediction against real data:** The last task to perform is to validate the model, since the further steps of the time series are known, it is possible to compare against the real data and calculate the accuracy of the prediction.

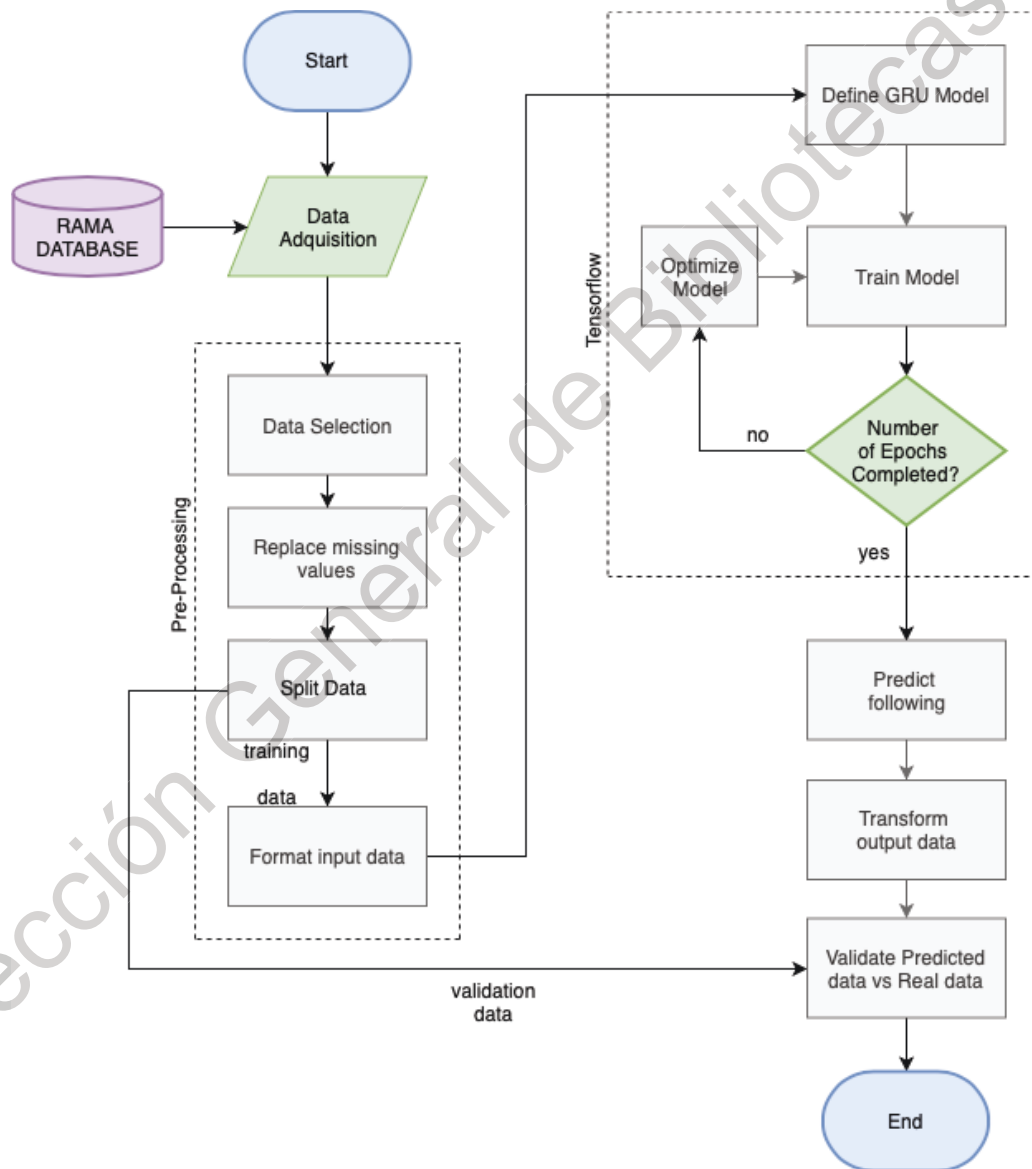


Figure 3.4: Methodology flowchart.

### 3.3 Data

The raw data used for this project, obtained from [SEDEMA, 2018] showed several missing values, so the four sites with less missing values among the twenty nine were selected as training and testing data, using the data obtained in year 2016, one of the sites is located in Mexico State and the rest of them in Mexico City, the location is shown in figure 3.5.

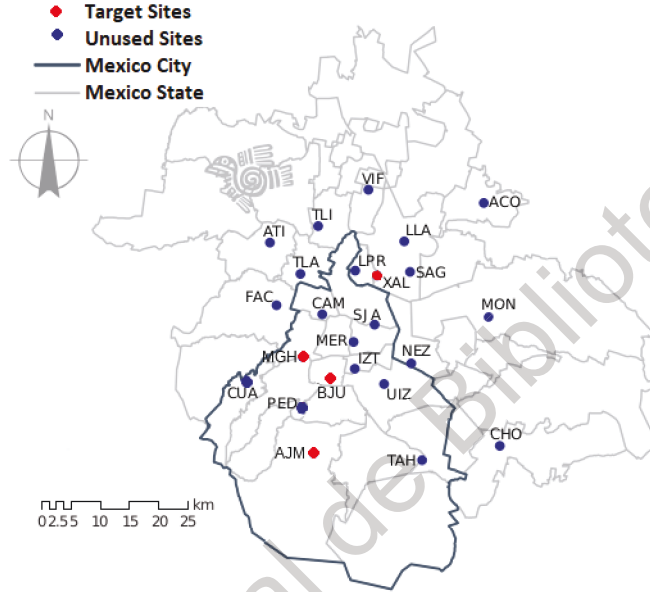


Figure 3.5: Geographic location of the monitoring stations where the data was obtained ([SEDEMA, 2018]).

The minimum values, maximum value, mean value, standard deviation and amount of missing values for each site are shown in table 3.4. The mentioned table shows how the site "XAL" have grater values of concentration of  $PM_{10}$  particles, which implies a grater standard deviation compared with the rest stations.

Table 3.4: Raw data statistics used to train the model.

Site	Min( $\mu/m^3$ )	Max( $\mu/m^3$ )	Mean ( $\mu/m^3$ )	$\sigma(\mu/m^3)$	Missing Values
AJM	1	150	31.48	20.12	472
BJU	1	281	39.44	23.50	187
MGH	2	177	34.13	20.14	124
XAL	2	340	66.38	43.29	213
Total Mean	1.5	237	42.86	26.76	249
Total Absolute	1	340	66.38	43.29	996

### 3.4 Implementation

A graphical user interface (GUI) was implemented to make easier the use of the model (see figure 3.6). The GUI allows the user to select the parameter to predict (for this parameter only  $PM_{10}$  is allowed), then choose between the sites mentioned in table 3.5, select a date, the prediction target of hours after the selected date and finally proceed with the "Predict" button.

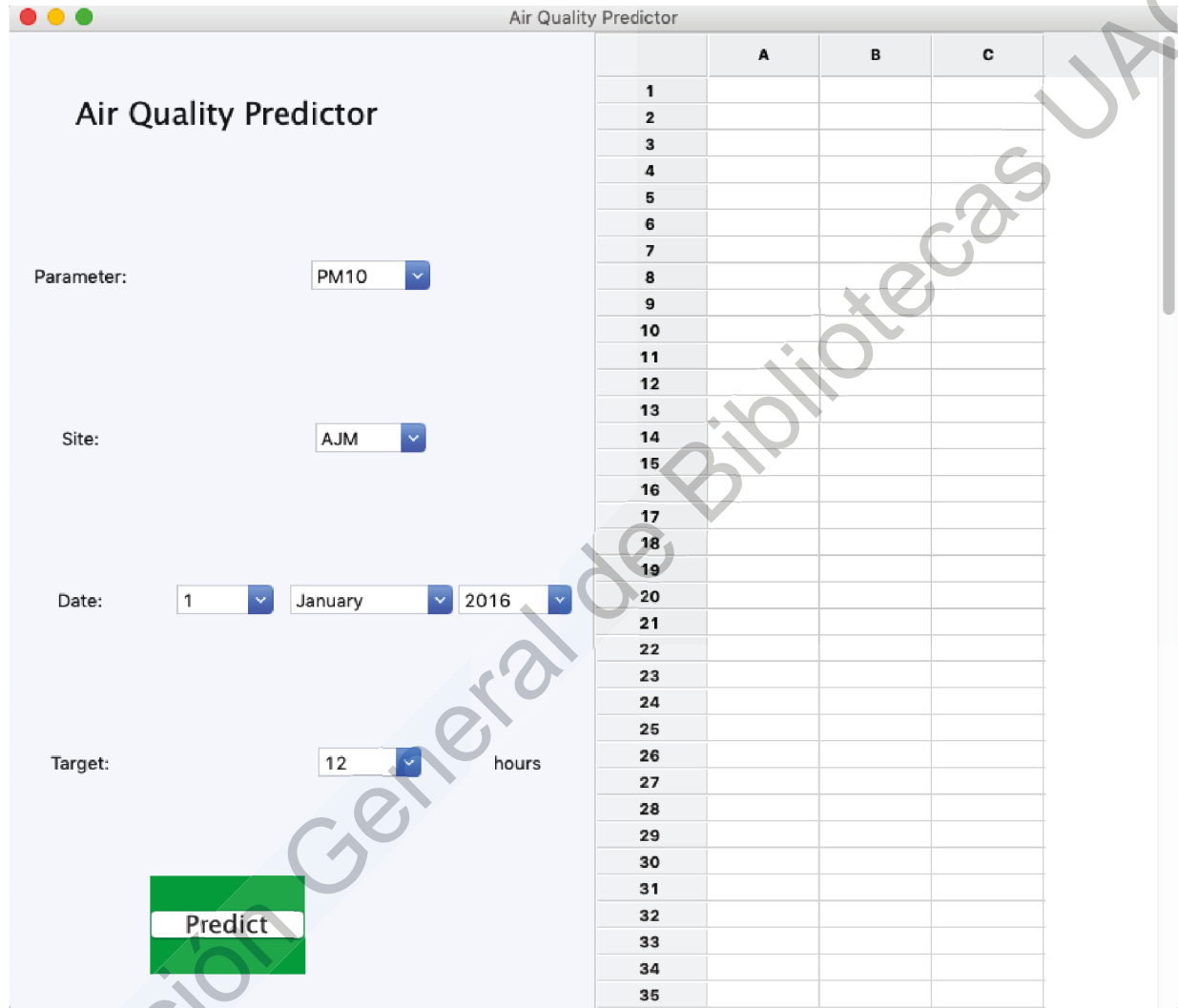


Figure 3.6: Graphical User Interface.



After pressing the button, the GUI takes between 40 and 200 seconds processing the model mentioned in this section and prints the real data, predicted data and the date/time (see figure 3.7).

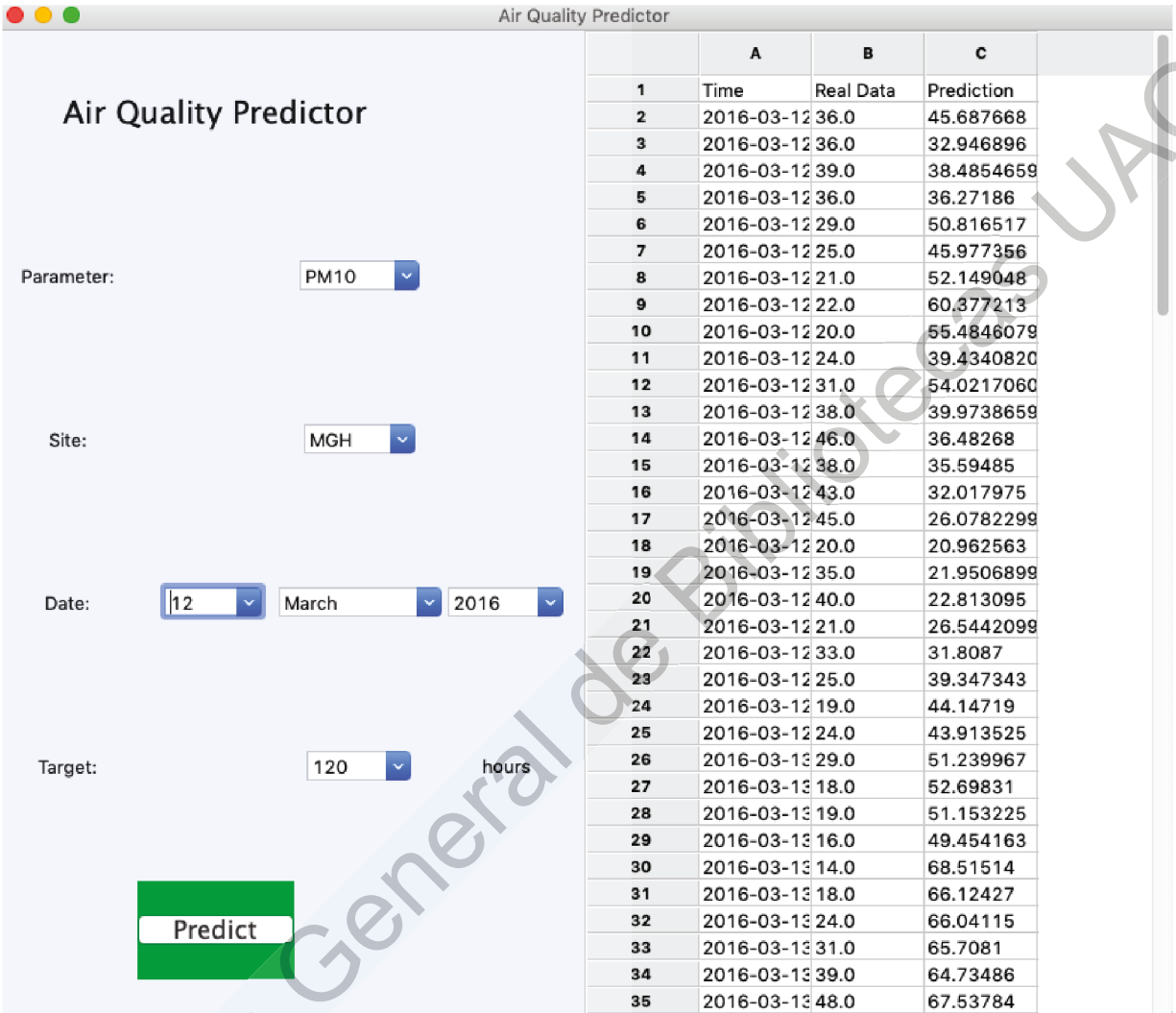


Figure 3.7: Graphical User Interface showing the results of predicting 120 hours of  $PM_{10}\mu/m^3$  particle pollution concentration in the air for the site "MGH" after March 12th, 2016.

Dirección General de Bibliotecas UAQ

---

# Results and Discussion

## 4.1 Results

Two different models were tested and analyzed, LSTM and GRU. Each of the two models were trained for the 12 months of 2016, using batches of 12, 24, 48 and 120 hours for each site mentioned in table 3.4 as stated in the Methodology section.

As result, 384 predictions were completed having a good overall performance.

The results and analysis are presented in the next sub-sections, firstly the LSTM results followed by the GRU results and finally a comparison between both model results.

### 4.1.1 Long Short Term Memory

The model was trained as stated in the methodology section.  $PM_{10}$  particle forecast using LSTM showed good results in most of cases. Figure 4.1 shows a bad performance while figure 4.2 shows better results for a 12h prediction, following the tendency and having a low deviation from the real data.

Predictions for 24h are shown in figures 4.4 and 4.3; for the first figure the prediction failed due to the outliers presented in the real data, the second one shows better results.

For 48h predictions, results are presented in figures 4.5 and 4.6 where the prediction was not as good as in shorter term predictions.

Finally 120h predictions are shown in figures 4.7 and 4.8 where the first of them did not show good results, and the second one had a better accuracy.

The root mean square error (RMSE) calculation for this model is shown in next section in figures 4.19, 4.21 and 4.17.

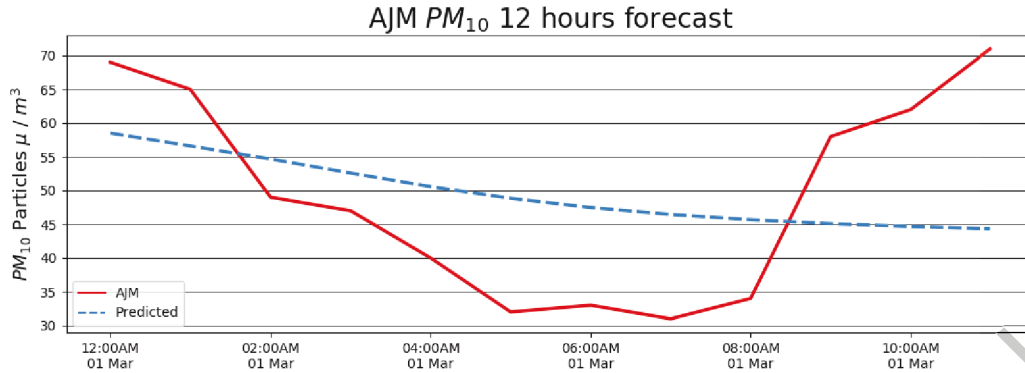


Figure 4.1: LSTM prediction of 12 hours of  $PM_{10}$  for AJM site.

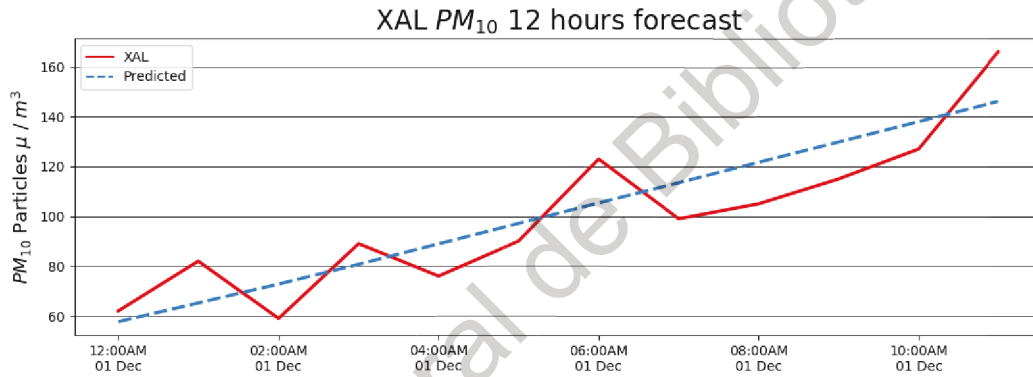


Figure 4.2: LSTM prediction of 12 hours of  $PM_{10}$  for XAL site.

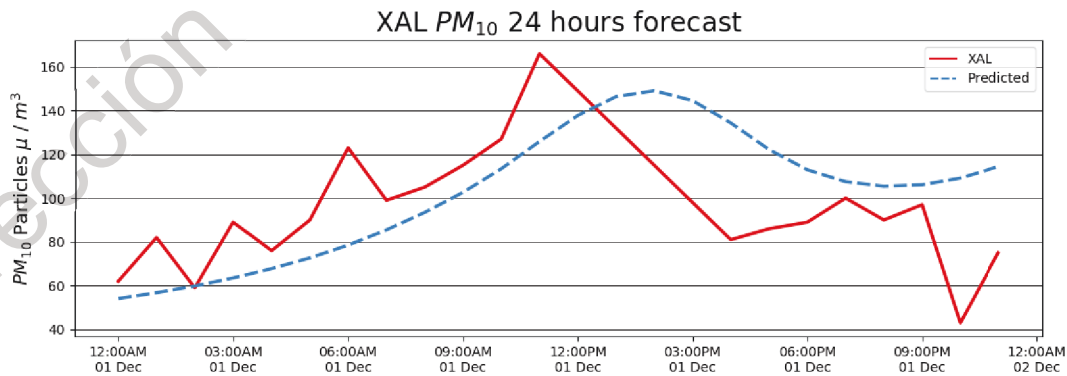


Figure 4.3: LSTM prediction of 24 hours of  $PM_{10}$  for BJU site.

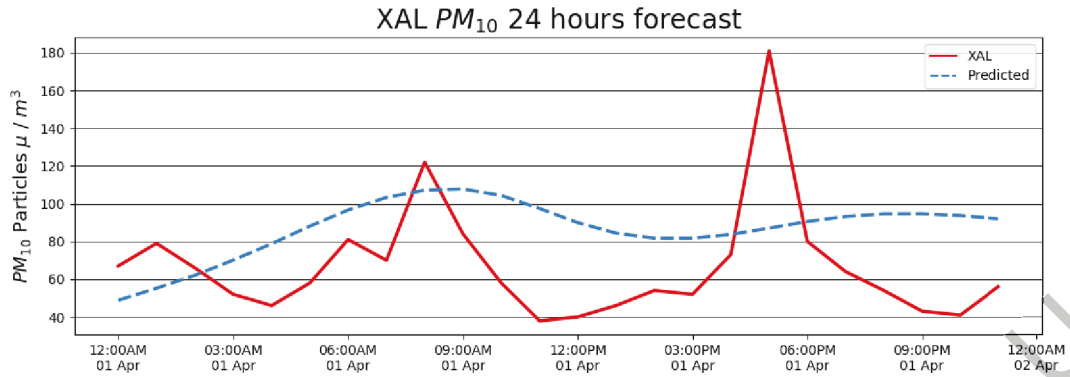


Figure 4.4: LSTM prediction of 24 hours of  $PM_{10}$  for XAL site.

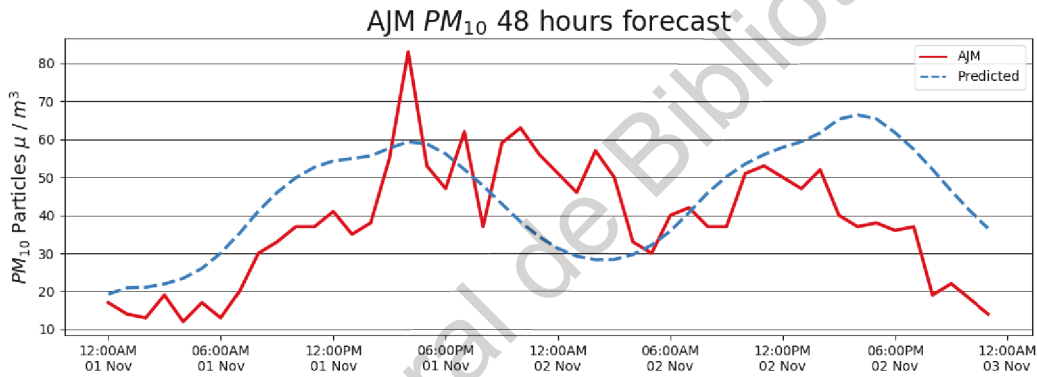


Figure 4.5: LSTM prediction of 48 hours of  $PM_{10}$  for AJM site.

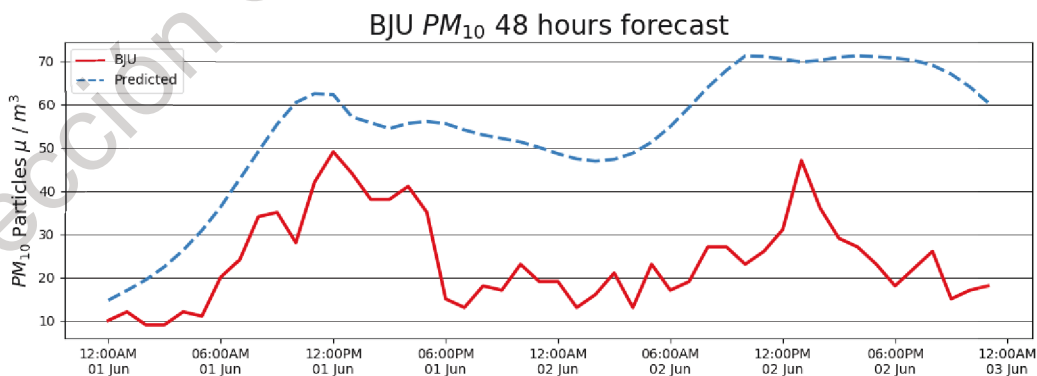


Figure 4.6: LSTM prediction of 48 hours of  $PM_{10}$  for MGH site.

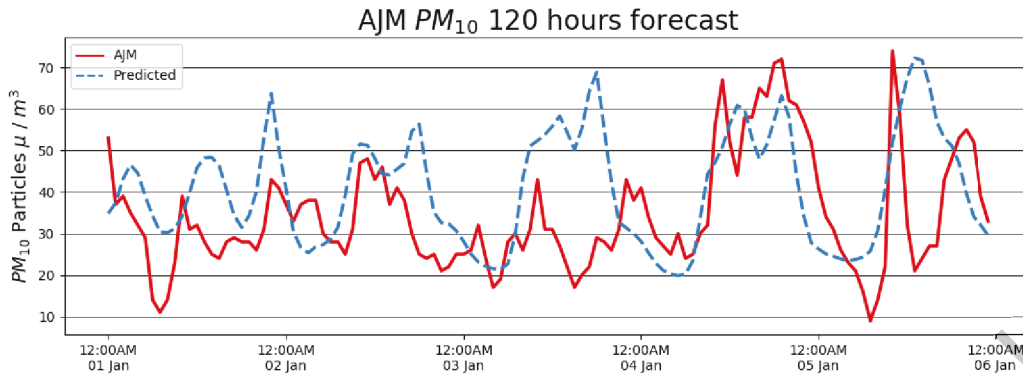


Figure 4.7: LSTM prediction of 120 hours of  $PM_{10}$  for AJM site.

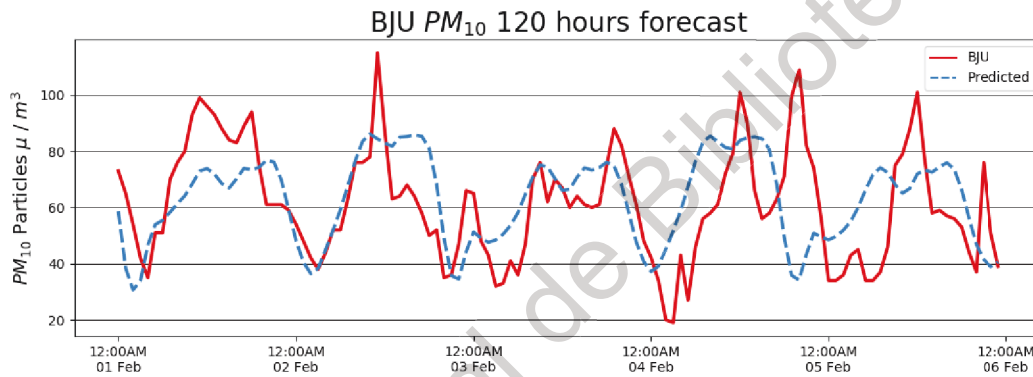


Figure 4.8: LSTM prediction of 120 hours of  $PM_{10}$  for BJU site.

### 4.1.2 Gated Recurrent Unit

The model was trained as stated in the methodology section. For GRU model,  $PM_{10}$  particle prediction did not have significant differences with the LSTM model, due to the similarities in the architecture and activation functions.

For 12h predictions, the GRU model had a good performance when forecasting. The results are shown in figures 4.9 and 4.10.

In 24h predictions the RMSE slightly increased compared to the 12h predictions. Figures 4.11 followed the tendency of the real data but did not match all the points, so the model could not deal with the real variation of  $PM_{10}$  particles, although the RMSE is low considering the range of the data mentioned in 3.4. Figure 4.12 shows also good results for 24h predictions, having some discrepancies in the middle compared to the real data.

For 48 hour predictions figure 4.13 shows how the model had a good prediction mostly at the end, but it missed several outliers too, figure 4.14 had a lower deviation but was not either as good as shorter predictions. Finally 120h predictions had very good results considering the complexity of long term predictions. Figure 4.15 shows really good results for the first 60 hours, but then the data increased much higher than the model predicted. Figure 4.16 seems to follow the tendency, but the value higher than  $400 \mu/m^3$  caused a huge RMSE compared to shorter predictions.

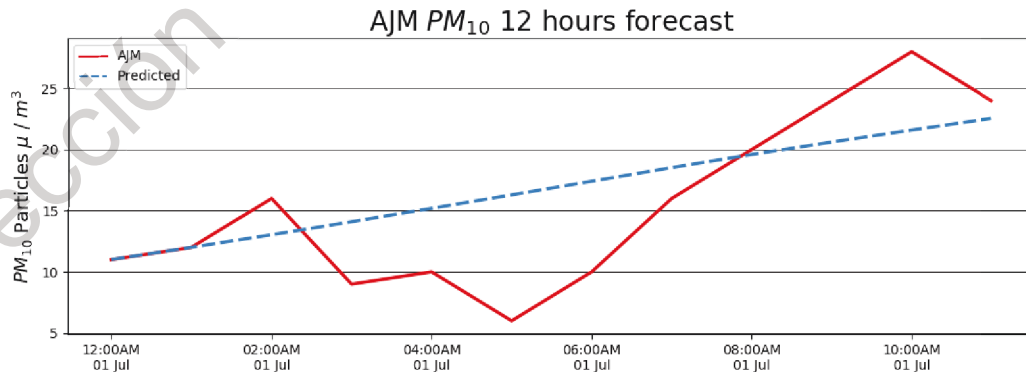


Figure 4.9: GRU prediction of 12 hours of  $PM_{10}$  for AJM site.

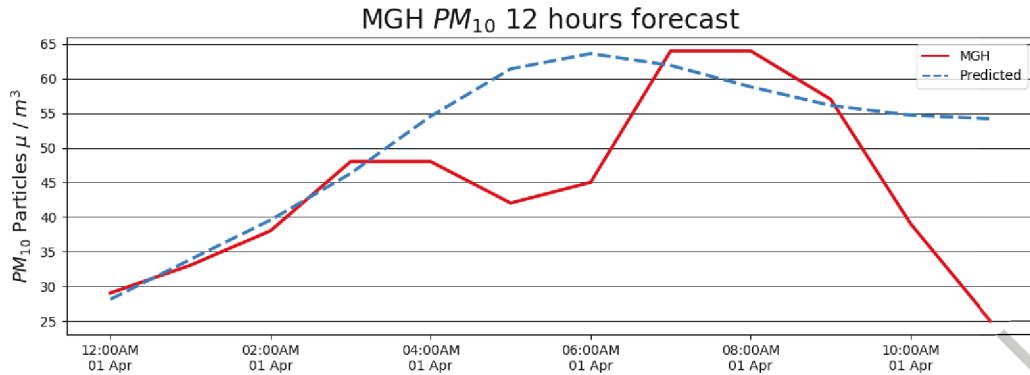


Figure 4.10: GRU prediction of 12 hours of  $PM_{10}$  for MGH site.

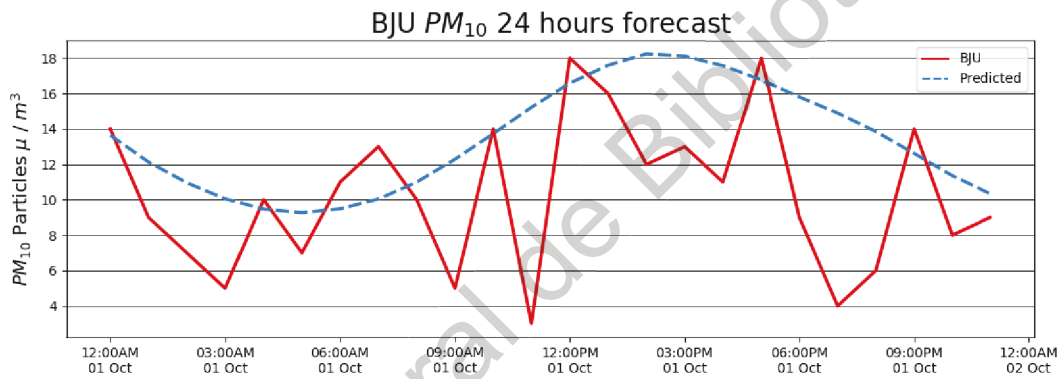


Figure 4.11: GRU prediction of 24 hours of  $PM_{10}$  for BJU site.

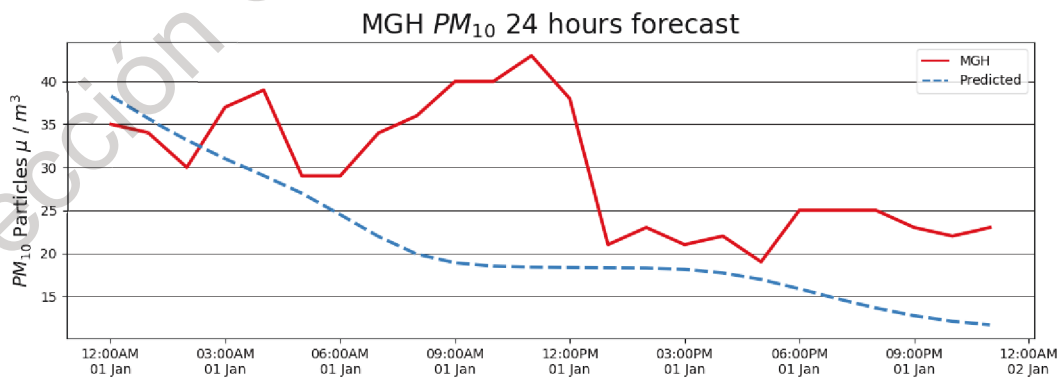


Figure 4.12: GRU prediction of 24 hours of  $PM_{10}$  for MGH site



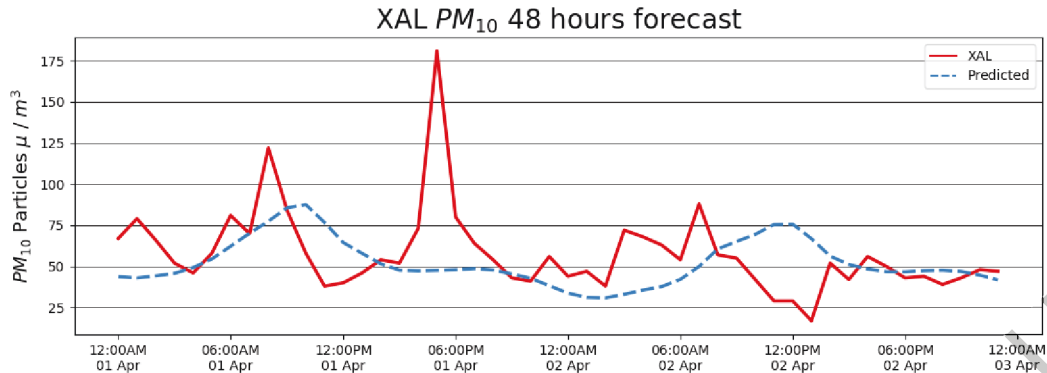


Figure 4.13: GRU prediction of 48 hours of  $PM_{10}$  for XAL site.

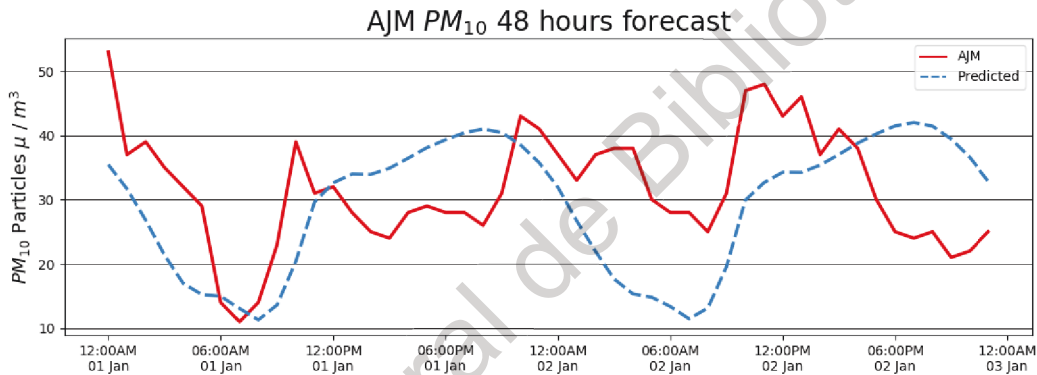


Figure 4.14: GRU prediction of 48 hours of  $PM_{10}$  for AJM site.

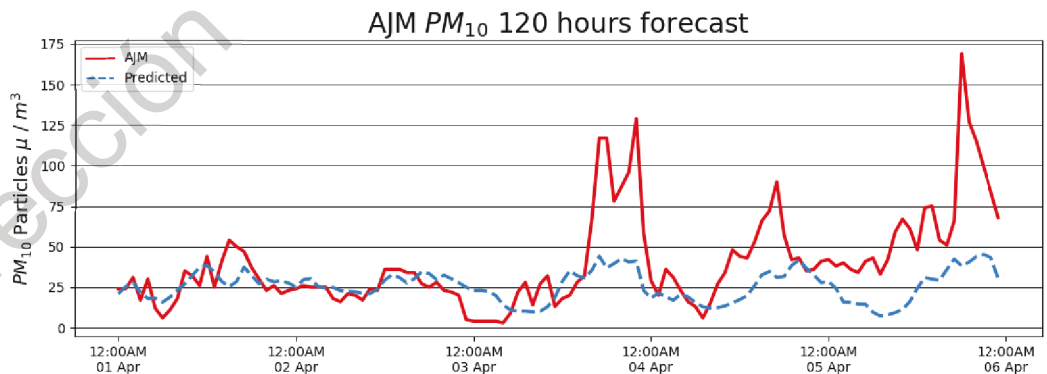


Figure 4.15: GRU prediction of 120 hours of  $PM_{10}$  for AJM site.

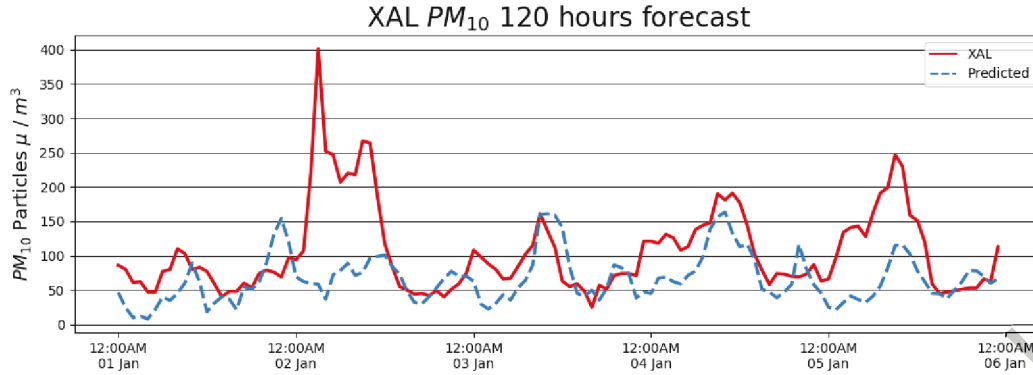


Figure 4.16: GRU prediction of 120 hours of  $PM_{10}$  for XAL site.

### 4.1.3 Error Estimation

The RMSE in LSTM model was low considering the mean standard deviation of 66.38 previously mentioned in table 3.4, the best result was for a batch length of 24 hours in the AJM station, with a RMSE of 1.36, the largest RMSE was 75.79 for a batch length of 120 hours in the XAL station. For GRU model the RMSE was greater in all cases, having as lower RMSE 3.09 for 48h predictions in the AJM station and 107.25 as maximum value for 12h prediction in XAL station.

Tables 4.1 and 4.1 shows the maximum, minimum and maximum value for each station and each length of prediction time.

Table 4.1: LSTM Root Mean Square Error in  $PM_{10}$   $\mu/m^3$ .

Hours	12			24			48			120		
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
AJM	3.64	40.42	13.30	1.36	33.69	19.08	8.21	41.40	18.31	14.58	35.86	20.20
BJU	8.38	55.15	18.86	8.03	45.60	20.81	14.31	39.54	24.46	8.11	38.47	21.98
MGH	7.38	28.41	15.06	3.24	28.89	16.98	2.03	42.02	16.81	2.13	23.19	16.85
XAL	6.84	84.51	31.04	3.04	91.11	43.22	46.19	46.19	46.19	26.98	75.79	51.16

Table 4.2: GRU Root Mean Square Error in  $PM_{10}$   $\mu/m^3$ .

Hours	12			24			48			120		
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
AJM	4.86	31.51	16.32	9.91	66.78	20.95	3.09	39.56	16.67	14.48	52.47	24.10
BJU	8.64	64.06	20.07	5.11	45.71	21.01	11.23	77.64	26.80	7.36	36.02	23.28
MGH	4.11	37.18	17.85	7.53	24.37	15.83	5.43	30.09	18.56	9.92	26.76	19.47
XAL	3.77	107.25	46.82	6.44	103.69	47.36	5.59	81.20	44.09	71.38	71.38	71.38

Figures 4.17 and 4.18 shows the mean RMSE and the variance in each of the stations. They show that in both LSTM and GRU models, AJM and MGH stations had better results than BJU and MGH stations, which implies that the training data for AJM and MGH was better.

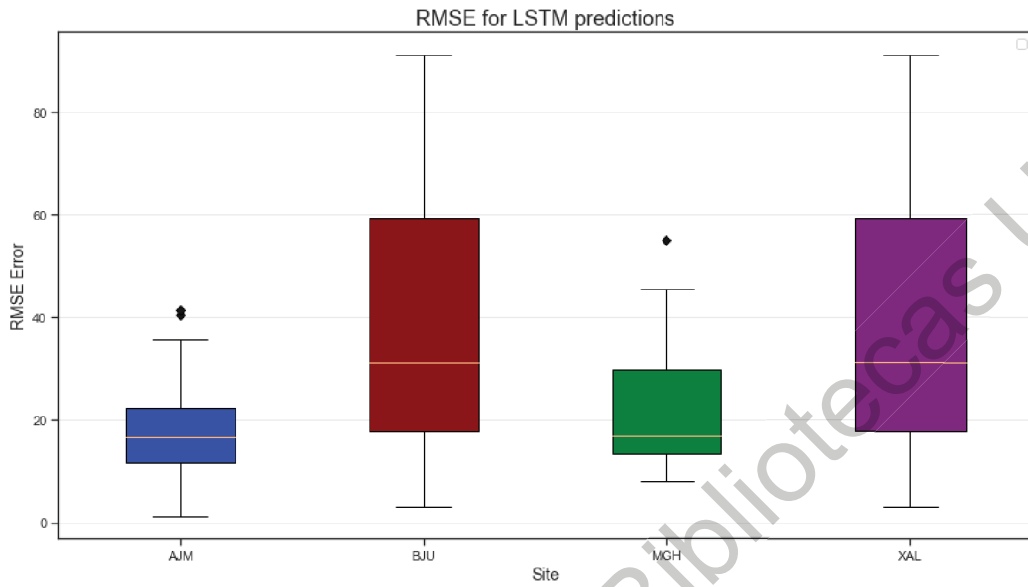


Figure 4.17: Boxplot of RMSE for LSTM model predictions by site.

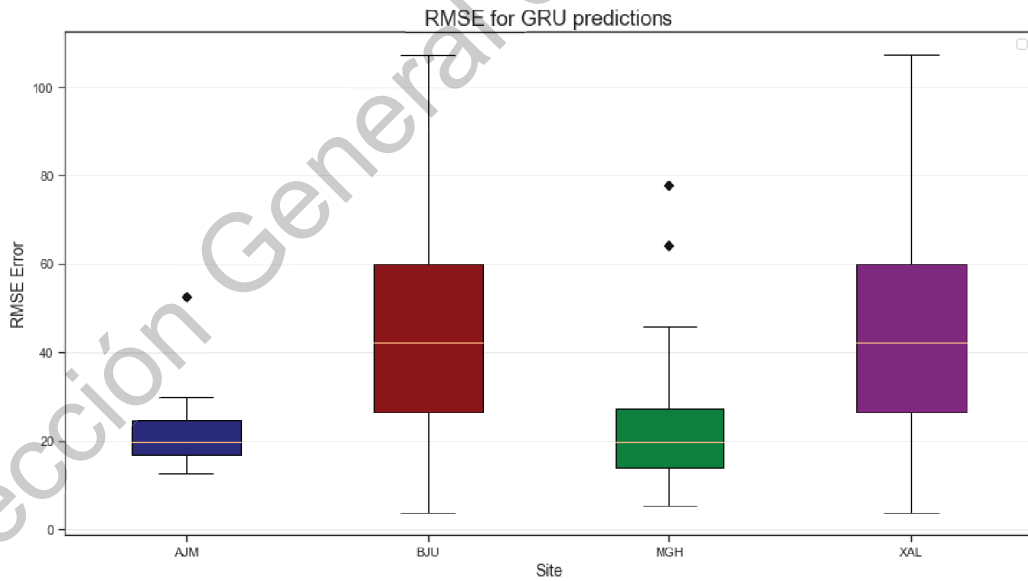


Figure 4.18: Boxplot of GRU for LSTM model predictions by site.

Figures 4.19 and 4.20 shows the mean RMSE and the variance for the different length prediction targets (12, 24, 48 and 120 hours). Where 48 and 120 hour predictions show more consistence in constrast to shorter predictions. 12 hour predictions showed the lowest mean RMSE but also had larger variance.

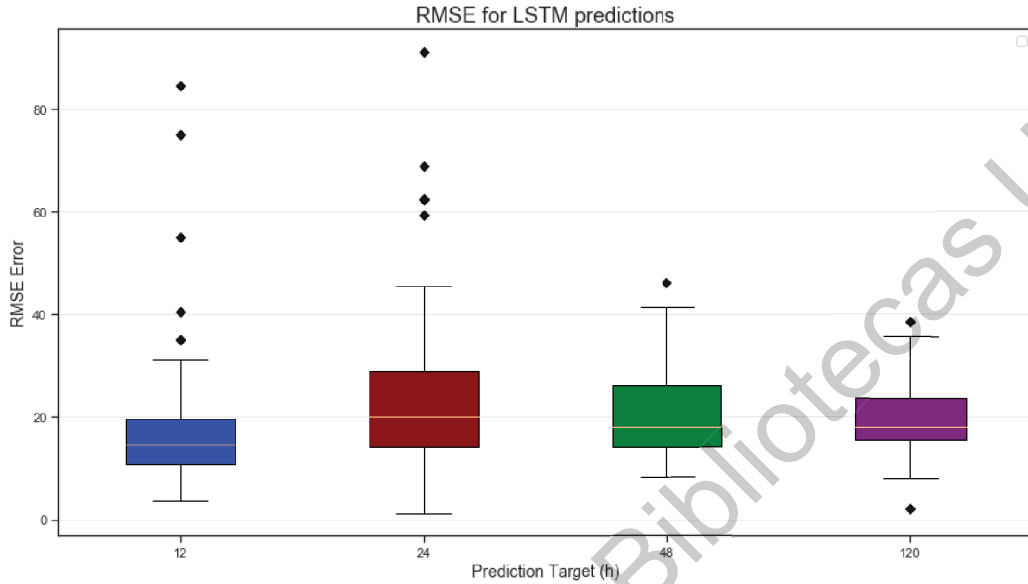


Figure 4.19: Boxplot of RMSE for LSTM predictions by prediction target length.

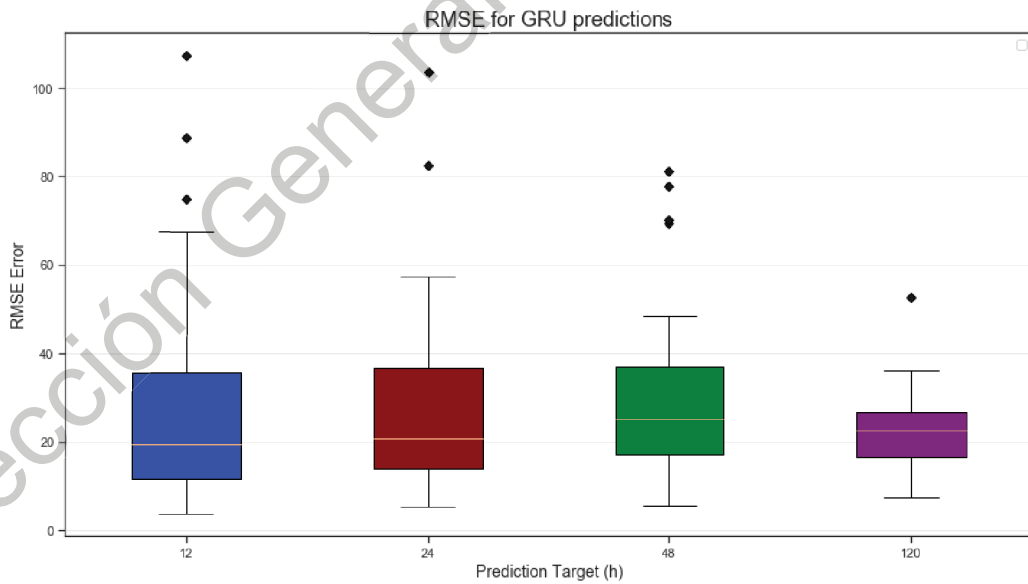


Figure 4.20: Boxplot of RMSE for GRU predictions by prediction target length.

The monthly prediction had notable variations due to the missing values in the training data. The missing values were replaced by a linear regression, which generated flat signals within the training data sets, increasing the RMSE when forecasting.

For both models GRU and LSTM, July and September had a significant better accuracy than the rest of the months, while May had a lot of variation and greater RMSE (see figures 4.21 and 4.22). This could result due to the missing values, but January, February and November had better results in LSTM model than GRU model using the same training data.

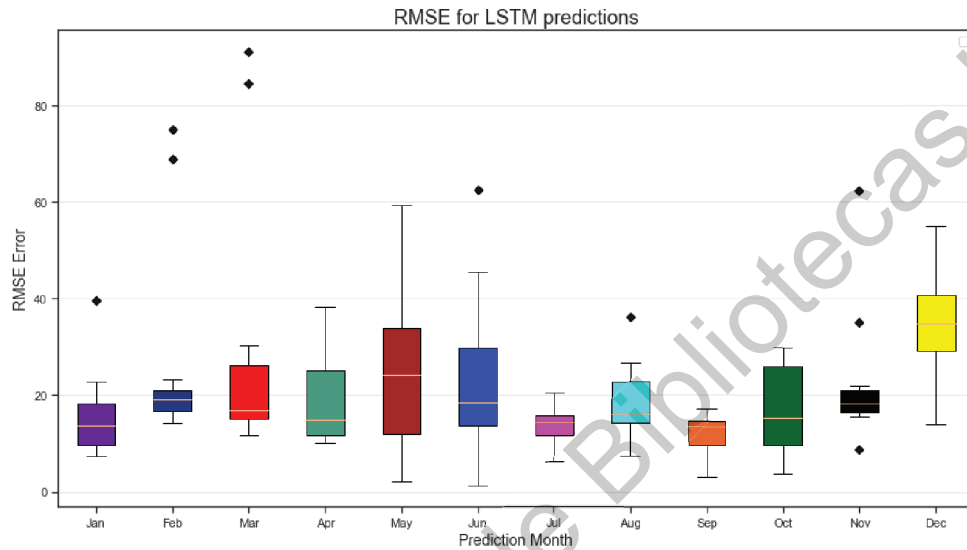


Figure 4.21: Boxplot of RMSE for LSTM predictions by month.

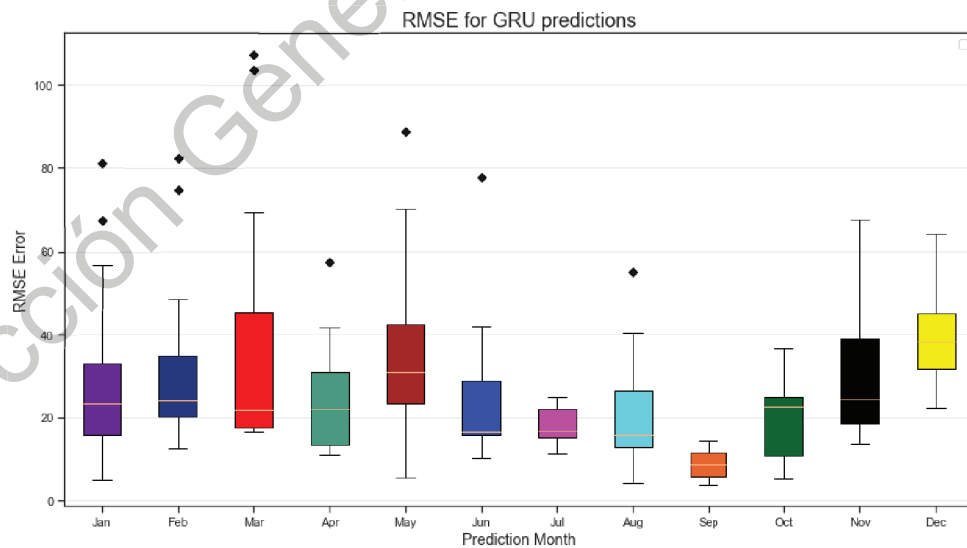


Figure 4.22: Boxplot of GRU for LSTM predictions by month.

## 4.2 Discussion

The results showed good predictions considering the mean standard deviation of  $26.76\mu/m^3$  (see table 3.4). The LSTM model had an overall better accuracy than the GRU model but it was not a significant difference. Hence both models can be used to predict  $PM_{10}$  particle time series.

It was found that long term predictions did not achieve good results as the results in short term predictions but considering the complexity of forecasting long range time series, the mean RMSEs were also lower than the standard deviation which implies that the model is robust and can handle  $PM_{10}$  particle time series for prediction.

## 4.3 Significance/Impact

This project has a direct impact on the environment, data science research and development. As it was stated in the survey section, air pollution has a direct impact on health. Being able to forecast this pollution allow us to take security measures and avoid breathing unhealthy air.

Also this research shows how feasible is to predict time series with aperiodic behaviour, which means that this models could be applied to any other time serie found in natural phenomena, physics among others.

## 4.4 Future Work

This research will be published in a scientific journal and more airborne pollution signals could be used to test the models. Also the proposed GUI will improved by adding more airborne particles, not only  $PM_{10}$  particles, solving bugs with the interface and optimizing the processing time.

---

## Conclusion

The aim of this project was achieved successfully using deep learning algorithms to forecast time series, which is a important task nowadays that the internet of things I.O.T. is increasing popularity. This research contribuites directly to the engineering and development since it proves that RNN and its variations can be used to predict data when the neural network is trained with the proper data. Also the environmental application is getting more important every day due to the high levels of pollution gained in the last years.

With this research basic topics about machine learning and deep learning were learnt. Also the research of Physical phenomena attached with these techniques are quite important in the Physics Eneineering branch. Those works have shown direct interaction between engineering developments and Physics.

Finally, several skills were gained while doing this research. Python programming language result quite usefull to developpe the code, construct the DL-model, generate the GUI and create the figures. Git was used as version control and LateX was used to write this thesis document.

This project was a challenging work due to the extense computer science background, but combined with the statistics and math functions used within the model, and the wide range of applications of time series forecasting, this research combines the engineering and physics side of the Bachelors degree of Engineering Physics.

Dirección General de Bibliotecas UAQ



---

# Bibliography

- [Aceves-Fernandez et al., 2015] Aceves-Fernandez, M., Estrada, A., Pedraza-Ortega, J., Gorrostieta-Hurtado, E., and Tovar-Arriaga, S. (2015). Design and implementation of ant colony algorithms to enhance airborne pollution models. *International Journal of Environmental Science and Toxicology Research*, 3(2):22–28.
- [Ackley et al., 1988] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1988). Connectionist models and their implications: Readings from cognitive science. *Ablex Publishing Corp.*, pages 285–307.
- [Aguirre-Salado et al., 2017] Aguirre-Salado, A. I., Vaquera-Huerta, H., Aguirre-Salado, C. A., Reyes-Mora, S., Olvera-Cervantes, A. D., Lancho-Romero, G. A., and Soubervielle-Montalvo, C. (2017). Developing a hierarchical model for the spatial analysis of pm10 pollution extremes in the mexico city metropolitan area. *International Journal of Environmental Research and Public Health*, 14(7).
- [Berger et al., 1996] Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71.
- [Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.
- [Bzdok et al., 2018] Bzdok, D., Krzywinski, M., and Altman, N. (2018). Machine learning: Supervised methods, SVM and kNN. *Nature Methods*, pages 1–6.
- [Cho et al., 2014] Cho, K., van Merriënboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.
- [Chollet, 2017] Chollet, F. (2017). *Deep Learning with Python*. Manning Publications Company.
- [Chung et al., 2014] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.
- [De La Calleja and Fuentes, 2004] De La Calleja, J. and Fuentes, O. (2004). Machine learning and image analysis for morphological galaxy classification. *Monthly Notices of the Royal Astronomical Society*, 349(1):87–93.
- [Engelbrecht, 2007] Engelbrecht, A. P. (2007). *Computational Intelligence: An Introduction*. Wiley Publishing, 2nd edition.

- [Feynman and Cohen, 1956] Feynman, R. P. and Cohen, M. (1956). Energy spectrum of the excitations in liquid helium. *Phys. Rev.*, 102:1189–1204.
- [Hartigan and Wong, 1979] Hartigan, J. A. and Wong, M. A. (1979). A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108.
- [Heck and Salem, 2017] Heck, J. and Salem, F. M. (2017). Simplified minimal gated unit variations for recurrent neural networks. *CoRR*, abs/1701.03452.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.
- [Hohenberg and Kohn, 1964] Hohenberg, P. and Kohn, W. (1964). Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–B871.
- [Hosmer and Lemeshow, 2000] Hosmer, D. W. and Lemeshow, S. (2000). *Applied logistic regression*. John Wiley and Sons.
- [Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
- [Kukkonen et al., 2003] Kukkonen, J., Partanen, L., Karppinen, A., Ruuskanen, J., Junninen, H., Kolehmainen, M., Niska, H., Dorling, S., Chatterton, T., Foxall, R., and Cawley, G. (2003). Extensive evaluation of neural network models for the prediction of no2 and pm10 concentrations, compared with a deterministic modeling system and measurements in central helsinki. *Atmospheric Environment - ATMOS ENVIRON*, 37:4539–4550.
- [Li et al., 2015] Li, Z., Kermode, J. R., and De Vita, A. (2015). Molecular dynamics with on-the-fly machine learning of quantum-mechanical forces. *Phys. Rev. Lett.*, 114:096405.
- [Lillicrap et al., 2016] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y., editors, *ICLR*.
- [Långkvist et al., 2014] Långkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, pages 11–24.
- [Luo and Clark, 2019] Luo, D. and Clark, B. K. (2019). Backflow transformations via neural networks for quantum many-body wave functions. *Phys. Rev. Lett.*, 122:226401.
- [Martinez et al., 2014] Martinez, E., Aceves, M., Palma, R., Sotomayor, A., and Gorrostieta, E. (2014). Enhancement of a neuro-fuzzy models using ant colony optimization for the prediction level of co pollution. In *2014 13th Mexican International Conference on Artificial Intelligence*, pages 141–146.
- [NOM, 2005] NOM (2005). Criterios para evaluar la calidad del aire ambiente, con respecto a material particulado. Norma oficial mexicana, Secretaria de Salud, Mexico.

- [Pathak et al., 2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.
- [Payne et al., 1992] Payne, M. C., Teter, M. P., Allan, D. C., Arias, T. A., and Joannopoulos, J. D. (1992). Iterative minimization techniques for ab initio total-energy calculations: molecular dynamics and conjugate gradients. *Rev. Mod. Phys.*, 64:1045–1097.
- [SEDEMA, 2018] SEDEMA (2018). Red automática de monitoreo ambiental. *SEDEMA*. Available at <http://www.aire.cdmx.gob.mx/>, consulted july 2018.
- [Skrzypski et al., 2009] Skrzypski, J., Kaminski, K., Jach-Szakiel, E., and Kaminski, W. (2009). Application of artificial neural networks for classification and prediction of air quality classes. *WIT Transactions on Ecology and the Environment*, 127:219–228.
- [Smola and Schölkopf, 2003] Smola, A. J. and Schölkopf, B. (2003). A tutorial on support vector regression. Technical report, STATISTICS AND COMPUTING.
- [Torlai and Melko, 2016] Torlai, G. and Melko, R. G. (2016). Learning thermodynamics with boltzmann machines. *Phys. Rev. B*, 94:165134.
- [Zeno, 2014] Zeno, P. J. (2014). Visualization tool for gpgpu programming. *ASEE2014 Zone IConference*, abs/1701.03452.
- [Zhang, 2016] Zhang, Z. (2016). Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, 4(11):218–218. 27386492[pmid].
- [Zhou et al., 2016] Zhou, G.-B., Wu, J., Zhang, C.-L., and cheng Zhou, Z. (2016). Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13:226–234.
- [Zocca et al., 2017] Zocca, V., Spacagna, G., Slater, D., and Roelants, P. (2017). *Python Deep Learning*. Packt Publishing.

## .1 Appendix

### .1.1 Hadamard Product

For two matrices,  $A$   $B$  of the same dimension,  $m \times n$ , the Hadamard product,  $A \circ B$ , is a matrix, of the same dimension as the operands, with elements given by

$$A \circ B = A_{i,j} B_{i,j} \quad (1)$$

For matrices of different dimensions ( $m \times n$ ) and  $p \times q$ , where  $m \neq p$  or  $n \neq q$  or both) the Hadamard product is undefined.

### .1.2 Activation Functions

#### 1. Linear function

$$f_{AN}(net - \theta) = \lambda(net - \theta) \quad (2)$$

Where  $\lambda$  is a constant.

#### 2. Step function

$$f_{AN}(net - \theta) = \begin{cases} \gamma_1 & \text{if } net \geq \theta \\ \gamma_2 & \text{if } net < \theta \end{cases} \quad (3)$$

Usually, a binary output is produced for which  $\gamma_1 = 1$  and  $\gamma_2 = 0$ .

#### 3. Ramp function

$$f_{AN}(net - \theta) = \begin{cases} \gamma & \text{if } net \geq \epsilon \\ net & \text{if } -\epsilon < net - \theta < \epsilon \\ -\gamma & \text{if } net - \theta \leq -\epsilon \end{cases} \quad (4)$$

The ramp function is a combination of the linear and step functions.

#### 4. Sigmoid function

$$f_{AN}(net - \theta) = \frac{1}{1 + e^{-\lambda(net - \theta)}} \quad (5)$$

The sigmoid function is a continuous version of the ramp function, with  $f_{AN}(net) \in (0, 1)$ . The parameter  $\lambda$  controls the steepness of the function. Usually,  $\lambda = 1$ .

#### 5. Hyperbolic tangent

$$f_{AN}(net - \theta) = \frac{e^{\lambda(net - \theta)} - e^{-\lambda(net - \theta)}}{e^{\lambda(net - \theta)} + e^{-\lambda(net - \theta)}} \quad (6)$$

The output is in the range  $(-1, 1)$ .

#### 6. Gaussian function

$$f_{AN}(net - \theta) = e^{-\frac{(net - \theta)^2}{\sigma^2}} \quad (7)$$

Where  $net - \theta$  is the mean and  $\sigma$  the standard deviation of the Gaussian distribution.