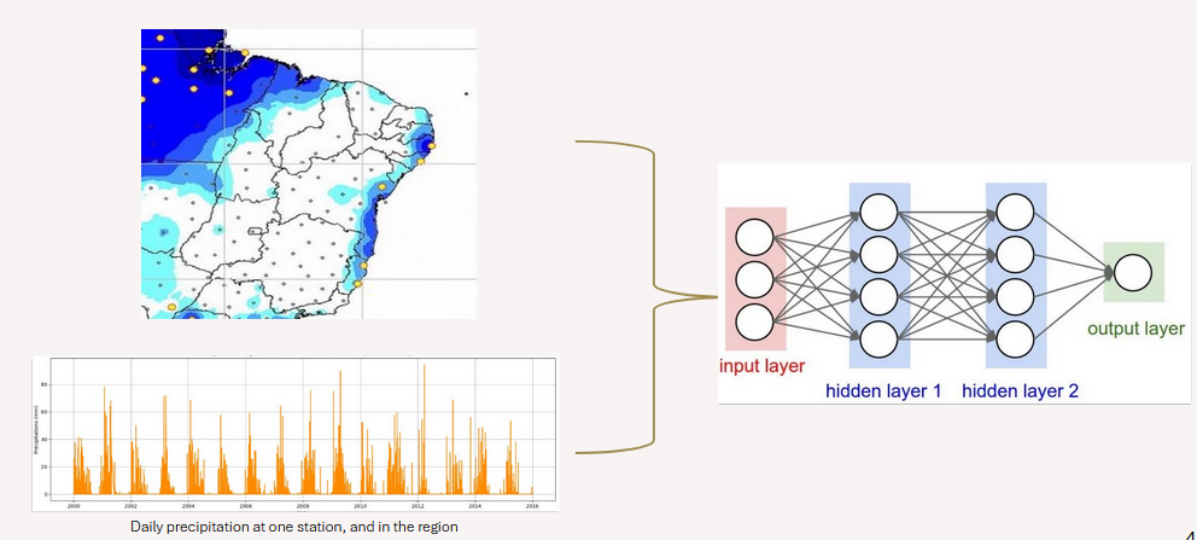
FINAL YEAR PROJECT
Machine Learning / Deep Learning / Oceanography - Meteorology

PFE Report

Using Deep Learning Models to Forecast Oceanic Variables and Rainfall in the Eastern Northeast Brazil

Author : Arthur Coron
Specialization : Master's in Mobile Robotics and Marine Applications, year 2024-2025



Supervisors : Nathalie Lefevre, Carlos Mejia
Institution : LOCEAN Laboratory, Sorbonne University

Table des matières

1 Introduction

1.1 Context

It is 05 :45 UTC (Coordinated Universal Time) in mid-October 2022 in Bologna, at the recently opened high-performance computing facility of the European Centre for Medium-Range Weather Forecasts (ECMWF).
For the past several hours, the Integrated Forecast System (IFS) has been running sophisticated calculations to forecast Earth's weather over the next days and weeks and its first predictions have just begun to be disseminated to users. This process repeats itself every 6 hours, every day, to supply the world with the best accurate weather forecasts available.

Weather forecasting, and especially precipitations, has always been a very difficult field to analyse and predict, and that is because the atmosphere is complex and constantly changing. Rain depends on many different variables like temperature, humidity, wind, and pressure, all interacting in 3D across time and space. One minute the sky can be clear, and the next minute there will be dark clouds and strong winds. This is especially true in big and diverse countries such as Brazil that possess a diverse range of meteorological regions from equatorial, tropical, semi-arid, subtropical, to coastal humid zones, making it one of the most climatically diverse countries in the world.

Traditionally, weather forecast systems use different mathematical equations in order to represent the physic laws that govern the movement of the atmosphere and the interactions of the systems that compose the climate system as truthfully as possible.
We have the Data-Driven Modelling (DDM) models which are not based on the physical laws we talked about, instead they are derived from learning algorithms using datas from previous observations.
It's purpose is then to reflect the patterns as accurately as possible, while still having a high level of generalization, i.e. the ability to provide an adequate prediction for unseen data.

On one hand, we have numerical weather forecasting system (with Numerical Weather Prediction NWP) made of numerous sub-processes such as the collection, modelling, and processing of atmospheric data from surface and altitude weather stations, satellites and weather radars.
The NWP systems, which combine mathematical models with observations, create datasets for climate monitoring and research, known as reanalysis data. The reanalysis data contains estimates of atmospheric parameters such as air temperature, pressure and wind at different altitudes, and surface parameters, such as precipitation, soil moisture content, height of ocean waves, and temperature of the sea surface. These estimates are produced for all locations on the planet and span several decades.

On the other hand, a new kind of DDM using Artificial Intelligence (AI) has been rapidly emerging for the last 20 years.
In particular, for extreme weather events, several approaches have applied artificial intelligence techniques for prediction and classification of extreme rainfall and severe winds (Mcgovern et al., 2017), predict cyclones, atmospheric rivers, frontal systems (cold / hot) (Liu et al., 2016) and prediction of heat waves (Iglesias et al., 2015). Weather forecasting based on DDM has series of challenges, such as abstraction of spatio-temporal information, high dimensionality, and imbalanced datasets.

1.2 The region of Brazil and precipitation complexity

1.2.1 The region of study

Across the world, intense rainfall events regularly disrupt human activities and livelihoods, often leading to natural disasters with severe human, economic, and environmental consequences.
These extreme weather phenomena are becoming more frequent and intense due to climate change, increasing the urgency for accurate precipitation forecasting.

Brazil is particularly vulnerable to the impacts of this intense seasonal precipitation, see [14]. During its rainy season, the country experiences episodes of heavy rainfall that are often highly variable in both space and time. These rains can result in different natural disasters such as flash floods, landslides, infrastructure damage, and widespread displacement, especially in densely populated or poorly planned urban areas. The combination of natural vulnerability and socio-economic exposure makes precipitation forecasting a critical issue for Brazil.

This project focuses on one of the most affected areas : the Eastern-Northeast Brazil (ENEB) region, which extends approximately from 47°W to 35°W longitude and from 18°S to 1°S latitude. This region is characterized by a strong spatial and temporal variability of precipitation, influenced by large-scale atmospheric phenomena such as the Intertropical Convergence Zone (ITCZ), sea surface temperature anomalies in the tropical Atlantic, and local topographic effects. The ENEB region frequently experiences both severe droughts and extreme rainfall events, making it a complex yet highly relevant area for improving the understanding and prediction of precipitation dynamics.

1.2.2 Rainfall complexity

Focusing on rainfall and precipitations, it is the result of an intricate set of atmospheric processes operating across multiple spatial and temporal scales at the same time. Its occurrence and intensity depend on the interaction of thermodynamics, fluid dynamics, and physical mechanisms within clouds.
At the macro scale, precipitation formation is influenced by atmospheric circulation patterns, frontal systems, convection, and orographic effects.
At the micro scale, processes such as condensation, coalescence of cloud droplets, and ice-phase interactions determine the size distribution of raindrops and the duration of rainfall events.

The variability of rainfall is amplified by numerous environmental factors. Temperature, humidity, or atmospheric pressure dictate the moisture-holding capacity of the air, while wind patterns influence the transport and convergence of moist air masses.
Moreover, rainfall is inherently intermittent and spatially heterogeneous, often varying significantly over distances of just a few kilometres and timescales of minutes to hours.

This multi-scale variability poses significant challenges for both measurement and prediction. Ground-based rain gauges provide point measurements, while radar and satellite observations offer spatial coverage but with their own uncertainties.
Numerical weather prediction models must integrate a multitude of interacting processes, many of which involve non-linear feedbacks and are sensitive to different conditions. As a result, accurate forecasting of rainfall, especially at fine spatial and temporal scales, remains one of the most challenging tasks in meteorology

1.3 Working environment

Throughout my internship, I was required to handle a large volume of data and run computationally intensive programs.

Therefore, it was essential to have a work environment suited to the needs of the project.

For this, I could count on the laboratory I have been working at for the whole internship, the LOCEAN.
The LOCEAN is a laboratory, member of the Pierre-Simon Laplace Institute (IPSL) whose research topic is the study of physical and biogeochemical processes that control the dynamics and variability of the ocean and climate for a better understanding of the climate system and its present, past and future evolution.
The laboratory is under the supervision of the CNRS, the National Museum of Natural History (MNHN), Sorbonne University (SU) and the Institute for Research and Development (IRD). The laboratory employs about 150 people divided into different teams : CYBIOM (study of ocean biogeochemical functioning), VALCO (long-term variability of climate and ocean), NEMO RD (numerical ocean model), VARCLIM (ocean and climate variability), VOG (ocean and sea ice variability), PROTEO (fine-scale ocean processes and interactions).

1.3.1 Local server

From the first day until the end, I had access to the local computing infrastructure of the LOCEAN laboratory, where I was based. The main server, named Acratopotes, is equipped with a GPU accessible remotely and hosts two NVIDIA graphics cards :

- A QUADRO RTX 5000 (16 GB),
- An RTX A5000 (24 GB).

This server was the most accessible within the laboratory's network and provided a lot of flexibility (e.g., remote SSH access and custom environment configurations). As a result, it became my primary working environment throughout the internship.

The LOCEAN network is well integrated into the wider infrastructure of the Institut Pierre-Simon Laplace (IPSL), offering possibilities of data transfer, shared resources, and access to internal and external computing platforms.

In parallel, I also had access to the ESPRI mesocenter (Ensemble de Services pour la Recherche à l'IPSL), which serves as the central computing and data infrastructure for the IPSL laboratory federation.
This platform offers different services, including data acquisition, long-term storage, and high-performance computing (HPC) capabilities.
Specifically, it provides two CPU-based compute clusters and a single GPU cluster intended for machine learning and simulation workloads.

Despite these resources, I made limited use of the ESPRI platform during my internship. The LOCEAN local infrastructure was sufficient for most of my needs, both in terms of computing power and data accessibility.
Furthermore, for tasks requiring intensive computation or dedicated GPU usage, I had the opportunity of using the national HPC resources provided by CNRS, which offered higher availability and better performance compared to the single shared GPU cluster at ESPRI.

1.3.2 National CNRS server

As I said, I was given the opportunity to access to the major computer center provided by the CNRS, called IDRIS (Institut du Développement et des Ressources en Informatique Scientifique).
In particular, I was being able to utilize the Jean Zay supercomputer, one of France's most powerful and modern HPC systems as of now [7] .

It is a hybrid architecture supercomputer hosted at IDRIS, designed to support both traditional scientific computing (HPC) and artificial intelligence/deep learning workloads. It is made of thousands of CPU cores and GPU

nodes, including NVIDIA V100 and A100 GPUs.

Despite Jean-Zay being especially well suited for important deep-learning tasks, there was two main setbacks for utilizing it.
The first one was getting the access rights to be able to use it. As it is a national infrastructure, the access was very limited and secured. Therefore, to be able to connect to it, my supervisor had to declare a project, with precise informations on the HPC requirement. Then, there was the local account creation and declaration (with the IT manager, creation of the IP connections,...). The whole process took around two month before I was able to use it.

On the other hand, IDRIS uses a special kind of command interpreter, called bash.
This interpreter is used to submit and execute jobs, following a very precise text architecture, see Listing 1.
Several things need to be specified, such as the SLURM partition, which defines the type of compute node I want to reserve, or the QoS (Quality of service), see the Table 16.

In the end, the local server was enough for the majority of the work, which is why I focused on this.

1.4 Objective of the internship

By using different data from the ENEB region, the objective is to improve the predictability of rainfall by developing deep learning models.
First, the purpose will be to collect data from different origins, we will also need to preprocess the data in order to transform them in a format more suitable and meaningful for future analysis and model training.

By trying, comparing and modifying different models, the final goal is to find the most accurate way to predict the precipitation in the region, and if possible extreme precipitation while taking into account spatio-temporal variability, what are the key oceanic and atmospheric variables to consider, and to what extent can these models outperform traditional forecasting approaches.

2 Data

2.1 Data sources

Data collection is a crucial component of projects such as mine, a fact I came to fully understand during this internship.
The meteorological data used in this study originate from two distinct sources : the Brazilian National Institute of Meteorology (INMET) and the ERA5 reanalysis dataset produced by the European Centre for Medium-Range Weather Forecasts (ECMWF).

INMET provides high-quality *local* observations from its network of meteorological stations distributed across Brazil, offering direct measurements of precipitation and surface atmospheric variables with high temporal resolution (hourly).
Historical rainfall measurements from INMET have been systematically recorded since 1961, providing more than six decades of continuous ground-based data for a total of 710 weather stations (more than a 100 in the northern east part of Brazil).

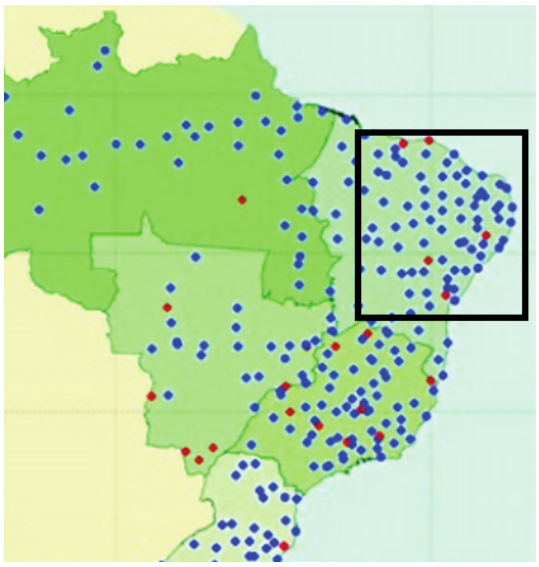


FIGURE 1 – Map of active INMET weather stations in Brazil in 2025

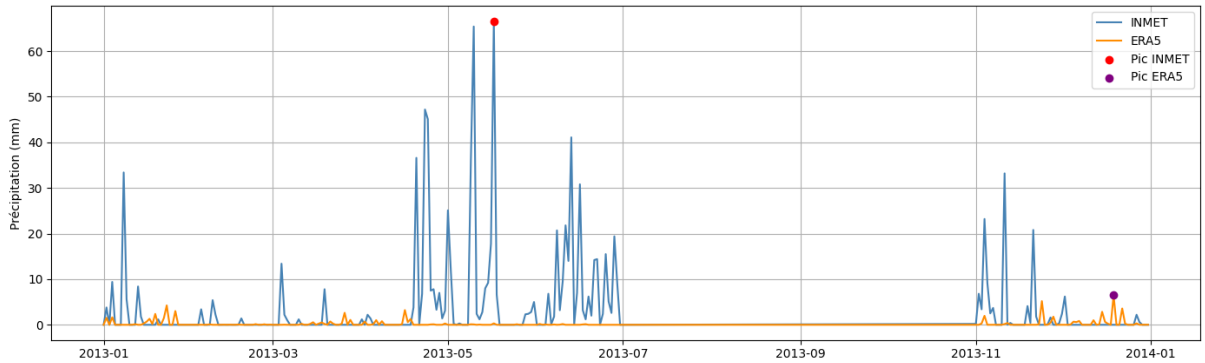
The main advantage of these observations lies in their accuracy and their ability to capture localised weather conditions, which ensures the reliability of values of the dataset.

ERA5, in contrast, delivers a globally consistent reanalysis of atmospheric conditions by assimilating a wide range of satellite, ground-based, and airborne measurements into a numerical weather prediction model.
ERA5 provides hourly estimates of a large number of atmospheric, land and oceanic climate variables. The data cover the Earth on a 31km grid and resolve the atmosphere using 137 levels from the surface up to a height of 80km from 1950 to the present, with hourly temporal resolution and high spatial resolution ($0.25^\circ \times 0.25^\circ$).
One of ERA5's key strengths is its ability to provide variables at multiple pressure levels, making it particularly valuable for representing the vertical structure of the atmosphere,an important factor in rainfall processes.

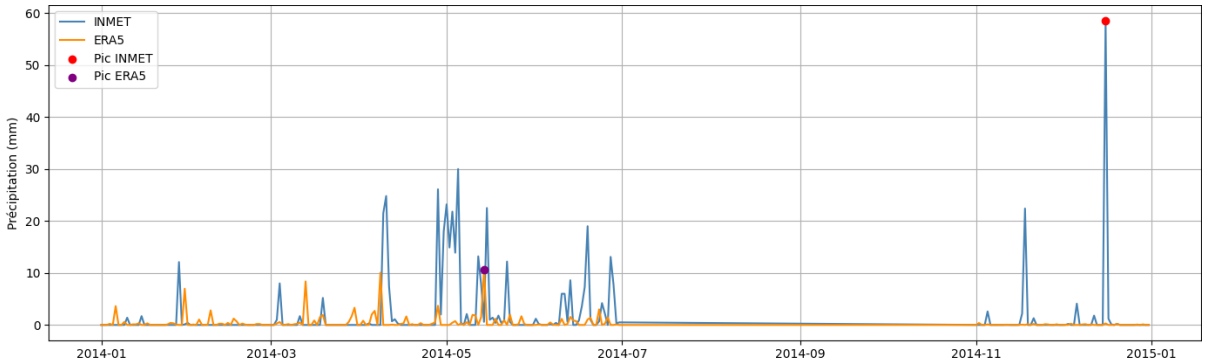
The primary difference between these two datasets lies in their precision and scope. On the one hand, INMET data offer fewer variables, but their ground-based, local nature ensures the highest possible accuracy for specific analysis.

On the other hand, ERA5 provides a full set of atmospheric variables with complete spatial coverage, but its reanalysis approach, integrating diverse observation types, makes it less precise when focusing on small-scale local areas, as is the case in this project.

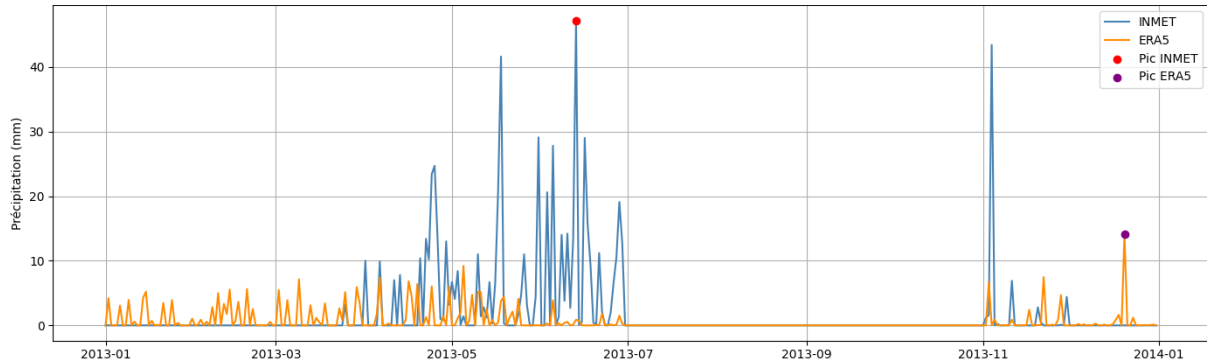
To show how big the difference is between the two sources, I made a comparison between data of precipitation over a year in three different stations from INMET, with ERA5 data of precipitation of the same year, in the pixel closest to the station :



(a) Précipitation annuelle à Maceio (-9.57°, -41.82°)



(b) Précipitation annuelle à Agua Brance (-4.03°, -37.21°)



(c) Précipitation annuelle à Porto de Pedras (-3.21°, -43.42°)

FIGURE 2 – Comparaison des précipitations récupérées d'INMET et de ERA5 pour 3 stations sur une période d'un an.

The analysis of the time series comparisons reveals very well the significant differences in terms of both intensity and distribution of precipitation.
In most cases, the daily totals recorded by INMET are substantially higher than those estimated by ERA5.
This discrepancy becomes even more pronounced when focusing on heavy precipitation events, which are responsible for severe weather occurrences in Brazil. The maximum precipitation values are noticeably underestimated by the ERA5 reanalysis, as evidenced by the peak maxima shown in each annual series.

Consequently, ERA5 struggles to capture local extremes, especially when they are highly localized in space or of short duration. Short, intense precipitation events, for example, could be poorly represented in ERA5 due to their localized nature.

It is also important to consider that a direct comparison between INMET and ERA5 data presents several significant methodological limitations. First, ERA5 data represent the average value over a pixel with a spatial resolution of approximately 30 km in this case, meaning they correspond to a relatively large area rather than a precise point. In contrast, INMET data are highly localized point measurements.

Second, there is some uncertainty regarding the temporal processing of the data. INMET timestamps are likely expressed in local time (often UTC–3 or UTC–4 depending on the region), whereas ERA5 reanalysis data are provided in Coordinated Universal Time (UTC). This time difference can cause a phase shift in the daily aggregation of data : precipitation occurring in the local evening may be attributed to the following day in ERA5 data.

Finally, the INMET station is not located exactly at the center of the ERA5 pixel, as can be seen by comparing their geographic coordinates, which adds further uncertainty to the comparison.

Moreover, INMET data have a major disadvantage compared to ERA5 data : they are point-based and entirely dependent on the stations and their proper functioning. We will examine later to what extent this has caused issues.

2.2 data description

As said before, precipitation is a complex atmospheric phenomenon influenced by many different meteorological variables.
Numerous variables related to moisture, temperature, wind, and pressure exist across different atmospheric layers, each playing a significant role in the formation and evolution of precipitation events.
Throughout this internship, I had to choose the most optimal ones to keep and use for the precipitation forecast, and these are the selected variables used to capture key aspects of these processes.

First, we have the surface variables :

- **Mean Sea Level Pressure (MSL) :** Reflects large-scale atmospheric circulation patterns and synoptic weather systems such as cyclones and anticyclones. These systems influence vertical air motions and modulate the spatial and temporal distribution of rainfall.
- **10 m u- and v-wind Components :** Measure the horizontal wind velocity near the surface in the zonal (u) and meridional (v) directions. These components are key for analyzing moisture transport, convergence zones that promote ascending motion, and surface wind patterns that can trigger or suppress precipitation.
- **2 m Air Temperature (T2M) :** Directly influences evaporation rates and thermal stability in the atmospheric boundary layer. Higher temperatures increase the air's capacity to hold moisture, thus affecting convective activity and precipitation processes.
- **Precipitation :** The target variable of this study, representing accumulated rainfall at the surface over a specified period. It directly measures the meteorological phenomena under investigation.

Then we have the atmospheric variables. These pressure levels were chosen to capture the vertical structure of the atmosphere from the upper troposphere (300 hPa) down to the near-surface layer (1000 hPa) :

- **Geopotential Height :** Indicates the height of a given pressure surface above mean sea level. It provides insight into large-scale atmospheric circulation, vertical stability, and the presence of features such as troughs and ridges that influence precipitation systems.
- **Temperature :** Describes the thermal structure at different altitudes, which is critical for assessing atmospheric stability and the potential for precipitation development.
- **Specific Humidity :** Measures the absolute amount of water vapour at a given level, directly impacting cloud formation and precipitation processes by quantifying moisture availability.
- **U and V Wind Components :** Represent the horizontal wind fields at various altitudes, crucial for identifying jet streams, moisture transport pathways, and vertical wind shear, all of which affect precipitation formation and intensity.
- **Relative Humidity :** Expresses the degree of saturation of the air. High relative humidity values are often necessary for cloud persistence and precipitation maintenance.

These linked interaction can be seen if I plot the a map of precipitation at one day and compare it to two other variables, such as the humidity at 850HPa and the geopotential at 500 HPa :

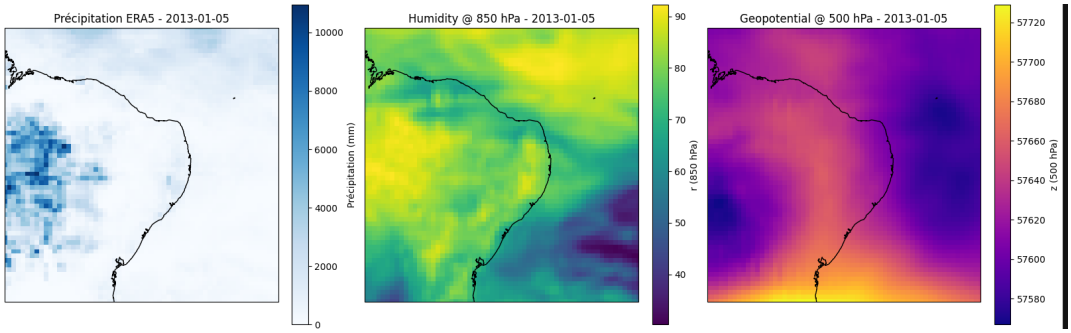


FIGURE 3 – Comparison between precipitation, specific humidity and geopotential, at the date 01/05/2013

This comparison of the spatial maps of precipitation, humidity, and geopotential height reveals a clear interconnection between these variables.

Regions exhibiting enhanced precipitation consistently coincide with areas of increased humidity, reflecting the availability of it necessary for rainfall formation.
Simultaneously, these precipitation zones are associated with a decrease in geopotential height, indicating the presence of low-pressure systems that promote upward motion and atmospheric instability, driving precipitation events.

In the end, I had 35 different physical variables, including the precipitation for this internship.
We chose not to take more, as it would increase the amount of data for processing and computing, but we have to take into account that there are more variables that influence or are influenced by the precipitation, such as total column water vapour or the potential vorticity, that could also be interesting to use to understand the patterns of precipitation.

3 Methods

3.1 Deep learning introduction

Deep learning is a subset of machine learning that models complex patterns in data using multi-layered neural networks [12, 13].
Unlike traditional machine learning algorithms, which often rely on classic features or simpler statistical relationships, deep learning models can automatically discover complicated, nonlinear relationships by optimizing thousands (or millions) of parameters through a process called *training*.

At its core, a deep learning model consists of layers of interconnected nodes, called *neurons*, each performing simple computations. These neurons transform input data through a series of weighted sums followed by nonlinear transformations, enabling the model to learn hierarchical representations, from simple features in early layers to more abstract concepts in deeper layers.

A key component enabling deep networks to model nonlinear functions is the *activation function* [3].
After each neuron computes a weighted sum of its inputs, the activation function introduces nonlinearity, allowing the network to approximate complex mappings beyond what linear functions could usually do.
Common activation functions include :

- **ReLU (Rectified Linear Unit) :** Outputs zero for negative inputs and passes positive inputs unchanged, promoting efficient representations [5].
- **Sigmoid :** Maps inputs to values between 0 and 1, commonly used in binary classification.
- **Tanh :** Similar to sigmoid but outputs values between -1 and 1, centering the data.

The training process involves optimizing the model's parameters to minimize the distance between the model's predictions and the true outcomes. This distance is quantified by a *loss function*.

A *loss function* is a mathematical function that measures the difference between predicted values and ground truth. The goal of training is to minimize this loss across the dataset, thereby improving the model's predictive accuracy on new, unseen data.
For regression problems, common loss functions include :

- **Mean Squared Error (MSE) :**

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

  MSE squares the differences between predicted values $\hat{y}_i$ and true values y_i, penalizing larger errors. The mean normalizes this sum by the number of samples. The square root of MSE, called *Root Mean Squared Error (RMSE)*, shares the same units as the target variable, making it more interpretable.

- **Mean Absolute Error (MAE) :**

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

  MAE computes the average magnitude of errors linearly by taking absolute differences. Unlike MSE, it treats all errors with equal weight and is less sensitive to outliers.

In addition to loss functions, model performance is often evaluated using statistical metrics such as the *coefficient of determination*, R^2, which measures how well the model explains the variance in the target variable :

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$$

where :

— SSE (Sum of Squared Errors) is the sum of squared differences between actual and predicted values, representing the unexplained variance.
— SST (Total Sum of Squares) is the total variance in the target, calculated by summing the squared differences between actual values and their mean.

Interpretation of R^2 [2] :
— $R^2 = 1$: The model perfectly explains all variability in the data.
— $R^2 = 0$: The model does no better than simply predicting the mean.
— $R^2 < 0$: The model performs worse than predicting the mean, indicating poor fit.

This project focuses on nonlinear regression, aiming to uncover complex relationships between meteorological input variables and precipitation outputs. These relationships are too intricate to be captured by simple linear models, so deep learning's ability to model nonlinearities makes it an appropriate choice.

3.2 Models of deep learning

The complex spatiotemporal dynamics of atmospheric processes make rainfall prediction a challenging problem, particularly for classic feedforward networks.
Different deep learning architectures have been developed to model such data, each with unique strengths to capture either temporal or spatial dependencies, or both.

Below, we review some of the most efficient model types used in precipitation forecasting.

3.2.1 Recurrent Neural Networks (RNNs)

Unlike traditional feedforward neural networks that treat each input independently, Recurrent Neural Networks (RNNs) are designed to process sequential data by maintaining a hidden state that evolves over time [1, 9].
This hidden state allows RNNs to "remember" information from previous time steps, which is very helpful for time series forecasting where past weather conditions strongly influence future rainfall.

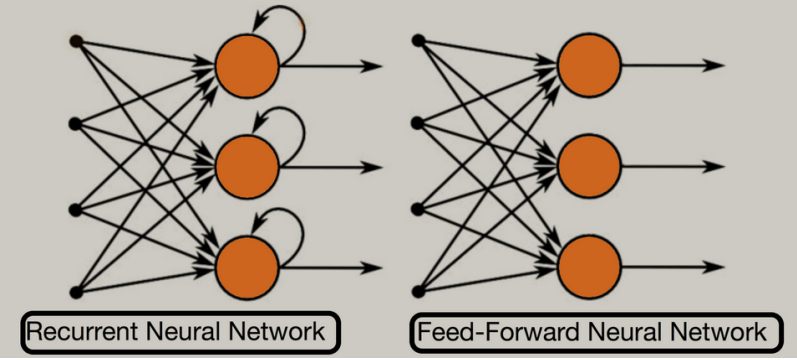


FIGURE 4 – Working of a RNN compared to a feed-forward neural network, by k21Academy.

To illustrate, we can think of it like reading a sentence : the meaning of each word depends on its context, i.e., the preceding words. Similarly, RNNs use loops in their architecture to process sequences.
At time step t, the input x_t is combined with the hidden state from the previous step h_{t-1} to produce a new hidden state h_t, which encapsulates both current and past information.

However, classic RNNs face a fundamental problem : they struggle to learn long-term dependencies due to vanishing or exploding gradients during training [6]. This limitation is an important problem as it is often required

in meteorological time series.

To address the limitations of standard RNNs, more advanced recurrent architectures were developed, notably Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs) [9, 10].

LSTMs enhance RNNs by introducing a memory cell and three gating mechanisms (forget, input, and output gates) that regulate the flow of information through the network :
— The **forget gate** decides which information to remove from the cell state.
— The **input gate** controls which new information to add to the cell state.
— The **output gate** determines which part of the cell state to output as the hidden state.
These gates are controlled by nonlinear activation functions such as sigmoid and tanh, enabling LSTMs to keep relevant information over long periods and effectively control the vanishing gradient problem encountered in classic RNNs [6].
This capability allows LSTMs to remember important atmospheric trends, weather cycles, and precipitation events spanning days or even weeks, thereby improving the accuracy of rainfall forecasts.

Gated Recurrent Units (GRUs) provide a simplified alternative to LSTMs, but just as effective, and often more energy efficient [10]. They combine the forget and input gates into a single update gate and merge the cell state and hidden state, resulting in a more streamlined architecture with fewer parameters.
Despite their simpler design, GRUs often achieve comparable performance to LSTMs, especially when data or computational resources are limited. Their efficiency made GRUs an attractive choice for this project with time series precipitation forecasting.

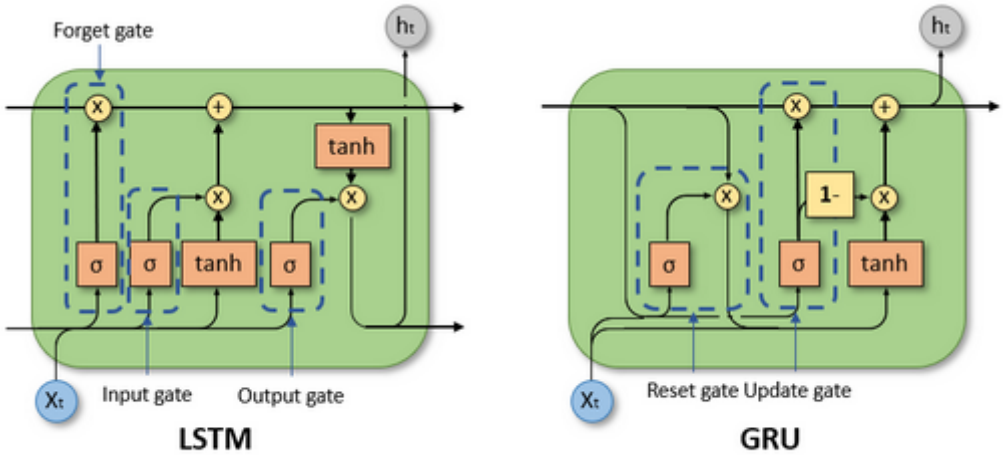


FIGURE 5 – Architecture of a GRU and a LSTM model, Suvankar Maity

However, both LSTMs and GRUs primarily focus on modeling temporal dependencies and do not explicitly capture spatial relationships present in meteorological data.

3.2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are designed to capture spatial patterns [4].
In meteorology, input data often come as 2D grids (e.g., satellite or radar imagery). CNNs use convolutional filters to extract features like cloud formations, rain bands, and storm structures, spatial information critical for rainfall

prediction.

CNNs alone is lacking temporal modeling, but when combined with recurrent layers, they can jointly learn spatial and temporal features.

3.2.3 Other interesting neural networks

Recent advances have introduced more sophisticated architectures that can handle model complex meteorological relationships with more flexibly.

First we have Graph Neural Networks (GNNs) that represent data as nodes connected by edges, allowing flexible modeling of spatial relationships that are not restricted to fixed grids.
For weather data, locations (e.g., weather stations or grid cells) are nodes, and edges represent spatial relationships or atmospheric flows. GNNs learn how information propagates through this graph, effectively capturing spatial dependencies that vary dynamically.

We also have Transformers, originally developed for natural language processing, they have revolutionized sequential modeling by using attention mechanisms.
Unlike RNNs, transformers do not rely on sequential processing but compute relationships between all elements of the input simultaneously, making them very effective at capturing long-range dependencies. This makes transformers a promising model for meteorological forecasting.

While these model looks promising, their complexity and computational necessity can also be a blockage if the resources are not there to follow.
It was the case looking at existing models using these architectures.

- FourCastNet,[11] : This model uses a transformer-based architecture to forecast global weather patterns, including precipitation. It achieves high accuracy by modeling complex atmospheric interactions at scale but requires massive computational resources.
- GraphCas, [8] : This model combines GNNs with transformer components to capture both spatial relationships in an adaptive graph structure and temporal dynamics via attention mechanisms. GraphCast has achieved impressive performance on precipitation forecasting by learning spatial flow and temporal evolution more flexibly.

While these architectures demonstrate lots of potential, they also come with significant challenges : training and deploying such models demands extremely high computational power and memory, which for me was a setback in this project.

3.3 Data preprocessing

In meteorological forecasting tasks using deep learning, one critical pre-processing step is data normalization. Meteorological variables are highly heterogeneous in both scale and distribution. For instance, air temperature typically ranges from -50°C to +50°C, atmospheric pressure spans around 900–1100 hPa, while precipitation values may remain near zero for extended periods, punctuated by rare extreme events. Without proper normalization, such disparities can distort the optimization dynamics of deep learning models, leading to unstable training and poor generalization.

For this project, we handle variables that are highly heterogeneous in both scale and distribution.
For instance we have atmospheric temperature that can range from -50°C to +50°C, atmospheric pressure that go from 900 hPa to 1100 Hpa, while precipitation values can remain between 0 and 1 mm for a certain period of time. Without a proper normalization, such disparities will impact the dynamic of deep learning models by putting weights to some variables with strong variation or intensities instead of having a more continuous spatial dynamic.
Therefore, if one feature dominates numerically, the model will become biased towards optimizing mainly that feature, forgetting the others.

Therefore, we applied a normalization strategy that was the most fitting to our project.
This normalization is called the **Z-score normalization**, also called the StandardScaler.
It transforms the features by subtracting the mean and dividing by the standard deviation :

$$z = \frac{x - \mu}{\sigma}$$

where x is the feature value, μ is the mean, and σ is the standard deviation.

The Z-score normalization is useful for us as it preserves Gaussian-like distributions, which is advantageous for gradient-based learning, as it helps maintain a stable learning rate across features. In recurrent networks (like GRUs used in my project), z-score normalized inputs prevent biased gradient accumulation during backpropagation through time (BPTT) which is another advantage of it.

3.4 Spatial interpolation of INMET precipitations

One of the main challenges when you deal with meteorology data is to work on data from different sources, weather stations, satellites, sensor, or reanalysis data. These data observations are inherently limited to specific points, with spatial and variability that can change from one data to another.

In my case for this precipitation forecasting project, I was using on one hand ERA5 reanalysis data, which are provided in the form of spatial grids resembling images.

However, the precipitation observations from INMET are recorded as point measurements at individual weather stations, resulting in data that are always sparse and unevenly distributed across the geographical area of interest, (see Figure 1).
To integrate INMET precipitation data with the ERA5 dataset, it is then necessary to transform the station-based observations into a spatial grid format compatible with ERA5.

This is where spatial interpolation becomes essential.
The spatial interpolation process converts the discrete precipitation measurements into continuous gridded data, aligning with the ERA5 spatial resolution and enabling joint analysis.
By doing so, the datasets become coherent and directly comparable, facilitating the application of deep learning models that rely on spatially structured inputs. This approach ensures the scientific rigor and consistency needed to improve the accuracy and robustness of the precipitation prediction models.

At its core, it is a mathematical process. We start with a set of known data points, each with a geographical value (often latitude and longitude coordinates) and a value (for example precipitation amount), and we apply a model that estimates values in between those points.
Hence, from a set of points in the space, we create a continuous surface that covers the area of interest and gives a value for every cell in it.

Interpolation is not about trying to discover the perfect value, it is about making the best educated guess possible based on the data we have. It is still uncertain, and the choice of method can significantly impact the accuracy and usability of the results.

For my project, I had to use spatial interpolation as I had local precipitation values from weather stations located in the region of Brazil I studied.
The model chosen for the project was the Inverse Distance Weighting (IDW) model that is widely used in meteo-

rology forecasting.
The core idea of its method is that the influence of the known points of the unknown points decreases with the distance. In the other words, nearby data points carry more weight than the distant ones when estimating a value at a new location. This is useful here because it reflects the typical spatial continuity of precipitation.

Mathematically, the IDW computes the unknown value as a weighted average of known values, where each weight decreases as the distance from the target point increases. The formula used is :

$$\hat{Z}(x_0) = \frac{\sum_{i=1}^{N} w_i(x_0) \cdot Z(x_i)}{\sum_{i=1}^{N} w_i(x_0)} \quad \text{where} \quad w_i(x_0) = \frac{1}{d(x_0, x_i)^p}$$

with :
— $\hat{Z}(x_0)$ is the estimated interpolated value (precipitation) at the point x_0,
— $Z(x_i)$ is the known value at the point x_i,
— $d(x_0, x_i)$ is the distance between the target x_0 and the known point x_i,
— p is the power parameter that controls how quickly influence decreases with distance,
— N is the number of surrounding points included in the computation.

The formula is a weighted average, where the weights are inversely proportional to the distance raised to the power p. This power parameter is therefore crucial, with it we can control how sharply the influence of a point decreases. A larger p means that closer points dominate more strongly, resulting in an interpolation that closely follows local values. Meanwhile, a smaller p allows more distant point to influence the estimate, which can lead to smoother and more generalized surfaces.

4 Results

4.1 First step with INMET data

As a preliminary step, I made first step with a simple GRU-based recurrent model using only daily precipitation values from INMET as both inputs and outputs.
As highlighted earlier, INMET measurements are considered more reliable than ERA5 reanalyses, as they directly result from ground-based observations and therefore provide more precise and realistic precipitation records. This makes them particularly suitable for an initial evaluation of deep learning approaches.

The purpose of this first experiment was mainly methodological : to become familiar with the behavior of recurrent neural networks (RNNs), and specifically GRUs, in the context of time series forecasting of precipitation.
The idea was to assess whether such a simple temporal setup, where future precipitation is predicted based solely on previous daily values, could already capture meaningful temporal dependencies.

4.1.1 Preprocessing of INMET data

Before training the GRU model, the INMET precipitation data were preprocessed to ensure consistency and compatibility with reanalysis data.
As discussed in Section 3.4, the sparse station-based observations were interpolated into continuous gridded fields using the Inverse Distance Weighting (IDW) method.

In practice, I tested several values of the power parameter p and the number of neighboring stations N, selecting the configuration that minimized interpolation error through cross-validation on held-out stations.
This provided spatially coherent inputs while preserving the local variability of precipitation events, see some results in the Figure 18.

4.1.2 GRU model design

What I use here is a GRU-based model that is followed by a fully connected output layer.
First, I put as an input a sequence of precipitation values over a fixed temporal window.
I then define the core of the GRU model and its parameters :

- input size : the number of input features (here only the precipitation, so input size=1),
- hidden size : the dimensionality of hidden states that control the model capacity,
- num layers : the number of GRU layers to increase or decrease the depth and at the same occasion to enable to capture complex dynamics,
- dropout : dropout between layers which randomly disable a percentage of neurons to avoid overfitting.

The input sequence length was set to a fixed temporal window of past days, and the model was trained on INMET data from 2010–2014, with the following dataset split :

- Training : 2010–2012 (3 years),
- Validation : 2013 (1 year),
- Test : 2014 (1 year).

4.1.3 Hyperparameter optimization

To assess the sensitivity of the GRU to architectural parameters, several configurations were modified :

- Hidden size : from 64 to 32,
- Dropout : from 0.2 to 0.3,
- Number of layers : from 1 to 2.

These modifications were made for balancing model capacity and generalization. The final choice of parameters was guided by validation loss evolution and prediction stability.

4.1.4 Results of the model

Figure 6 shows the temporal comparison between real and predicted precipitation on the full test set.
The GRU model tries to capture the main precipitation temporal dynamics, although extreme peaks remain underestimated, which we will see later, is a common limitation of RNN-based models in meteorology.

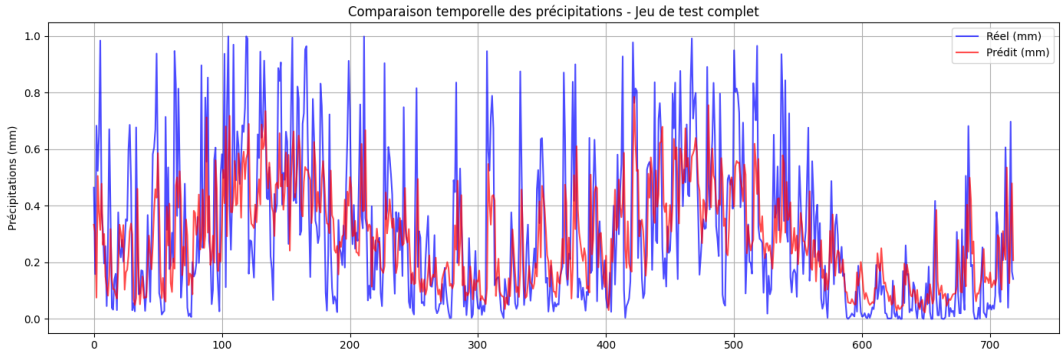


FIGURE 6 – Temporal comparison between observed and predicted precipitation (test set, INMET data).

The training and validation loss curves (Figure 7) show that the model converges in a good way with limited overfitting. The validation error stabilizes after approximately 10 epochs, which indicates a good balance between learning and generalization.

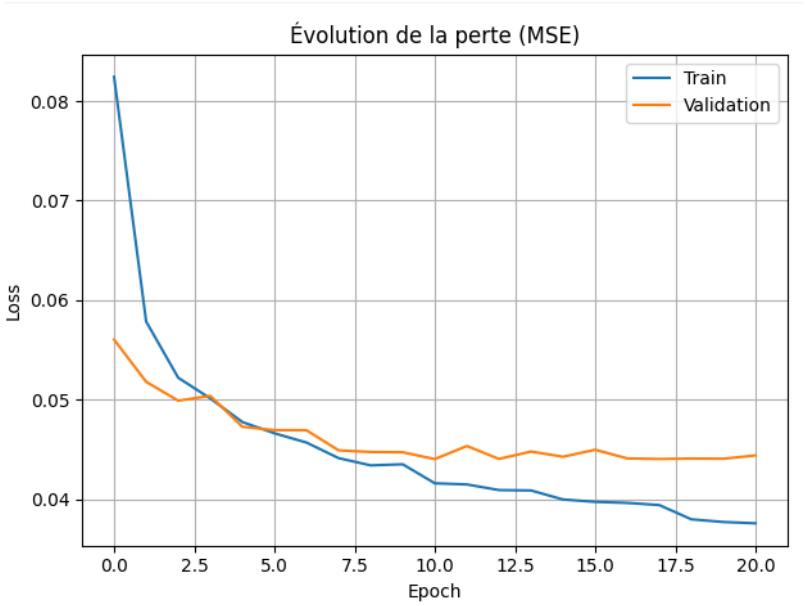


FIGURE 7 – Training and validation loss (MSE) during GRU model training.

4.2 Multi-Layer Perceptron (MLP) pixel model

The previous INMET-based model gave encouraging results but relied solely on precipitation as input and suffered from incomplete spatial coverage.
To improve predictive skill, I implemented a new model based on a Multi-Layer Perceptron (MLP) architecture, integrating additional meteorological variables and using a pixel-focused prediction approach.

4.2.1 Model overview

The MLP is a fully connected feedforward neural network designed to predict next-day precipitation at a fixed pixel location.
Its inputs consist of 34 meteorological variables from the previous day, including 4 surface variables and 6 atmospheric variables at 5 pressure levels (Section 2.2).
To account for seasonal variability, (see Figure 19) the month of the year was encoded as an 8-dimensional one-hot vector after removing the four driest months (July–October). All inputs and outputs were normalized using `StandardScaler` from `scikit-learn`, and outputs were later denormalized to mm for evaluation.

About the choice of the precipitation data, at first I continued using the data from the INMET stations and the spatial interpolation, but I quickly ran with major problems.
Indeed, the weather station are not always reliable. Be it some stations that shutdown and don't give results for days, weeks, or month, or precipitation values that sometimes can be false due to local perturbations, I found myself with incomplete datasets with many "holes" in data.
On the other hand, data from ERA5 was complete and clean, albeit more generalized.
Therefore, in the end, the choice was made to use fully the data from ERA5, including the precipitation, because even though the values of precipitation will be globally underestimated, at least the datasets will be clean and complete.

For the architecture, I progressively reduces the number of neurons per layer to compress and abstract features before prediction, with dropout applied after each activation to mitigate overfitting.
The model was trained with the Adam optimizer and Mean Squared Error (MSE) loss, with hyperparameters (learning rate, layers, activation functions, dropout rate) tuned to maximize generalization while avoiding parameter counts larger than the training set size.

4.2.2 Experimental setup

To limit spatial variability, the first experiments focused on two specific pixels :
— **High-precipitation pixel** – latitude 3°S, longitude 44.25°W, selected for its high monthly rainfall and data homogeneity.
— **Recife pixel** – latitude 8°S, longitude 34.5°W, chosen for its strategic importance to my supervisors.

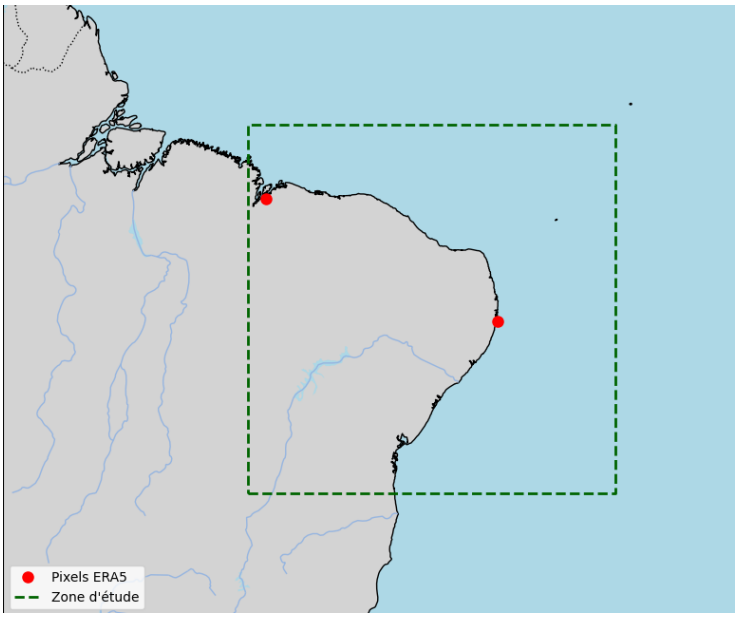


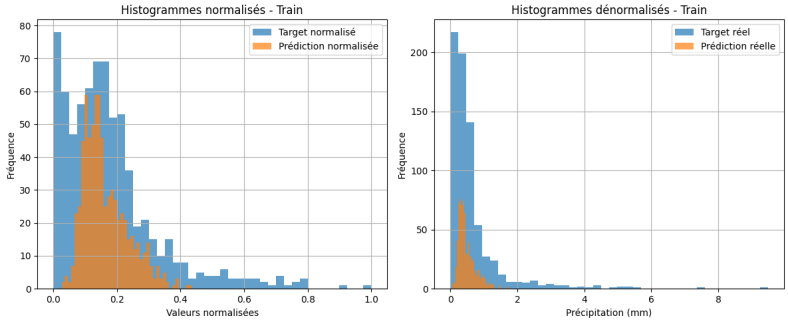
FIGURE 8 – Plotting of the localisation of the two studied pixel in the region of study.

Unless otherwise specified, results are reported for the high-precipitation pixel.

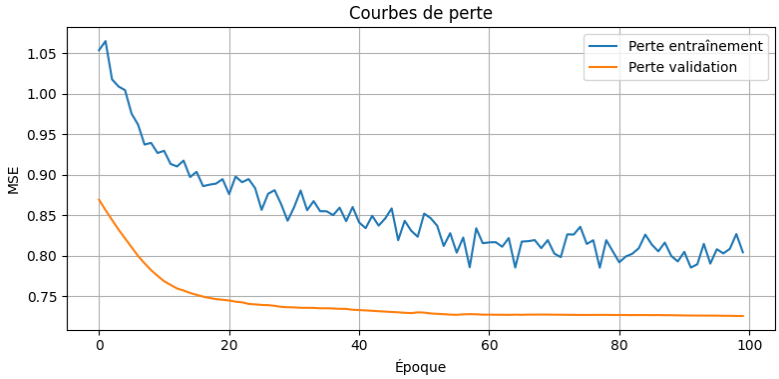
The input data was stored in monthly `NetCDF` files containing time, latitude, and longitude dimensions for each variable. A preprocessing script extracted the pixel-specific time series, applied one-hot encoding to the month, normalized the values, and constructed training, validation, and test sets.

4.2.3 Initial results with 5-year dataset

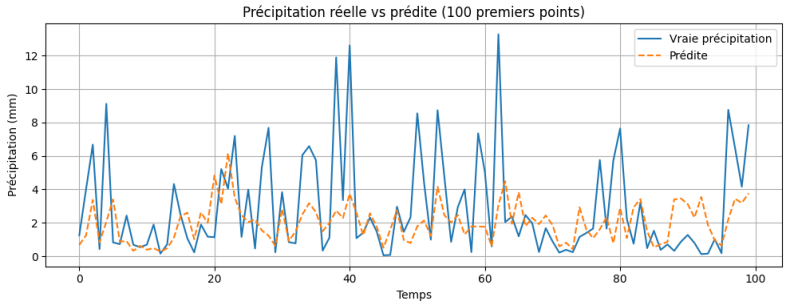
The first version of the MLP was trained on 5 years of ERA5 reanalysis data (2010–2014), split into :
— Training : 726 days,
— Validation : 242 days,
— Test : 242 days.



(a) Histogram compoarison between actual and predicted precipitation



(b) Loss function evolution



(c) Comparison between actual and predicted precipitation values over time

FIGURE 9 – First plottings of the MLP model

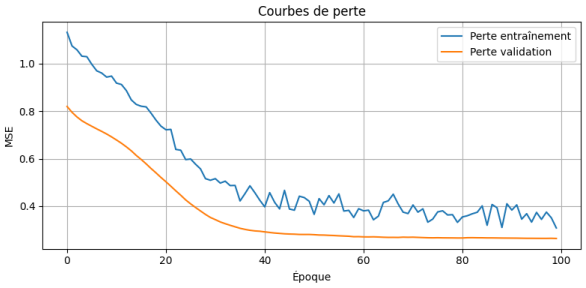
Figures 12a and 12b show the histogram comparison between observed and predicted precipitation, the loss function evolution, and time series comparisons.

A clear pattern emerged : while the model reproduced moderate rainfall reasonably well, it systematically underestimated extreme precipitation events. This behaviour—regression towards the mean—is typical of regression models when rare extremes are underrepresented in the training data. Predicted values were concentrated in a narrower range, leading to empty bins in the histogram compared to observed values spread over a wider distribution.

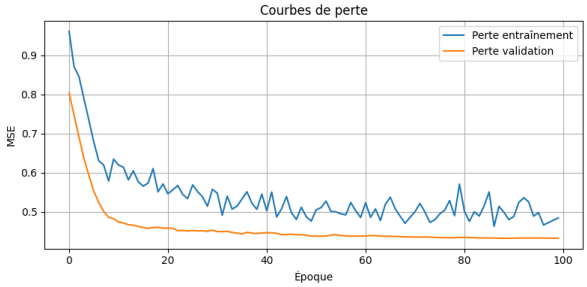
4.2.4 Impact of hyperparameter tuning and pixel variability

To assess robustness, I tuned learning rate, layer configuration, activation functions, and dropout rates, then tested the model on two contrasting pixels :

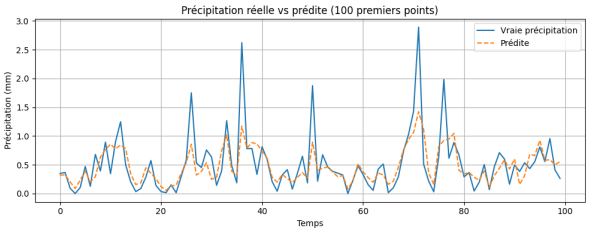
— A homogeneous pixel with low rainfall variability,
— A pixel with occasional intense rainfall events.



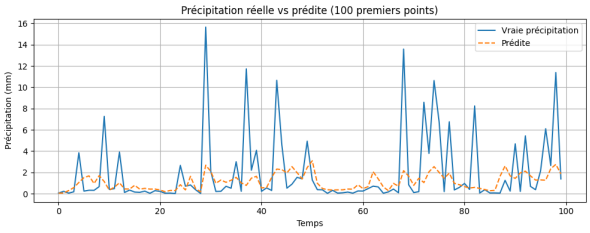
(a) Loss function evolution in a pixel without intense rain



(b) Loss function evolution in a pixel with intense rain



(c) Comparison between actual and predicted precipitation in a pixel without intense rain



(d) comparison between actual and predicted precipitation in a pixel with intense rain

FIGURE 10 – Plots of the first improved MLP model for two different pixels

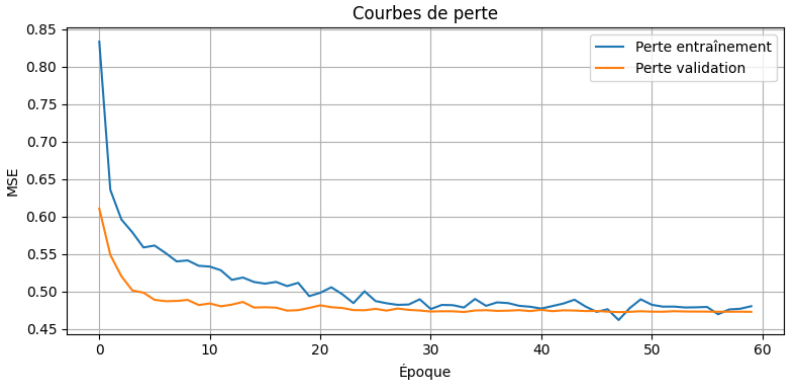
As shown in Figure 10, tuning improved convergence stability and reduced loss in the homogeneous case, but the intense rain pixel remained challenging, with larger errors and higher variance in predictions.

4.2.5 Extending the dataset to 32 years

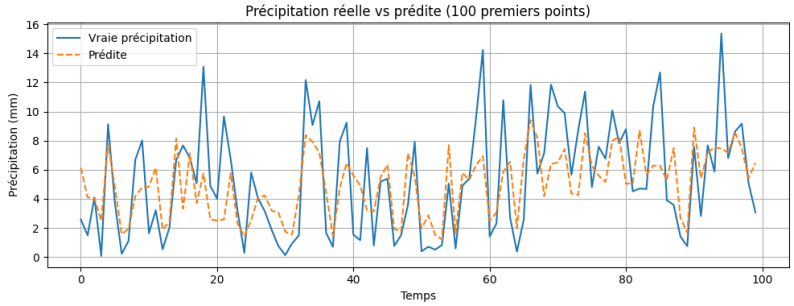
To address data scarcity, the training period was expanded from 5 to 32 years (1991–2022), keeping the same architecture and preprocessing pipeline.
The new dataset contained :

— Training : 7,025 days (1991–2019),
— Validation : 484 days (2020–2021),
— Test : 242 days (2022).

I then tested the model with the new dataset :



(a) New MLP dataset loss function evolution



(b) New MLP dataset comparison between actual and predicted precipitation

FIGURE 11 – Plots from the MLP using the new dataset

This sevenfold increase in training data significantly improved model generalization and reduced overfitting, as seen in Figures 12a and 12b.
Loss curves became smoother, and predicted time series aligned more closely with observations, particularly for moderate-to-high rainfall days.

4.2.6 Key observations and results

The new test metrics for the optimized MLP model are showed and compared with the baseline in the Table 1 :

TABLE 1 – **Test Metrics comparison, Baseline vs. Optimized MLP model**

Metric	Baseline normalized	Optimized normalized
MSE	0.68	0.54
MAE	0.61	0.54
RMSE	0.82	0.73
R^2	0.18	0.35

From these experiments, several conclusions can be drawn :

— Incorporating multiple meteorological variables (Section 2.2) improved predictive skill compared to precipitation-only baselines.
— Pixel-based modeling simplified the learning task and allowed better focus on local meteorological patterns, but limited spatial generalization.
— Rare, extreme rainfall events remain difficult to predict accurately without targeted sampling or specialized loss functions.
— Dataset length has a major impact on performance : extending the period from 5 to 32 years reduced overfitting and improved both stability and accuracy.

These points will be discussed in more detail in the next section.

4.3 classification of the data

Even if the results were encouraging with our first model for predicting precipitation, we still had some questions about the model.
One of the main questions was whether or not the model could effectively understand the difference between rain, and no rain.

To answer this question, the idea was to create a small new model of deep learning, different from the kind of model I used before that is used to predict how much it will rain at one time step, this one would predict whether it will rain or not.
This kind of model is called in deep learning a classification model (here, it is a supervised binary classification model), where :

— Inputs are the atmospheric variables
— The output is the precipitation and is compared directly to the actual precipitation
— The labels are binary :
  — 1 $\Rightarrow$ precipitation $>$ Rain treshold
  — 0 $\Rightarrow$ precipitation $<$ Rain treshold

4.3.1 Architecture of the classification model

For the base of the classification model, I used the MLP from before, as the MLP is also suitable for tabular time-series data.
So we have the same architecture as before with the MLP :

```
model = nn.Sequential(
    nn.Linear(input_dim,32),    # First layer, dense
    nn.Relu(),                  # Activation function
    nn.Dropoout(0.3),           # Dropout of 30%
    nn.Linear(32,1),            # Output layer
    nn.Sigmoid()                # new lined added for probability output
)
```

The sigmoid function added is an output activation layer, it's purpose is to convert the raw output of the FC layers into a probability :

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

Therefore this ending layer create a probability score $\hat{y} \in [0, 1]$, which enable binary decision making for the classification.

This sigmoid function is used in conjonction with a new loss function, the Binary Cross Entropy (BCE), which is a widely used loss function when the output is a probability.
Also called Log Loss, this function measures the difference between the predicted probability distribution and the

actual binary labels (0 or 1 here). It quantifies how well the predicted probabilities align with the true class labels. Mathematically the BCE for a single sample is defined as :

$$\mathcal{L}_{\text{BCE}}(y, \hat{y}) = -\left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right]$$

where :
- $\hat{y}$ is the predicted probability in $[0, 1]$
- y is the true binary level in $[0, 1]$

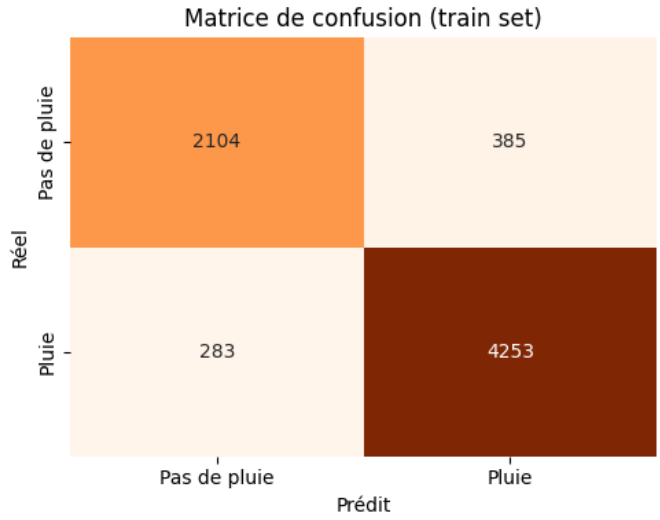
4.3.2 Results of the classification

I implemented the classification model and tested it for the training dataset and the test dataset, to see how well the model could classify the precipitation for the two dataset.
To analyse and understand the results, I made what we call a confusion matrix. It is a 2x2 matrix summarizing the prediction results :

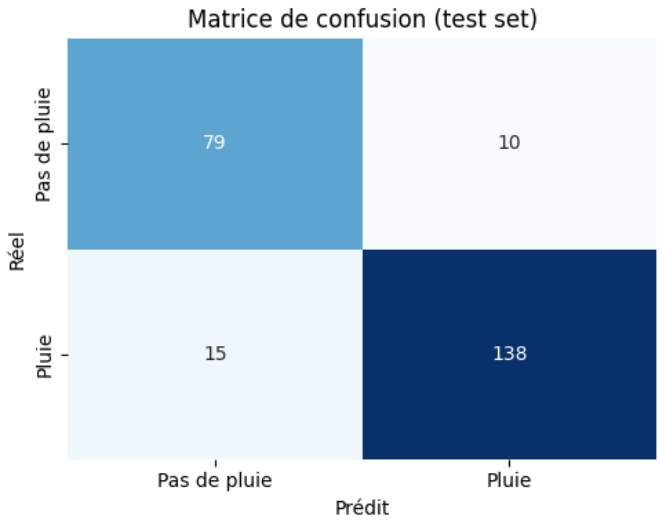
		Predicted	
		No Rain	Rain
Actual	No Rain	TN	FP
	Rain	FN	TP

Where :
- TP (True Positives) : correctly predicted rain events
- TN (True Negatives) : correctly predicted no rain
- FP (False Positives) : predicted rain but actually no rain
- FN (False Negatives) : predicted no rain but it actually rained

We obtained for the training and the test dataset :



(a) Training dataset



(b) Test dataset

FIGURE 12 – Confusion matrix plots for the training and the test dataset

To analyse this confusion matrix, we can take a look at the usual metrics used for that :

Accuracy : The proportion of correctly classified samples among all samples.

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

Accuracy is an indicator of how well the model performs overall.

Precision : The proportion of true positive predictions among all predicted positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision measures how many of the predicted "rain" events were actually rain.

Recall : The proportion of true positive predictions among all actual positive cases.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Recall measures how many actual rain events were successfully detected.

F1-score : The harmonic mean of precision and recall, balancing both metrics.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score is an indicator of the balance between precision and recall.

Therefore after computing, we get the following results for the metrics, one for the training set, and one for the test set :

TABLE 2 – **Classification metrics on train and test datasets**

Metric	Train set	Test set
Accuracy	0.9008	0.9033
Precision	0.9329	0.9168
Recall	0.9085	0.9352
F1-score	0.9205	0.9259

4.4 GRU Model Enhancements

4.4.1 Revised Architecture

Following the initial development of a MLP model that predicted precipitation at time t using data from the previous day $t-1$, the next logical step was to incorporate more temporal context. Since precipitation is inherently a temporal process where past values influence the present, it was hypothesized that providing the model with a sequence of multiple days rather than a single day would improve predictive performance.

To this end, the architecture was changed from a feedforward MLP to a recurrent neural network (RNN) based on the Gated Recurrent Unit (GRU).
As said in the presentation of the model **??**, The GRU is well-suited for capturing temporal dependencies in sequential data, and its relative simplicity compared to other RNN variants made it a suitable choice for this project.

The pixel of study didn't change, I still took the pixel at the coordinates latitude 3°S, longitude 44.25°W to be able to compare with the former model.
However, you can see the results of the GRU model for the Recife pixel which is less rainy, in the annexe,

Recife pixel – latitude 8°S, longitude 34.5°W The implementation largely reused the codebase from the MLP model, with key architectural modifications :

— The input data format was changed from a static feature vector of shape (N, F), where N is the batch size and F the number of features, to a sequence input of shape (N, T, F), where T represents the sequence length (initially $T = 1$ in the MLP).

— The model architecture shifted from a feedforward neural network to a recurrent neural network. The core GRU block was introduced to learn temporal patterns within the input sequences.
— To translate the high-dimensional hidden state output of the GRU into a scalar precipitation prediction, two fully connected (FC) layers were appended. These layers serve as a decoder to convert the GRU's final hidden state into a measurable output.
— Consequently, the model hyperparameters required re-optimization, as the previous values from the MLP were no longer suitable.

After experimentation, the GRU module was configured with 2 layers, each with 128 hidden units, and a dropout rate of 0.15 to reduce overfitting.

The final prediction $\hat{y}$ is obtained by passing the GRU's last hidden state $\boldsymbol{h}_T \in \mathbb{R}^{128}$ through two FC layers with intermediate nonlinearities and dropout, defined as :

$$\hat{y} = \boldsymbol{W}_2 \cdot \text{Dropout}_{0.15}\left(\text{LeakyReLU}\left(\boldsymbol{W}_1 \cdot \boldsymbol{h}_T + \boldsymbol{b}_1\right)\right) + \boldsymbol{b}_2$$

where :
— $\boldsymbol{h}_T$ is the final hidden state output by the GRU after processing a sequence of length T.
— $\boldsymbol{W}_1 \in \mathbb{R}^{64\times128}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{64}$ are the weight matrix and bias vector of the first FC layer, reducing the 128-dimensional GRU output to a 64-dimensional representation.
— LeakyReLU introduces nonlinearity, allowing a small gradient for negative inputs to mitigate neuron "dying" issues.
— Dropout with rate 0.15 is applied after the activation function to prevent overfitting.
— $\boldsymbol{W}_2 \in \mathbb{R}^{1\times64}$ and $\boldsymbol{b}_2 \in \mathbb{R}$ are the weights and bias of the final linear layer mapping the 64-dimensional vector to a scalar prediction $\hat{y}$.

4.4.2 Initial Results

The first results were obtained by training the GRU model with default parameters, without any hyperparameter optimization. To allow a direct comparison with the previous MLP model, the same evaluation metrics were used :

TABLE 3 – **Test Metrics for the Baseline GRU Model**

Metric	Normalized	Actual (mm)
MSE	0.3101	5.76
MAE	0.4107	1.64
RMSE	0.5645	2.40
R^2	0.6232	0.62

Among these, as I presented earlier, the MSE measures the average squared difference between predicted and true values, the RMSE expresses the error in the original units, and the R^2 indicates the proportion of variance explained by the model.

Compared to the MLP baseline (which had an MSE of 0.381, RMSE of 2.58 mm, and R^2 of 0.58 for the same pixel), the basic GRU model already demonstrates improved performance across all metrics.

The evolution of the loss during training is shown in Figure 13 :

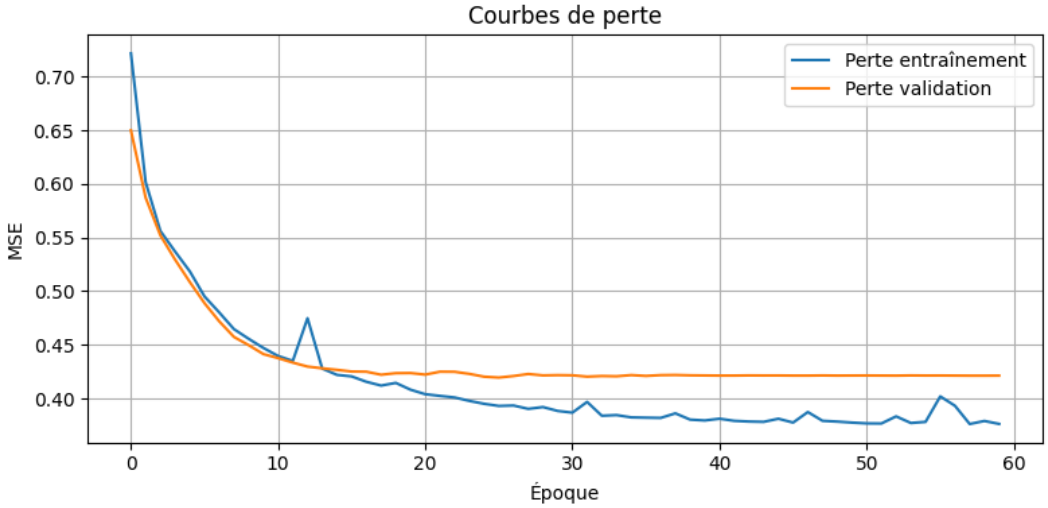


FIGURE 13 – Training and validation loss curves for the baseline GRU model.

Both training and validation losses decrease significantly during the initial epochs, indicating effective learning of patterns. Around epochs 15–20, the validation loss stabilizes near 0.42, while the training loss continues to decline to approximately 0.37, suggesting slight overfitting.
However, since validation loss does not increase substantially, overfitting appears limited and controlled.

The model's predicted precipitation compared to the true values over a rainy period is illustrated in Figure 14 :

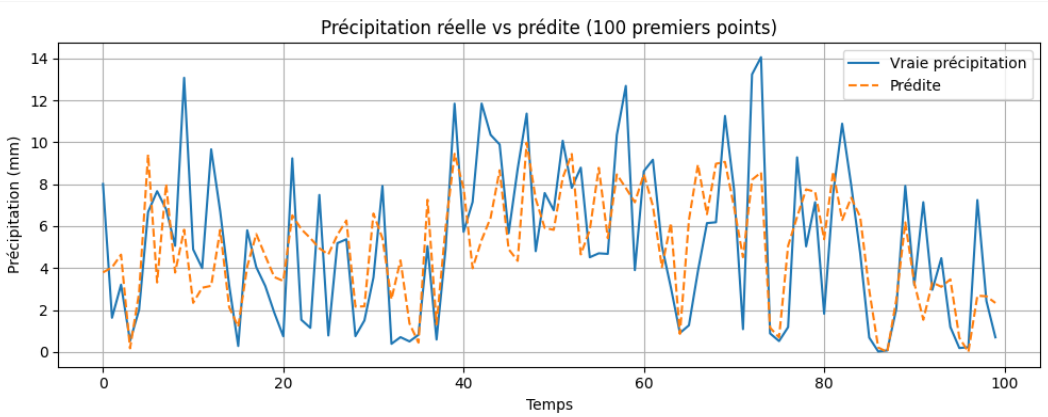


FIGURE 14 – Comparison of actual and predicted precipitation values over time for the baseline GRU model.

The model generally captures precipitation patterns, except for some notable discrepancies at $t = 25$, $t = 55$, and $t = 90$, where the predicted values diverge from observations. The model also struggles with accurately estimating precipitation magnitude, especially during the initial period ($t = 0$ to $t = 25$).

This limitation is further evidenced by the scatter plot in Figure 15, which shows a tendency for the model to both under- and over-estimate precipitation values :

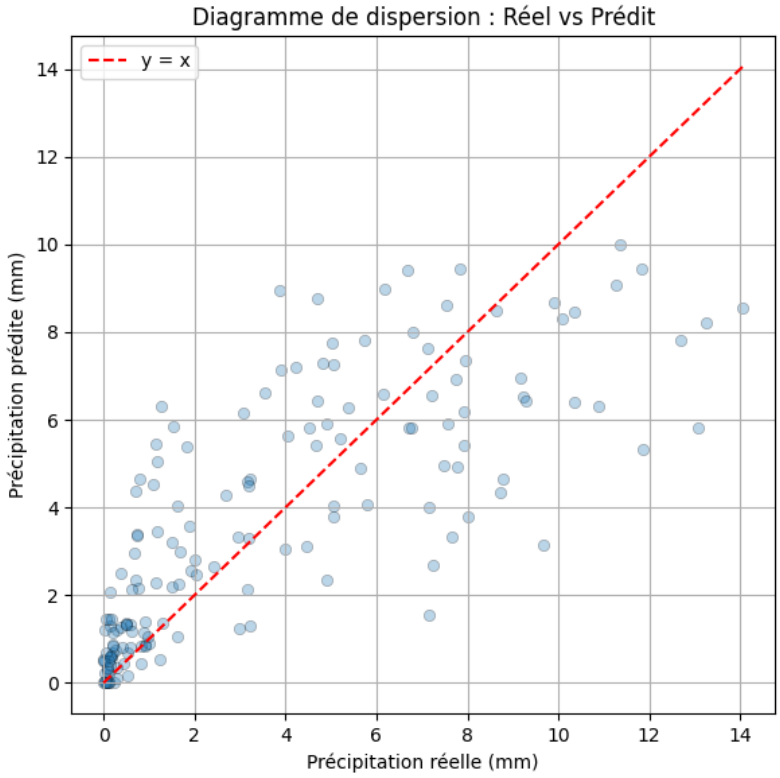


FIGURE 15 – Scatter plot of predicted versus actual precipitation for the baseline GRU model.

4.4.3 Hyperparameter Optimization

To improve the model, key hyperparameters were optimized. The initial hidden size of 128 was reduced to 32 to significantly decrease model complexity and training time, given the GRU parameter count formula for two layers :

$$\text{GRU parameters} = 3 \times [(D_{in} + H + 1) \times H + (L - 1) \times (H + H + 1) \times H]$$

where D_{in} is input dimension, H hidden size, and L number of layers. Reducing H from 128 to 32 decreases the parameter count by over a factor of 12.
Furthermore, the sequence length T was increased from 1 to 10, allowing the model to leverage more temporal context, essential for capturing precipitation dynamics.

Other hyperparameters, including learning rate, dropout rate, weight decay, and batch size, were also tuned. The final optimized parameters are summarized as :

- Hidden size : 32
- Number of layers : 2
- Dropout rate : 0.3
- Sequence length (number of time steps) : 10
- Batch size : 32
- Learning rate : 2.5×10^{-4}
- Weight decay : 1×10^{-7}
- Loss function : MSE

4.4.4 Results of the Optimized Model

The test metrics for the optimized GRU model are shown in Table 4 and compared with the baseline in Table 5 :

TABLE 4 – **Test Metrics for the Optimized GRU Model**

Metric	Normalized	Actual (mm)
MSE	0.2311	5.12
MAE	0.3934	1.58
RMSE	0.5057	2.26
R^2	0.7410	0.74

TABLE 5 – **Comparison of Test Metrics : Baseline vs. Optimized GRU Models**

Metric	Baseline		Optimized	
	Normalized	Actual (mm)	Normalized	Actual (mm)
MSE	0.3101	5.76	0.2311	5.12
MAE	0.4107	1.64	0.3934	1.58
RMSE	0.5645	2.40	0.5057	2.26
R^2	0.6232	0.62	0.7410	0.74

The optimized model shows clear improvements : MSE decreased by about 25%, and R^2 increased from 0.62 to 0.74, indicating stronger predictive power. The reduction in model size also accelerated training and reduced overfitting risk.

5 Discussion

5.1 A first step for the model

The classification model showed impressive performance for accurately making a first detection on whether or not there is enough precipitation on a pixel or not.
From the table of metrics (table 2), we can see that the classification performance of the model on both the training and testing datasets demonstrates a strong and efficient generalization capacity.
The accuracy remains consistent across the two sets (90.08% on training vs. 90.33% on testing), indicating that the model does not suffer from overfitting and maintains stable predictive capabilities on unseen data.

Furthermore, the high F1-scores in both sets, with a slight improvement on the test set (92.59% vs. 92.05%), suggest that the model achieves a good balance between correctly identifying rainfall events and minimizing false predictions.
Notably, the precision and recall values remain high and complementary between train and test sets. The slight drop in precision on the test set is compensated by a higher recall, meaning the model is slightly more likely to flag rainfall, but in doing so, it successfully captures more actual rainy events, which is a desirable trade-off in a real-world rainfall detection scenarios.

Overall, the consistency and strength of the metrics indicate that the model is robust, effectively managing the balance between false positives and false negatives while adapting well to new data.

This model could be used later on as a first big layer to accurately make a first prediction, to differentiate rain and no rain, before working more precisely and the regression part of the model.

5.2 The MLP model and its limitations

The pixel-based MLP approach demonstrated that daily precipitation prediction at a local scale is feasible and can yield consistent results, especially when supported by a sufficiently large and diverse training dataset.
Extending the input period from 5 to 32 years increased the number of training samples sevenfold, which substantially reduced overfitting, stabilized the loss function, and improved alignment between predicted and observed precipitation (Figures 12a and 12b).
These results confirm that deep learning models for precipitation forecasting require extensive temporal coverage to generalize well, particularly in tropical climates where rainfall regimes are highly variable.

However, the experiments also revealed clear limitations. Even with the larger dataset, the MLP systematically underestimated extreme rainfall events, a behaviour consistent with regression towards the mean when rare events are underrepresented.
Since intense precipitation days contribute little to the overall loss compared to the majority of moderate rainfall cases, the optimization process favours accuracy in the central distribution at the expense of extremes.
From an operational standpoint, this limits the model's utility for early-warning systems, where correctly predicting high-impact rainfall is very important.

The choice of fixed pixel prediction reduced spatial complexity and allowed the network to focus on learning local meteorological patterns.
Yet, this design also restricts spatial generalization : a model trained for one pixel cannot be expected to perform equally well on a distant location with a different rainfall regime without retraining or adaptation.
Testing on a pixel with infrequent but intense rainfall confirmed this limitation, showing higher loss values and more pronounced prediction errors.

From a methodological perspective, incorporating multiple meteorological variables (Section 2.2) clearly improved predictive skill over precipitation-only baselines.
Nevertheless, the simplicity of the fully connected MLP architecture limits its ability to capture spatio-temporal dependencies that are often critical in precipitation processes.

Overall, the MLP served as a strong first proof of concept, while also defining the methodological priorities for more robust and operational precipitation forecasting.

5.3 GRU improvements

The GRU-based recurrent neural network demonstrated that incorporating temporal sequences significantly improves daily precipitation prediction at the local scale.
By modeling the temporal dependencies within input sequences of meteorological variables, the GRU improved both prediction accuracy and the stability of training compared to the pixel-based MLP approach.

Using a sequence length of 10 days allowed the model to leverage preceding atmospheric conditions, which are known to influence rainfall patterns. This temporal context led to a notable increase in R^2 and reductions in MSE and RMSE metrics (see Table 5), confirming that recurrent architectures are better suited than feedforward networks for capturing dynamic weather processes.

These findings reinforce the importance of sequence modeling in deep learning for precipitation forecasting, particularly in climates with high rainfall variability where temporal dependencies can be strong and complex.

However, the experiments also revealed persistent challenges.
Despite improvements, the GRU still systematically underestimated high/extreme precipitation events, consistent with the general difficulty of learning rare, high-impact phenomena in imbalanced datasets. Since once again extreme rainfall occurrences contribute relatively little to the overall loss, the model tends to optimize for the more

frequent moderate or low rainfall cases. (see Figure 14).

I used another time a fixed sequence length and a pixel-based input structure that allowed the network to focus on temporal patterns at a local scale. Yet, this design still prevents spatial generalization, as the model's ability to predict precipitation in different regions or under distinct meteorological regimes remains untested and likely limited without retraining or adaptation.

Also, the optimization the model improved its ability to generalize over the training period, though some overfitting tendencies remained (Figure 13).
The recurrent design successfully captured short-term temporal dependencies but it might be useful to add other architectural enhancements to address abrupt precipitation changes and longer-range dependencies.

Overall, the GRU approach provided a substantial advancement from the baseline MLP, highlighting the necessity of sequence modeling in precipitation prediction.
It also defined key directions for future work, including handling class imbalance more effectively or integrating spatial context.

6 Conclusion

This project investigated the potential of different deep learning architectures for precipitation prediction in the northeastern region of Brazil, with a particular focus on extreme rainfall events.
Three complementary modeling strategies were explored :

- a pixel-based Multi-Layer Perceptron (MLP) for local-scale regression,
- a binary classification model to discriminate rain/no-rain events,
- a Gated Recurrent Unit (GRU) network to incorporate temporal dependencies.

The MLP experiments confirmed the feasibility of local-scale precipitation regression and highlighted the critical role of long-term datasets : extending the training period from 5 to 32 years substantially reduced overfitting and improved prediction stability. However, the MLP systematically underestimated extreme events, reflecting the class imbalance inherent to tropical rainfall records.

I also implemented a classification model that achieved high and stable performance across training and test datasets (a precision of more than 90%), demonstrating a good generalization and suitability as a first-stage rainfall detection tool.

The GRU-based approach provided a clear improvement over the MLP, with a 25 % reduction in MSE and an increase of the coefficient of determination R^2 from 0.62 to 0.74 on the test set.
By leveraging 10-day input sequences, the GRU better captured temporal patterns and short-term variability. Nevertheless, it retained the tendency to underestimate rare high-intensity rainfall, indicating that additional strategies—such as loss functions tailored to extremes, resampling methods, or hybrid spatio-temporal architectures—are required to improve high-impact event prediction.

Overall, the results demonstrate that deep learning models can effectively capture daily rainfall variability in northeastern Brazil when provided with sufficient input, temporal context and meteorological predictors.
Yet, operational forecasting of extremes remains challenging, particularly under spatially localized and temporally imbalanced conditions.

Future work should integrate spatial dependencies to improve the model prediction from one pixel to a whole zone, address class imbalance explicitly, and evaluate generalization across diverse rainfall regimes to progress toward a reliable, regionally adaptable precipitation forecasting framework.

7 Annexes

Exemple of a script SLURM

```
#!/bin/bash

# ##############################################################
# Script generique, pour demande de 2h avec carte A100

#SBATCH --job-name=XXXJOBNAMEXXX    # nom du job
#SBATCH --output=train_A100-%j.out # fichier de sortie (%j = job ID)
#SBATCH --error=train_A100-%j.err  # fichier  d erreur  (%j = job ID)
#SBATCH --constraint=a100          # demander des GPU A100 80 Go
#SBATCH --nodes=1                  # reserver 1 n ud
#SBATCH --ntasks=1                 # reserver 1 taches (ou processus)
#SBATCH --gres=gpu:2               # reserver 2 GPU par noeud
#SBATCH --cpus-per-task=10         # reserver 10 CPU par tache
#SBATCH --time=20:00:00            # temps maximal (HH:MM:SS)
#SBATCH --qos=qos_gpu_a100-t3      # QoS
#SBATCH --hint=nomultithread       # desactiver  l  hyperthreading
#SBATCH --account=knn@a100         # projet

#SBATCH --mail-user=XXXTON-MAILXXX
#SBATCH --mail-type=all

module purge
conda deactivate
module load arch/a100
module load pytorch-gpu/py3/2.5.0
set -x
export PYTHONUNBUFFERED=1

srun python -u script_python_a_executer.py

returned_code=$?
echo "> script completed with exit code ${returned_code}"
exit ${returned_code}
```

Listing 1 – SLURM script for a GPU A100 allocation

Quality of Service (QoS) for IDRIS resources

Partition	*QoS*	Limite en temps	Limite en ressources			
			par job	par utilisateur	par projet	par *QoS*
CPU	qos_cpu-t3 (*)	20 h	20480 cœurs	48000 cœurs	48000 cœurs	
	qos_cpu-t4	100 h	160 cœurs	1280 cœurs	1280 cœurs	5120 cœurs
	qos_cpu-dev	2 h	5120 cœurs	5120 cœurs	5120 cœurs	48000 cœurs
GPU	qos_gpu-t3 (*)	20 h	512 GPU	1024 GPU	1024 GPU	
	qos_gpu-t4	100 h	16 GPU	128 GPU	128 GPU	512 GPU
	qos_gpu-dev	2 h	32 GPU	32 GPU	32 GPU	512 GPU

FIGURE 16 – QoS resources needed for each partition

Activation functions

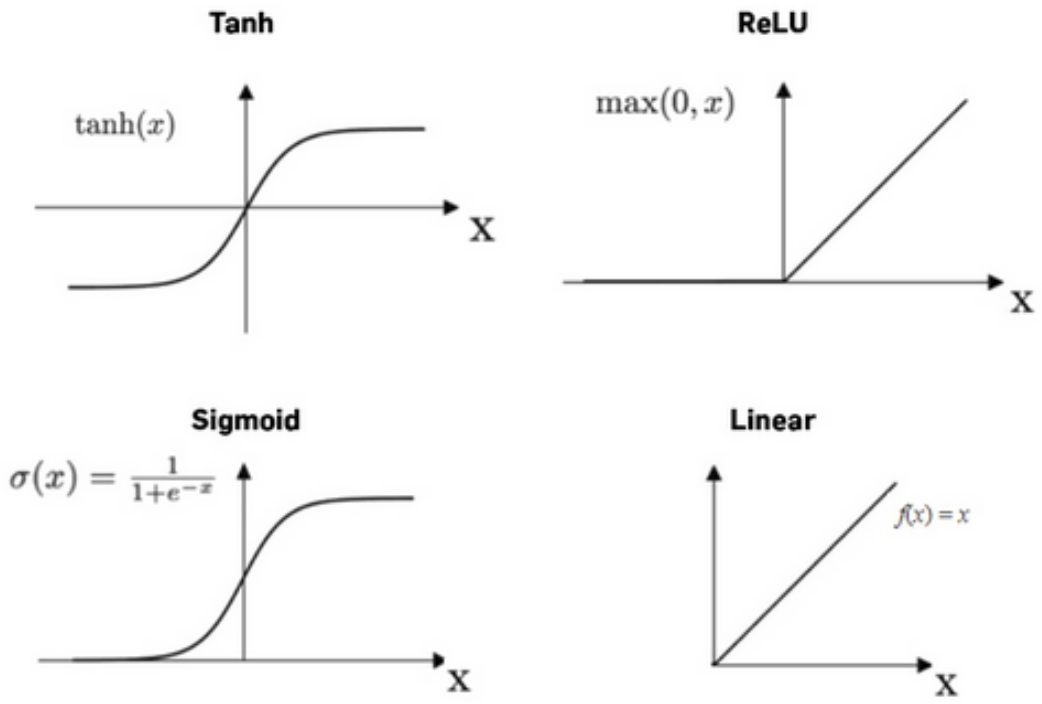


FIGURE 17 – Representation of the three main activation functions, tanh, ReLU nad sigmoid, compared to a linear function

IDW spatial interpolation results

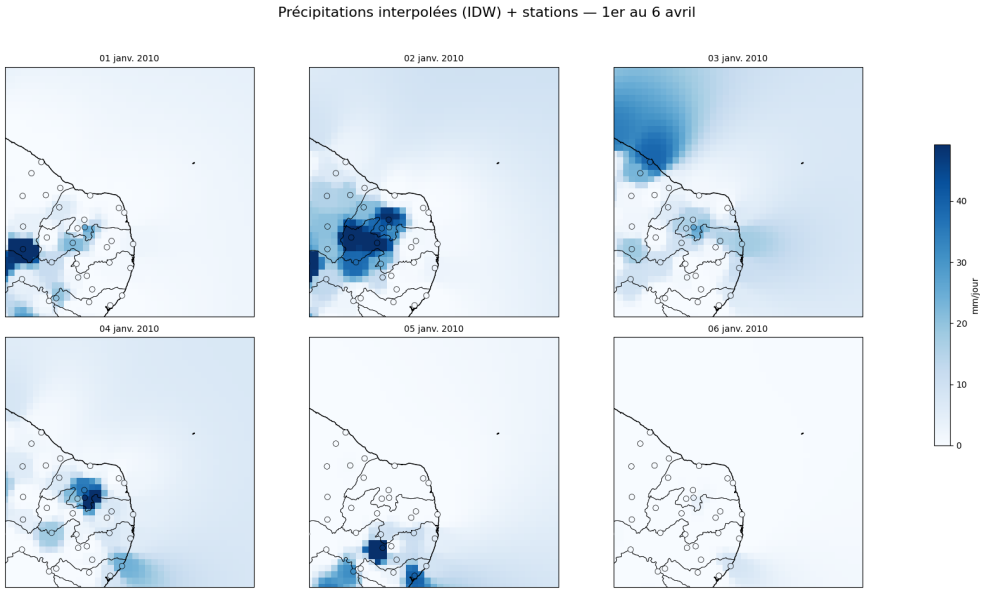


FIGURE 18 – Plot of IDW spatial interpolation on a reduced zone of the study, between the first and the sixth day of 2010

Precipitation cyclic pattern

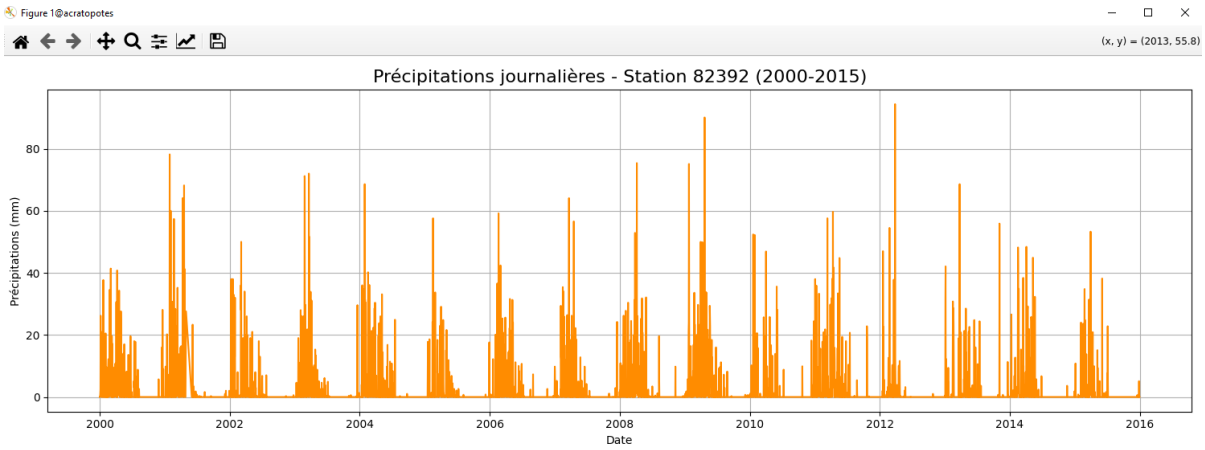


FIGURE 19 – Plot of the annual precipitation on a random station of Brazil, with the seasonal cyclic pattern visible

GRU optimized model results for Recife

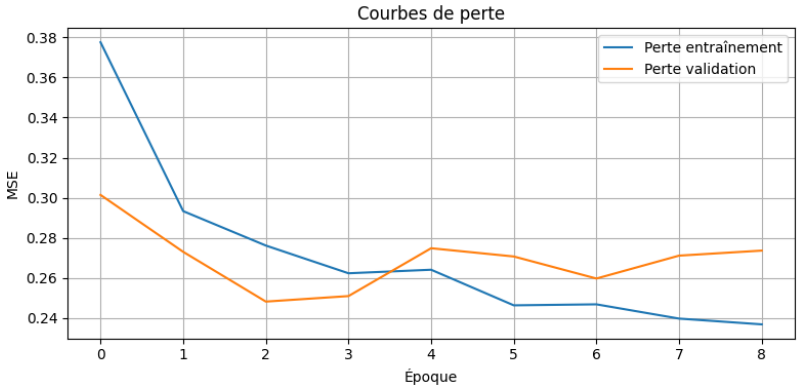


FIGURE 21 – Evolution of the loss function of the GRU optimized model on Recife, with early stop

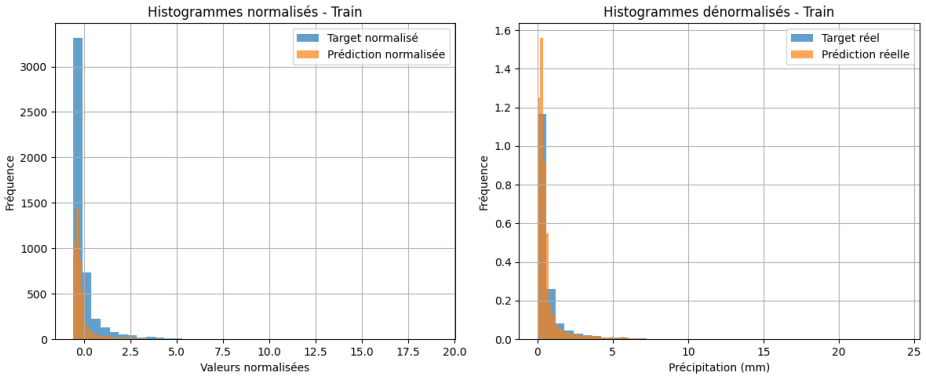


FIGURE 22 – Comparison histogram of the training set between real and predicted values

Métrique	Normalisée	Réelle (mm)
MSE	0.2942	0.93
MAE	0.3259	0.40
RMSE	0.7842	0.96
R^2	0.7515	0.75

FIGURE 20 – Metric results of the GRU model on Recife

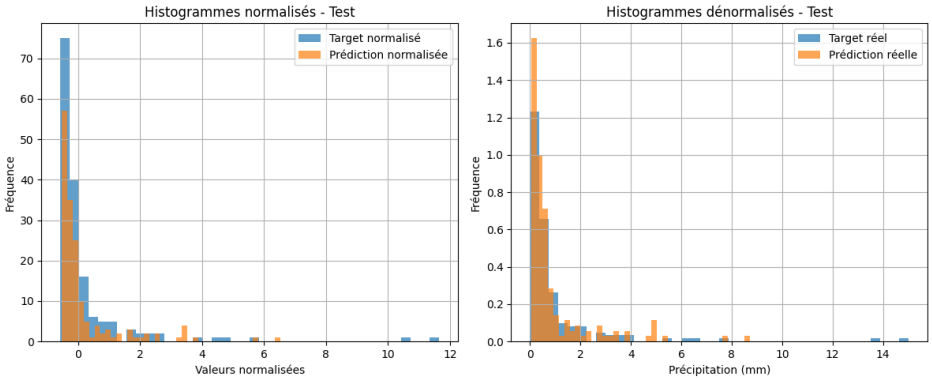


FIGURE 23 – Comparison histogram of the testing set between real and predicted values

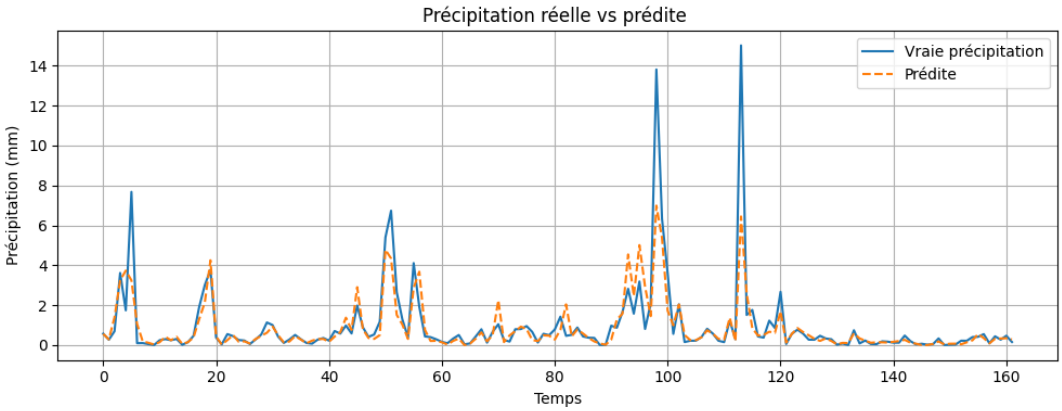


FIGURE 24 – Temporal comparison between real and predicted precipitation values

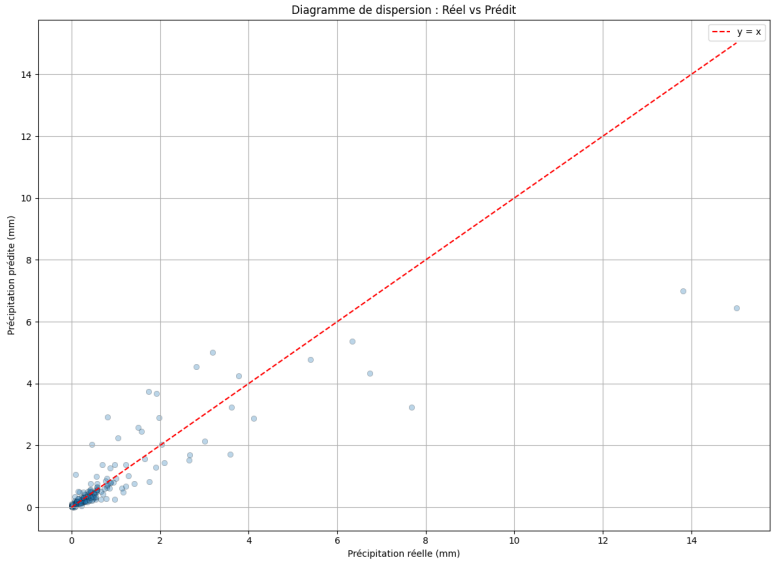


FIGURE 25 – Scatter plot of the bias between the real precipitation and its prediction

8 Tableau prévisionnel, et réalisé

TABLE 6 – Comparaison entre prévisionnel et réalisé pour le stage

Phase	Description	Prévisionnel	Réalisé
1	Découverte du stage, et prise en main de l'environnement de travail	Semaine 1–2	Semaine 1–3
2	Mise en place des objectifs du projet	Semaine 1–2	Semaine 1–2
3	Étude bibliographique et revue de l'état de l'art	Semaine 2–4	Semaine 2–5
4	Recherche approfondie sur des modèles similaires et leur possible utilisation dans le cadre du projet	Semaine 5–7	Semaine 3–6
5	Récupération et travail des données nécessaires au projet	Semaine 6–7	Semaine 5-9
6	Ecriture d'un premier modèle	Semaine 7-9	Semaine 7-11
7	Amélioration, optimisation, écriture nouveaux modèles	Semaine 10-18	Semaine 12-19
8	Rédation du rapport et préparation de la soutenance	Semaine 19-22	Semaine 20-22

Références

[1] —. "Recurrent Neural Networks: A Comprehensive Review of Architectures and Applications". In : *Information (MDPI)* (2024). Review of RNN and its sub models). URL : https://www.mdpi.com/2078-2489/15/9/517.

[2] D. Chicco. "The coefficient of determination R-squared is more informative and truthful than SMAPE". In : *Scientific Reports* (2021). "explanation of R^2 compared to other metrics). URL : https://www.nature.com/articles/s41598-021-80894-1.

[3] Shiv Ram Dubey, Satish Kumar Singh et Bidyut Baran Chaudhuri. "Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark". In : *Neurocomputing* (2021). comparison study of different activation functions. URL : https://doi.org/10.1016/j.neucom.2021.05.059.

[4] Florian Dupuy et al. "ARPEGE Cloud Cover Forecast Postprocessing with Convolutional Neural Network". In : *Weather and Forecasting* (2021). example of a CNN model in meteorological forecast. URL : https://journals.ametsoc.org/view/journals/wefo/36/1/WAF-D-20-0117.1.xml.

[5] K. Fukushima. *Visual feature extraction by a multilayered network of analog threshold elements*. first historical use of ReLU in a neural network. 1969. URL : https://ieeexplore.ieee.org/document/4307261.

[6] Benyamin Ghojogh et Ali Ghodsi. "Recurrent Neural Networks and Long Short-Term Memory Networks: Tutorial and Survey". In : *arXiv preprint arXiv:2304.11461* (2023). tutorial about different keywords of DL: RNN, LSTM, GRU, vanishing gradients, bidirectionnel, ELMo. URL : https://arxiv.org/abs/2304.11461.

[7] Institut du Développement et des Ressources en Informatique Scientifique (IDRIS). *Jean Zay : Supercalculateur convergé HPC / IA*. 2025. URL : https://www.idris.fr/jean-zay/.

[8] Remi Lam et al. "GraphCast: Learning skillful medium-range global weather forecasting". In : *Science* (2023). URL : https://deepmind.google/research/publications/22598/.

[9] Zachary C. Lipton, John Berkowitz et Charles Elkan. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In : *arXiv preprint arXiv:1506.00019* (2015). historical review and critics of RNN, LSTM, BRNN and its applications. URL : https://arxiv.org/abs/1506.00019.

[10] Farhad Mortezapour Shiri et al. "A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU". In : *arXiv preprint arXiv:2305.17473* (2023). comparison of different DL models (RNN, LSTM, GRU, Bi-LSTM, Bi-GRU, etc.) on a public dataset. URL : https://arxiv.org/abs/2305.17473.

[11] J. Pathak, S. Subramanian, P. Harrington et al. "FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators". In : *arXiv preprint arXiv:2202.11214* (2022). high resolution meteorological forecast model for precipitations. URL : https://arxiv.org/abs/2202.11214.

[12] I. H. Sarker. "Deep Learning: A Comprehensive Overview on Techniques and Applications". In : *Applied Sciences* (2021). show an overall view of the owrking of deep learning, including different kinds of network. URL : https://www.mdpi.com/2076-3417/11/3/1313.

[13] J. Schmidhuber. "Deep learning in neural networks: An overview". In : *Neural Networks* (2015). history of deep learning and its technics. URL : https://doi.org/10.1016/j.neunet.2014.09.003.

[14] Thiago Luiz do Vale Silva et al. "Ocean–Atmosphere Feedback during Extreme Rainfall Events in Eastern Northeast Brazil". In : *Journal of Applied Meteorology and Climatology* 57.5 (2018), p. 1211-1229. DOI : 10.1175/JAMC-D-17-0232.1. URL : https://doi.org/10.1175/JAMC-D-17-0232.1.