Development of a Long Short-Term Memory Recurrent Neural Network for Cyclone Trajectory Forecasting

Sam Barba

MSc Machine Learning and Autonomous Systems

The University of Bath

September 2022

Development of a Long Short-Term Memory Recurrent Neural Network for Cyclone Trajectory Forecasting

submitted by Sam Barba

for the degree of MSc in Machine Learning and Autonomous Systems

at the University of Bath

September 2022

Sam Barba

Sam Barba

Abstract

This report details the development of a Deep Learning model – specifically a Long Short-Term Memory (LSTM) network, a type of Recurrent Neural Network (RNN) – for the purpose of cyclone trajectory forecasting. The final model uses a historical window of wind speed, pressure, latitude, and longitude data, forty-eight hours long, to forecast the next twelve hours of movement with considerable accuracy.

The report includes: an introduction on the motivation and challenge behind the problem this project aims to solve; a literature review describing cyclonic phenomena and contemporary machine learning techniques applied to cyclone forecasting; a description of the dataset used; a model implementation section; model evaluation; and finally, discursive conclusions about the model and project as a whole.

Acknowledgements

First and foremost, I wish to express gratitude to my supervisor, Dr Marina De Vos, for her invaluable guidance. Her critiquing and advice during every stage were instrumental.

Special thanks go to my family and friends, for their perpetual support and belief in me which fuelled this project and my general academic career.

Contents

List of Figures

List of Tables

List of Equations

1 Introduction

1.1. Motivation and Challenge

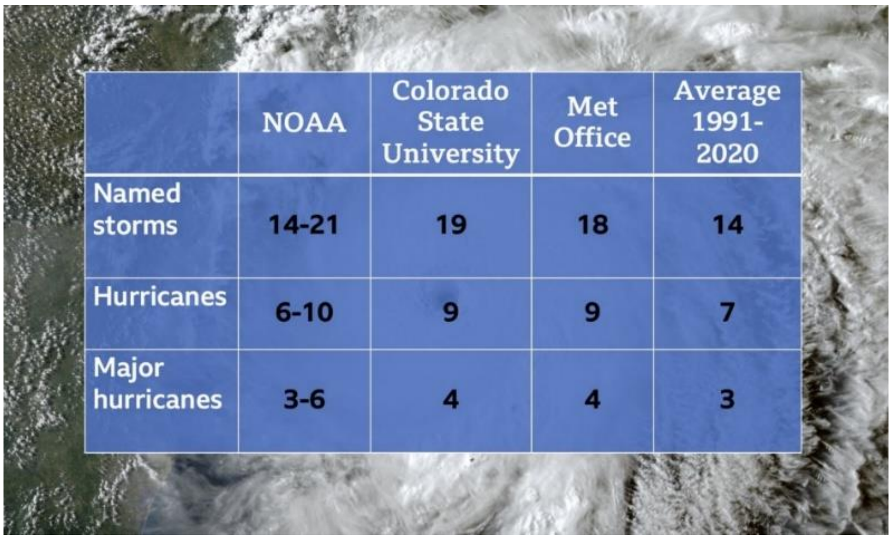
Long-term forecasting of extreme weather events is a critical task in meteorological operations. It can be vital in preventing loss of life and tremendous damage costs. Precise, accurate forecasts should come from a detailed comprehension of cyclonic phenomena: cyclones have a "complex three-dimensional structure, and the surrounding atmosphere is the driving force of their development" [1].

Before continuing, this appendix clarifies any misconceptions about cyclones, hurricanes, and typhoons (tornadoes are completely unrelated, as they form on land) [2].

Thus, 'cyclone' will be used to describe these phenomena henceforth. Forecasting cyclone tracks is a *spatiotemporal* sequence problem [1]: the model must consider the extraction and application of temporal and spatial features simultaneously. An optimal choice of architecture for this task is Long Short-Term Memory, as will be explained.

Increased Cyclone Activity

Forecasters predict a highly active cyclone season for 2022: the National Oceanic and Atmospheric Administration (NOAA) has released the official outlook for the cyclone season, which suggests above-average numbers of storm systems. It follows 2021's third most active season on record. The outlook, shown below [3], makes it the seventh consecutive cyclone season with above-average activity.



	NOAA	Colorado State University	Met Office	Average 1991-2020
Named storms	14-21	19	18	14
Hurricanes	6-10	9	9	7
Major hurricanes	3-6	4	4	3

Figure 1: Outlook for 2022 cyclone season

The most influential factor on such forecasts is the El Niño Southern Oscillation (ENSO), a weather pattern in the eastern Pacific Ocean which has implications on weather across the globe. It refers to changing trade winds and sea surface temperature in this region with a warmer than average El Niño, cooler than average La Niña, and a neutral phase. Trade winds are air currents close to Earth's surface that blow from east to west near the equator [4]; during El Niño and La Niña events, trade winds are weaker or stronger than usual, respectively [5].

Warmer than average sea temperatures (in the Atlantic Ocean and Caribbean Sea), together with weaker tropical Atlantic trade winds, may be other contributing factors to increased cyclone activity according to the NOAA. See this heatmap [3] depicting the former factor.

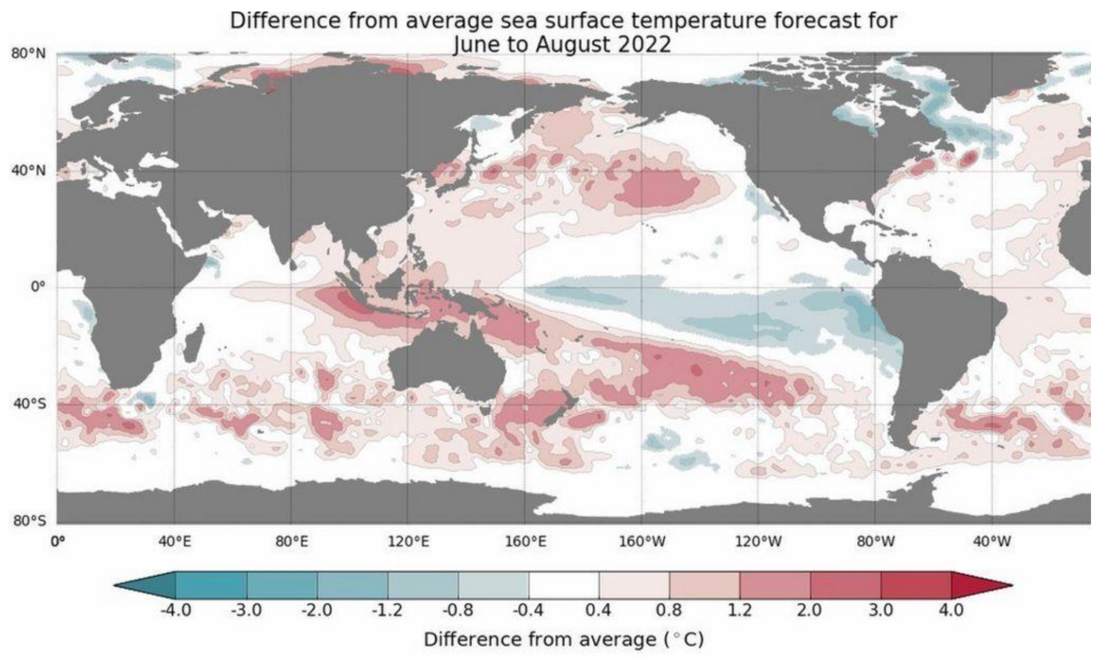

Difference from average sea surface temperature forecast for June to August 2022
80°N
40°N
0°
40°S
80°S
0°
40°E
80°E
120°E
160°E
160°W
120°W
80°W
40°W
-4.0
-3.0
-2.0
-1.2
-0.8
-0.4
0.4
0.8
1.2
2.0
3.0
4.0
Difference from average (°C)

Figure 2: Relative sea surface temperature forecast for June to August 2022

Climate change is also a potential factor under scrutiny, as another active cyclone season is expected in 2022. However, as the principal contributing element for tropical cyclone development is the ENSO pattern, scientists suggest climate change "is not pivotal for an increase in storms" [3].

On the other hand, the other factor of cyclone development is warmer than average sea temperatures. So, this could be effectual: warmer ocean waters signify a greater abundance of fuel for cyclogenesis and intensification. According to the Intergovernmental Panel on Climate Change (IPCC), there is "high confidence that the proportion of tropical cyclones will increase on a global scale with increasing warming" [6].

1.2. High-level Project Requirements

The main purpose of this work is to design and implement a long short-term memory network for the purpose of cyclone trajectory forecasting, and evaluate its predictive accuracy. Development includes experimenting with different topologies, for instance the number of layers in the network and what kind of activation functions to use. Each model iteration is trained and evaluated on the same dataset to avoid bias, with results analysed to inform how the hyperparameters of consequent model iterations should be tuned.

1.2.1. Why Neural Networks?

If algorithms such as logistic regression and support vector machines (SVMs) exist, why are neural networks (NNs) growing in popularity? Why use one here? The below figures [7] illustrate a principal reason: when both algorithms are trained on large amounts of data, performance of logistic regression models plateaus at a certain point. But in the case of NNs, performance continues growing at a logarithmic rate, depending on the size of the net.

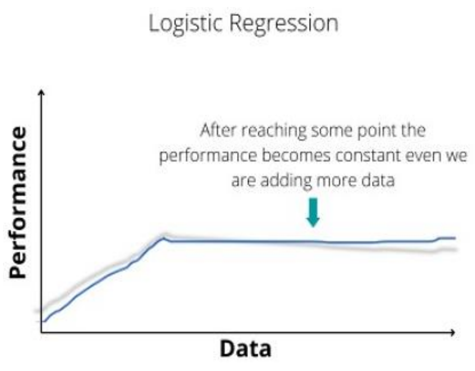


Figure 3: Logistic regression performance vs data

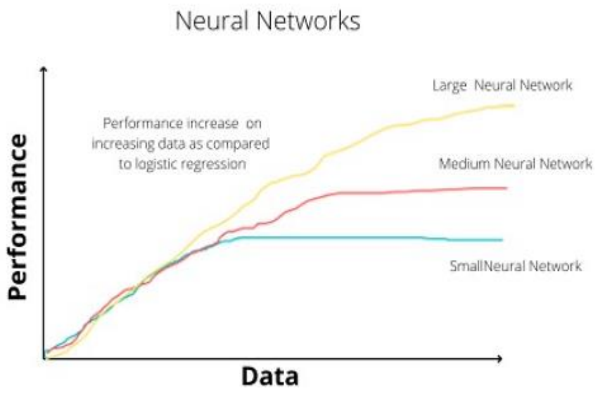


Figure 4: Neural Network performance vs data

Since nowadays there are vast volumes of data available, it is essential to extract the best knowledge from them, which is why NNs are vital paradigms – especially when it comes to predicting cyclone trajectories, as datasets can be very large (e.g. generated by many atmospheric science organisations) and complex.

Furthermore, NNs are adept function approximators: a neuron itself, in this context, is simply a function – one that can take many inputs, and produces one output. Each input is multiplied by a weight, and all are added together along with a bias. The weights and bias comprise the parameters of the network.

An NN can be described as a function f that depends on parameters α to make predictions:

$$\hat{y}_i = f(x_i|\alpha)$$

Equation 1: Neural network described as a function

The goal is to make $\hat{y}_i$ (predictions) and y_i (true values) as close as possible by modifying the values of α (via *learning*). In general, a loss function $L(y, \hat{y})$ is defined: the more similar the predictions to the true values, the smaller the value of L. For instance, a common loss function is the mean squared error (MSE):

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

Equation 2: Example loss function (mean squared error)

Another important concept to note is that of activation functions. These are mathematical 'gates' between the input feeding a neuron and its output. They essentially decide whether the neuron should activate or not. They crucially introduce non-linearities into a network: if they were not used, the weights and bias would simply result in a linear transformation. A linear equation is simple to solve, but is limited in its capacity to solve real-world problems, as it has less ability to learn complex functional mappings from data. Neural networks lacking activation functions would simply be linear regression models.

Finally, virtually no matter what dataset is handed to a neural network, it can be learned. This is because NNs can be rigorously proven to be universal function approximators [8] – they can approximate any function, to virtually any degree of precision needed (so long as enough neurons are added, and there is sufficient data).

1.2.2. Why Long Short-Term Memory?

This choice of architecture can be corroborated by exploring the details of time series forecasting. This is an important area of machine learning (ML) that is often neglected; it is important because there are countless prediction problems that involve temporal components. Such problems are neglected because it is these components that render time series problems difficult to manage [9]. An example problem could be forecasting whether an EEG trace indicates a patient is having a seizure or not.

Time series datasets differ from normal ML datasets, as they add an explicit order dependence between observations: a time dimension. This extra dimension is simultaneously a constraint and a source of further information.

To forecast a time series (predict future observations about one or multiple features), a model must be trained on historical data. A key distinction to note is that the future is completely unattainable and must only be approximated using what has already occurred.

For example, given the sequence (1, 2, 3, 4, 5), we must predict (6):

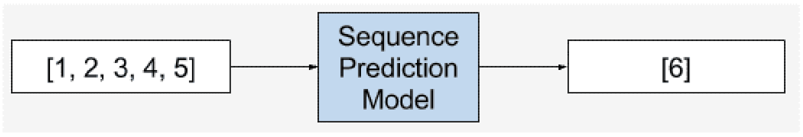


Figure 5: Simple example of a sequence prediction problem

The skill of a time series forecasting model is determined by its predictive performance. This is often at the expense of [9]: being able to explain why a specific prediction was made; confidence intervals; and better understanding of the underlying causes behind the problem.

Secondly, recurrent neural networks (RNNs) must be defined. The structure of standard feedforward NNs is as follows:

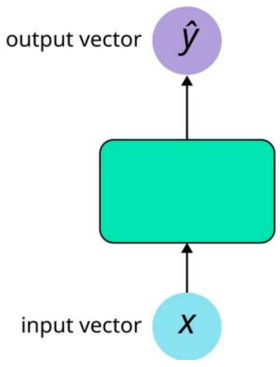


Figure 6: Structure of standard feedforward NNs

RNNs, conversely, have an internal loop:

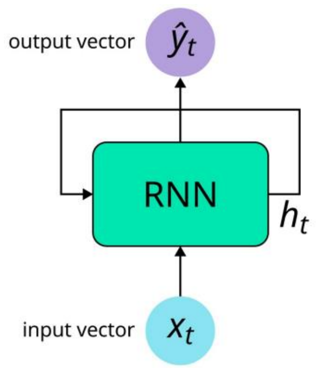


Figure 7: General RNN structure

The key idea is that RNNs have a hidden state h_t, considered their 'memory', which can represent context information. At each time step, a recurrence formula is applied to update h_t:

$$h_t = f_W(h_{t-1}, x_t)$$

Equation 3: Recurrence formula to update RNN hidden state

where:

- f_W = function parameterised by W (weight matrix)
- h_{t-1} = old state ($h_{t-1} = f_W(h_{t-2}, x_{t-1}), \dots, h_1 = f_W(h_0, x_1)$)
- x_t = input vector.

As explained in [10], "signals passing through recurrent connections constitute an effective memory for the network, which it can then use to better predict future time series values". This is a principal reason for the selection of RNNs for the task this project aims to solve. However, a 'vanilla' RNN is insufficient: they have difficulties modelling long-term dependencies because of the *vanishing gradient problem*. This is a problem experienced with NNs that are trained with gradient-based learning methods: in such techniques, during every training iteration, each network weight is updated proportionally to the loss function's partial derivative (with respect to the weight). In some cases, the gradient may be extremely small and *vanish*, effectively impeding the weight value from changing. In the worst case, this could altogether terminate the learning process [11].

A solution for this is to use a more complex recurrent unit with 'gates' to control information flow: a long short-term memory network. These gates include: a forget gate (discards irrelevant or old information); an input gate (stores relevant information from current input); and an output gate (a filtered version of the LSTM's cell state, which itself stores long-term relevant information).

Bidirectional LSTMs process sequences in both directions, detecting trends that may be missed by the chronological-order version alone. Hence, the final model version of this project uses bi-LSTM layers.

1.2.3. Choice of Programming Language

Python is the language of choice for this project, as it is arguably the best language for ML tasks, largely due to its syntactic simplicity and substantial collection of third-party libraries, many of which are suited to developing ML models.

One of these libraries is TensorFlow, one of the most popular for ML development [12]. TensorFlow offers many advantages over alternatives like PyTorch, including the fact that model training can be done in one line by calling *model.fit* with necessary parameters [13]. Hence, TensorFlow is used for this project, via the Keras API.

1.3. Project Plan and Report Structure

The first step of any ML development pipeline is data collection [14]. Structured, time-series cyclone data is required, with enough features to aid meaningful exploratory data analysis (EDA). The dataset used for this project is the NHC's *Hurricanes and Typhoons, 1851-2014* dataset, which will be described in section 3.

The second task is data cleaning and EDA. This allows comprehension of the structure of a dataset. The primary goal of EDA is to make data 'clean': a dataset may have outliers, missing values, or duplicate values. EDA can be used to resolve these undesirable qualities. It also aids in comprehending the relationship between the variables, providing a broader view of the data.

The data cleaning and EDA steps are described in 3.1 and 3.2, respectively, followed by detailing of feature selection (3.3), and finally, data pre-processing steps required for feature selection and model training/testing in 3.4.

Feature selection is a vital task, as it helps reduce the 'curse of dimensionality': this is the observation that, with more feature dimensions, exponentially more data is needed to capture the essence of these features. Feature selection involves choosing a subset of features to use for model training, whilst ignoring the others. Obviously, the subset which yields optimal performance should be chosen. Some typical approaches for this include:

- Train models with many subsets of features, and select the best
- Use an ML algorithm with feature selection built in (embedded feature selection).

However, feature selection is done manually here, as the number of features is manageable.

The penultimate task is model development, which is a cyclical process of experimenting with different architectures and hyperparameters, and evaluating performance in order to aid selection of the next architectural paradigm and hyperparameter values, and so on. Hence, this project uses an agile developmental approach, primarily because a tenet of this technique is the fact that documentation, design, implementation, and testing are cyclical and interleaved.

The model development process will be discussed in section 4, together with an outline of the results of each iteration. This will be followed by the final task of in-depth model evaluation and results analysis, in section 5.

This report finishes with discursive conclusions (section 6), including limitations of the final model, possible future work, and reflections.

2 Literature Review

2.1. Genesis, Structure, and Movement of Cyclones

Cyclones can span up to hundreds of miles across, and can devastate coastal areas with massive rains, surging waves, and sustained winds of more than 160mph [15]. These phenomena form over warm oceans in tropical latitudes (about 23.5°N – 23.5°S, between the Tropic of Cancer and the Tropic of Capricorn [16]). Here, they gain vast amounts of heat energy and moisture. Cyclone intensity depends on how much energy can be pulled from warm ocean water [15].

2.1.1. Genesis

In order for a cyclone to form, several atmospheric conditions must be met [17], [18]:

- **Temperature and humidity:** ocean waters should be at least 26.5°C at the surface, and warm for a 45m depth. They also need an atmosphere which cools fast enough with increasing height so that the difference between the top and bottom can create storm conditions. A moist mid-troposphere (3 miles high) is also needed, as dry air ingested into thunderstorms at mid-level can suppress circulation.
- **Spin and location:** the Coriolis force is an apparent force that deflects movement to the right coming from the Northern hemisphere, and to the left coming from the south. This force is greatest at the poles and zero at the equator, so the storm must be at least 300 miles from the equator in order to have spin. This causes cyclones in the Northern hemisphere to rotate anticlockwise, and vice-versa in the south. This spin may play some role in helping tropical cyclones to organise. (Side note: the Coriolis force is not strong enough to affect small containers such as toilets. The notion that water spins the other way when flushed in the opposite hemisphere is a myth.)
- **Wind:** low vertical wind shear (the change of wind velocity with height) between the surface and the upper troposphere favours thunderstorm formation, which provides the energy for cyclones. Too much wind shear will disrupt or weaken convection.

Other than these cyclone-favourable conditions, many regular atmospheric phenomena contribute to cyclogenesis. For example, the Pacific Decadal Oscillation (PDO) and Atlantic Multi-decadal Oscillation (AMO) are oceanic temperature fluctuations occurring over decades, which can influence overall cyclone activity over the world's tropical oceans [19]. For instance, when the tropical North Atlantic Ocean is warmer than usual, cyclones tend to form and intensify more often.

2.1.2. Structure

Cyclones are comprised of an eye, eyewall, and multiple rainbands. Air spirals in towards the centre, then exits at high altitude in the opposite direction. At the very centre, air sinks, forming the eye as depicted [20]. This area is relatively free of clouds.

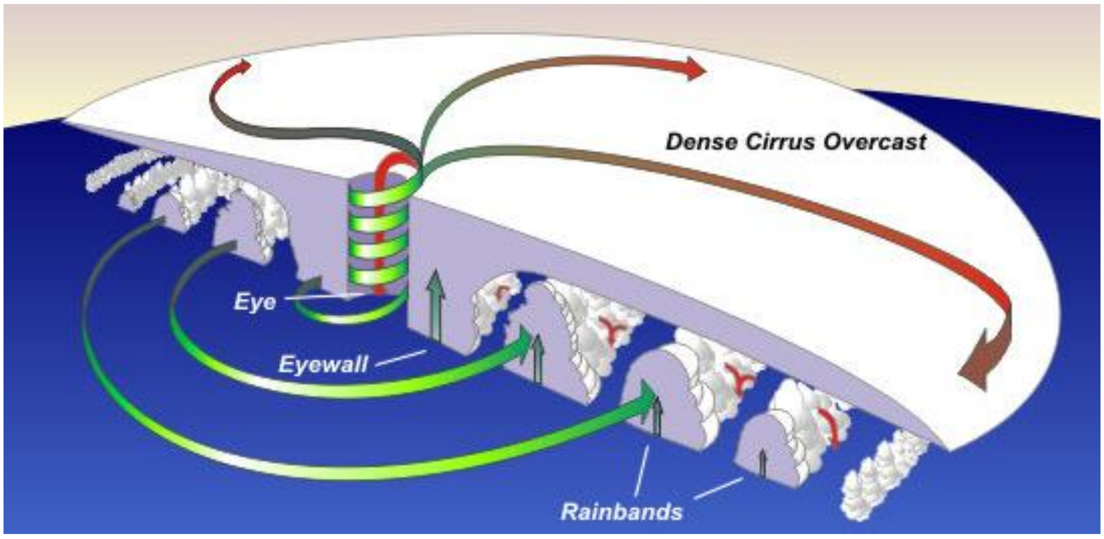


Figure 8: Cross section of a typical cyclone

A cyclone's centre is around 20-40 miles across, and is relatively calm. The speed of the light winds here usually stays below 15mph. An eye usually develops when the maximum sustained wind speeds go above 74mph. Although the nature of eye formation is still not fully understood, scientists argue that a combination of conservation of angular momentum and centrifugal force is probably a major factor [20]. Conserving angular momentum means that objects rotate faster as the radius of their spin decreases. So, the speed of air increases as it heads towards the centre. However, an apparent outward-directed (centrifugal) force occurs as this speed increases, because the wind's momentum wants to carry the wind in a straight path. Since it is turning about the centre of the cyclone, an outward pull is experienced.

Eyewall formation occurs when rotation of air around the cyclone exceeds about 74mph, at which point the rotating air balances inflow to the centre, causing air here to ascend 10-20 miles, creating the eyewall. Eyewalls are essentially concentric rings of thunderstorms. An indicator of storm intensity comes from changes of wind speed produced by transformations in the eye structure or eyewall structure. In intense cyclones, some outer rainbands may organise into a ring of thunderstorms that slowly moves inward, robbing the inner eyewall of moisture and momentum. During this phase, the cyclone is weakening. Eventually, the outer eyewall replaces the inner one completely [20].

Finally, rainbands are curved bands of thunderstorms that trail away from the eye wall spirally. There are occasional gaps between rain bands where no rain or wind is found [20].

2.1.3. Movement and Lifecycle

The surrounding flow throughout the depth of the troposphere (from the surface to approximately 12km) can be thought of the principal steering force for cyclones. Former director of the US National Hurricane Centre (NHC), Dr Neil Frank, used the analogy that cyclone movement "is like a leaf being steered by currents in a stream, except that with cyclones the stream has no boundaries" [19].

The predominant circulation pattern in the tropics is the subtropical ridge, a semi-permanent high-pressure cell approximately between the tropics of Cancer and Capricorn. In the Atlantic, this ridge is often called the Bermuda High due to its latitude. South of this ridge, the circulation drives cyclones westward with a slight poleward component. But when a cyclone reaches the west side of the ridge, it will tend to move around the Bermuda High, first poleward then in an easterly direction. This is known as recurvature [21], and is evident in the dataset used for this project (3.4.1). This motion signifies that many tropical Atlantic cyclones may curve back out into open ocean without once making landfall.

However, occasionally a cyclone will make landfall. Its circulation would decay, should it cross an island, especially a mountainous one. If it reaches a continent, it will be deprived of its fuel source – warm, moist sea air. The cyclone will also begin to draw up dry continental air; "when combined with increased friction over land, this leads to the weakening and eventual death of the cyclone" [22].

2.2. Machine Learning and Deep Learning

Before discussing any deep learning (DL) application to cyclone path prediction, the main difference between DL and regular ML must be noted: deep learning is an end-to-end process, as it performs feature extraction automatically. The below figure [23] illustrates this:

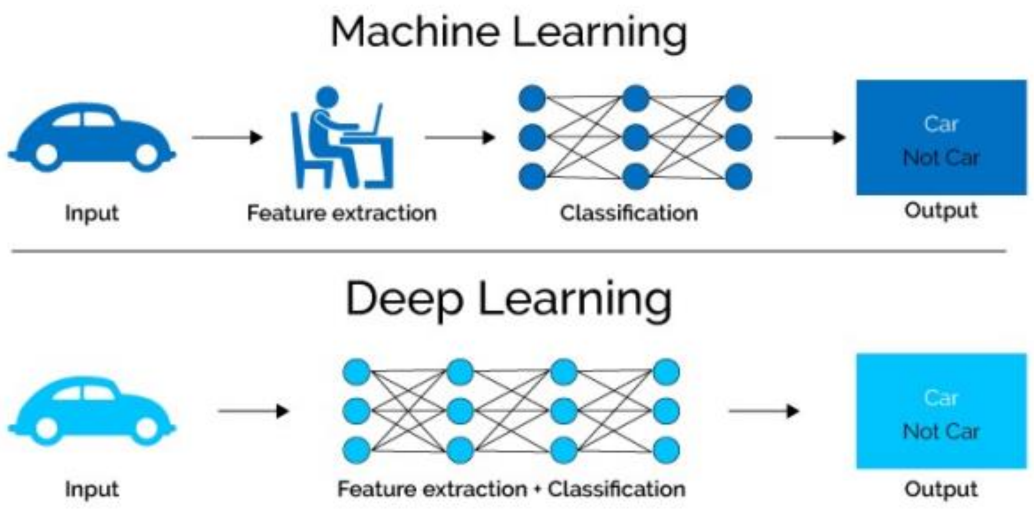


Figure 9: Comparison of ML and DL

But what is machine learning in the first place? ML is described in [24] as "the study of computer algorithms that can improve automatically through experience and use of data".

ML can be seen as a subdiscipline of artificial intelligence: ML algorithms construct a model based on data samples (training data) in order to perform prediction or classification without any explicit programming to do so. This figure [25] illustrates the four ML paradigms:

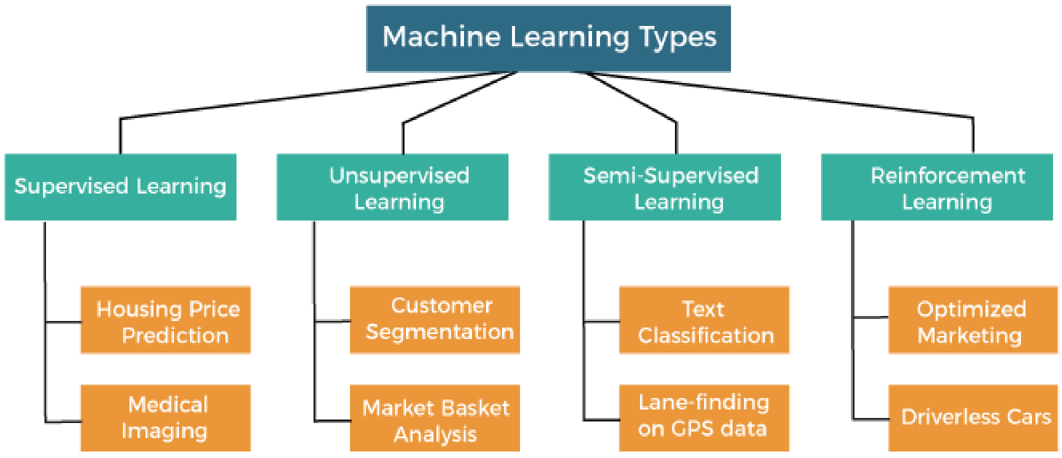


Figure 10: ML paradigms

The problem of this project is of *multivariate*, *multi-step* supervised learning:

- Multivariate: the model must not just predict one variable value, but multiple values (of wind speed, pressure, latitude, and longitude)
- Multi-step: the model cannot just predict the next unit of time, but multiple timesteps ahead (i.e. the aforementioned values 12 hours ahead as opposed to just 1 hour)
- Supervised: supervised ML infers a function from labelled training data. Involved in this is a comparison of actual output to required output. The aim is to minimise a defined loss function. In this case, the data labels (*x*) would be the previous 48 hours of values, and the target data to predict (*y*) would be the future 12 hours.

Figure 11 illustrates this. Say there are three sequences, A, B, and C, for which a model must predict future values multiple (2) timesteps ahead.

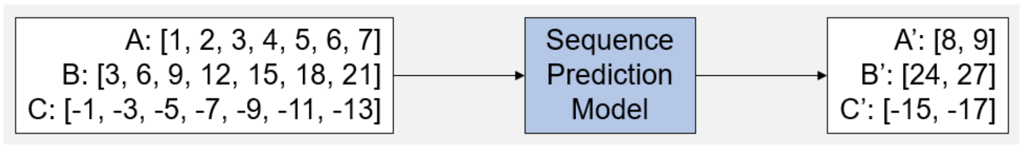

A: [1, 2, 3, 4, 5, 6, 7]
B: [3, 6, 9, 12, 15, 18, 21]
C: [-1, -3, -5, -7, -9, -11, -13]
Sequence Prediction Model
A': [8, 9]
B': [24, 27]
C': [-15, -17]

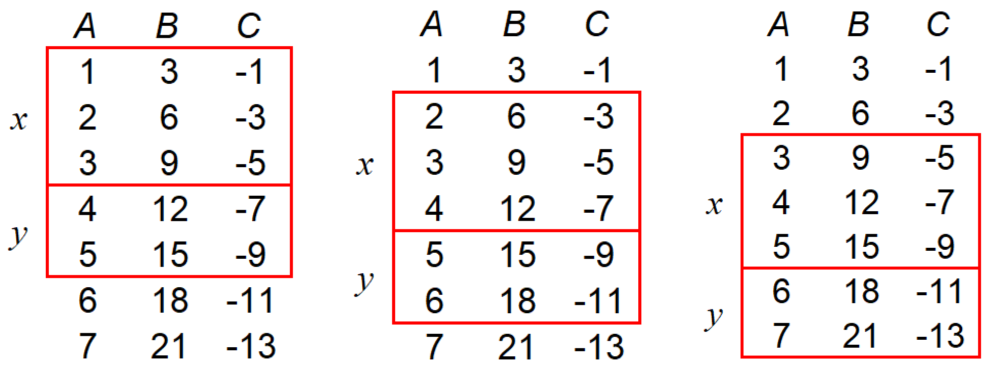
Figure 11: Example of a multivariate, multi-step prediction problem

The width of the prediction vector *y* (A', B', C') is 2 by definition, but the model could be trained with a larger input size of 3, for instance. Using sequence A as an example, part of the training dataset would be:

Table 1: Multivariate, multi-step training data example

A_x	A_y
[1, 2, 3]	[4, 5]
[2, 3, 4]	[5, 6]
[3, 4, 5]	[6, 7]

The rest of the training set would consist of similar matrices for B and C. Essentially, a 'sliding window' moves down the dataset to form the *x* labels and corresponding *y* outputs, as follows:



A	B	C
1	3	-1
2	6	-3
3	9	-5
4	12	-7
5	15	-9
6	18	-11
7	21	-13

Figure 12: Training data generation method

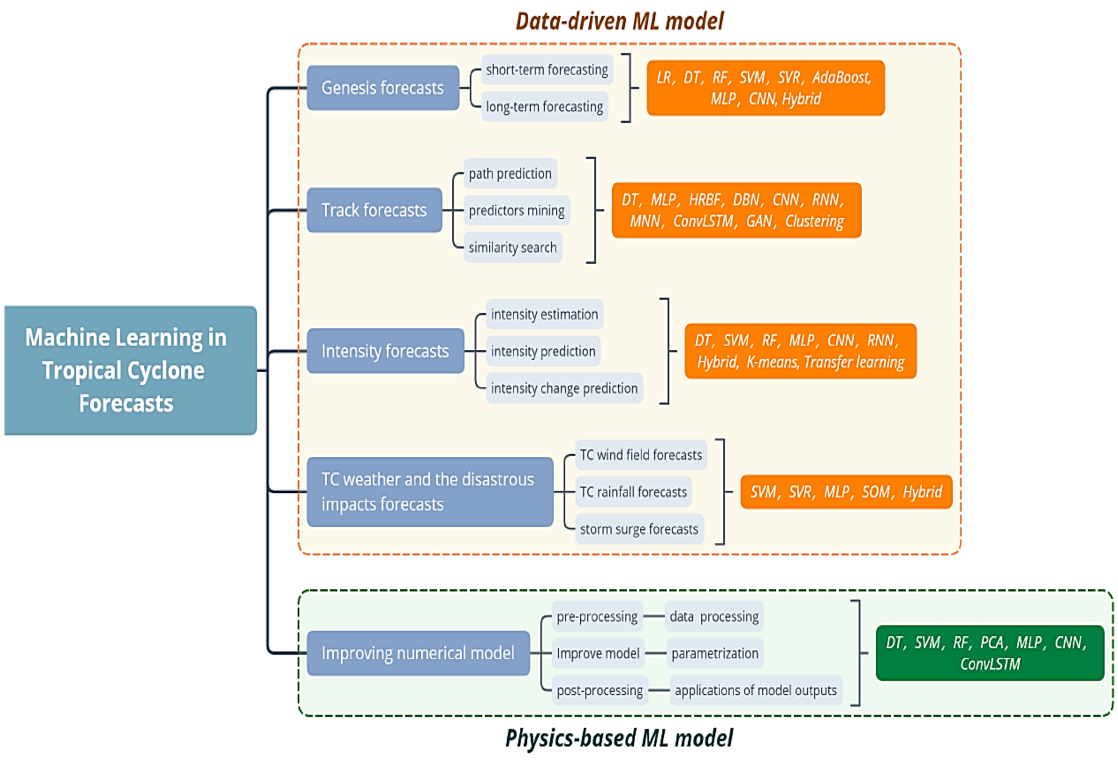
Once these data are learned, any sequence of length three of A, B, C values may be used to predict the consequent two values of each feature.

Now that this research field has been illuminated, deep learning applications in cyclone events can be critiqued.

2.3. Deep Learning in Cyclone Forecasting

Contemporary cyclone path prediction has seen much advancement throughout the past few decades. Other than rare abnormal paths, predictive systems can attain relatively accurate results [26]. However, the major challenges still holding back these DL systems include: the fact that cyclone location is much better defined than their genesis and magnitude; and the inability to accurately describe the physical processes governing cyclones, even with modern statistical models or dynamic equations. The one thing that allows DL to yield accurate trajectory results is data abundance [26]. Hence, the growing availability of satellite and reanalysis data offers tremendous opportunities for DL applied to these weather events.

The figure below [26] shows that the applications of DL in cyclone prediction can be split into five aspects. Focusing on the main task, track forecasts, DL models are typically based on statistical learning: that is, they use the intrinsic properties of cyclones and related atmospheric variables to forecast the location. Because cyclones evolve in both space and time, such models must be spatiotemporal as described in section 1.1. A primary example is convolutional LSTM (ConvLSTM), which yielded adequate predictions [27], [28], [26].



Data-driven ML model
Machine Learning in Tropical Cyclone Forecasts
Genesis forecasts
short-term forecasting
long-term forecasting
LR, DT, RF, SVM, SVR, AdaBoost, MLP, CNN, Hybrid
Track forecasts
path prediction
predictors mining
similarity search
DT, MLP, HRBF, DBN, CNN, RNN, MNN, ConvLSTM, GAN, Clustering
Intensity forecasts
intensity estimation
intensity prediction
intensity change prediction
DT, SVM, RF, MLP, CNN, RNN, Hybrid, K-means, Transfer learning
TC weather and the disastrous impacts forecasts
TC wind field forecasts
TC rainfall forecasts
storm surge forecasts
SVM, SVR, MLP, SOM, Hybrid
Improving numerical model
pre-processing
data processing
Improve model
parametrization
post-processing
applications of model outputs
DT, SVM, RF, PCA, MLP, CNN, ConvLSTM
Physics-based ML model

Figure 13: Chart organising ML-based cyclone forecasts

2.3.1. Path Prediction

To yield faster results for cyclone path forecasting which are also more effective, modern research has attempted to apply DL to build continuously ameliorated predictive models, as tabulated here [26]:

Table 2: Deep learning in cyclone track forecasts

Algorithm	Principle	Sample references
Hermite Radial Basis Function (HRBF) network	Develop an automatic cyclone identification and track mining system	[29]
Multi-layer perceptron (MLP)	Forecast cyclone paths based on MLP with backpropagation algorithms	[30], [31]
RNN	Propose an RNN with sparse, flexible topology for trajectory forecasting	[32], [33]
Multimodal attentional NN (MNN)	Develop a predictive model that preserves spatial data from cyclone paths	[34]
ConvLSTM	Propose a convolutional paradigm spatiotemporal model to track and predict cyclone paths	[27], [28]
Convolutional NN (CNN)	Design a model that uses fused multi-source data to forecast cyclone trajectories	[35]

Two decades ago, a cyclone recognition and trajectory mining system comprising two NN-based modules was proposed in [29]: the first is a cyclone pattern recognition system using a neural oscillatory elasticity map matching (NOEGM) model; the second is a cyclone trajectory mining system using a HRBF network with a time difference and structure learning (TDSL) algorithm. During experimentation of storm pattern recognition using satellite imagery, a correct segmentation rate of 98% was achieved, with a minimum correct classification rate of 97%. Moreover, the HRBF networks attained accuracies of over 86% in cyclone trajectory prediction testing. The proposed RBF network reduced prediction errors by up to 30%, compared to the one-way-interactive tropical cyclone model (OTCM) used by the Joint Typhoon Warning Centre (JTWC) in Guam and the track forecast system (TFS) used by the Central Weather Bureau in Taiwan [26].

The authors of [30] developed an MLP to predict 24-hour cyclone coordinates by using observations from the past twelve hours. Their results showed that the root-mean-square error (RMSE) between the predicted latitude and the actual latitude was 1.01 degrees, and 1.16 degrees between the predicted and actual longitude. Compared to the Climatology and Persistence (CLIPER) model, Florida State University based Limited Area Model (LAM), and Quasi Lagrangian Model (QLM), this MLP achieved greater accuracy. However, the authors observed an accuracy decrease beyond 24 hours, proving that MLP still had significant limitations in forecasting.

Some researchers considered cyclone movement as a time series in order to perform longer predictions: RNNs were used to negotiate this problem by using old information in conjunction with new inputs, to gain prediction results with more effectiveness [32], [33]. For instance, [32] used an RNN to diminish any discrepancies in statistical dynamic models. Furthermore, this algorithm can capture the features of non-linear, complex atmospheric data. When this is compared to contemporary techniques used by the NHC to predict cyclone paths, the accuracy of prediction is seen to be significantly greater [26], further corroborating the choice of RNN architecture.

Regarding the results produced by Zhang et al. [34], who developed an MNN that preserves spatial information from cyclone tracks, they show that their proposed method was more accurate compared to a diverse range of techniques – such as gated recurrent units (GRUs), MLPs, and RNNs. This is because MNNs are better suited for the chaotic nature of cyclones [26]: since traditional NNs require vectorial input, this process can be problematic as matrix input must be converted. This vectorisation can result in loss of spatial information – one of the key attributes needed for cyclone trajectory prediction, as explained next. MNNs solve this issue, as they can take matrices as direct inputs.

A final crucial consideration to make is the fact that cyclonic motion concerns both spatial and temporal dimensions. Hence, failure to consider either of these can drastically hinder model accuracy. Thus, recent research exists which attempts to use constructs of spatiotemporal models, such as the Deep-Hurricane-Tracker [28]. This ConvLSTM model was seen to perform significantly better than various baselines, following much experimentation.

Conversely, the authors of [35] incorporated past trajectory data into their developed NN model. They also performed reanalysis of atmospheric images (3D fields of wind and pressure), and used a moving frame that followed the centre of the storm for 24h forecasts [26]. The advantage of this method is that it can provide forecasts with second-like precision.

2.4. Discursive Conclusions

2.4.1. Opportunities

The task of cyclone forecasting opens the door to many bottlenecks, such as quantitative forecasts of cyclogenesis, the intense precipitation caused by cyclones, and cyclone wind field forecasts. Since the field of cyclone forecasting is still a very new research specialty, these problems require much further exploration in order to yield effective models.

Secondly, modern ML has proven useful in the effective detection and study of a variety of remote sensing-related issues. Since most satellite and reanalysis 'big data', together with general observation data, have not been fully developed and utilised, the prediction of cyclones based on multi-source data is a promising topic.

Lastly, despite the fact that numerical models are currently invaluable for cyclone path prediction [26], they have many physical processes that cannot be expressed by equations, and are costly to operate. Hence, another goal of contemporary experimentation includes the production of purely data-driven cyclone trajectory forecasting systems that ensure high levels of efficiency while being relatively inexpensive.

2.4.2. Challenges

The obfuscated, black-box nature of DL is one of its biggest disadvantages. This phenomenon has been disparaged by countless experts specialising in atmospheric sciences for its relative inability to forecast extreme weather phenomena in a realistic and stable manner. This is because the "majority of ML algorithms only discover rules hidden within the data, and is detached from the real physical rules" [26]. The development of explicable ML models, while maintaining stability, precision, and accuracy, is a challenge which has existed virtually as long as the field itself.

Climate change threatens the planet in many ways, one of which is an increased probability of cyclogenesis. Since modern DL approaches only provide a relatively short predictive time frame, model accuracy often fails to meet expectations when making long-term spatiotemporal estimates. Generating such predictions, while still conserving a high accuracy level, is perhaps the biggest problem that must be resolved as quickly as possible.

The final challenge to ponder comes from cyclones spending the majority of their lifetime over open ocean. This leads to rarity of in-situ observations, which presents the problem of data scarcity – a bane of ML modelling. Despite the large volumes of satellite image data and reanalysis data about cyclones, in addition to some airborne reconnaissance data, the total amount is nowhere near enough to aid the understanding of how cyclones truly manifest and evolve. Therefore, more observations and experiments are required to verify the ability of ML to capture the inherent physical rules of such atmospheric phenomena.

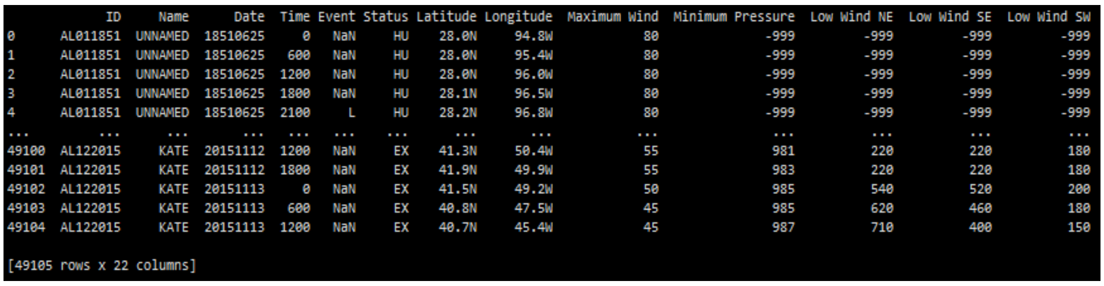
3 Dataset

Now that the programming side of this project will be discussed, it is recommended to have the code open alongside this report. This can be found here (note that the driver code *main.py* is split into sections by descriptive docstrings).

The dataset used is the NHC's Hurricane Database v2 dataset, HURDAT2. This is about 6MB large, with six-hourly information such as the location, maximum winds, and central pressure of Atlantic cyclones. It is available from Kaggle [36].

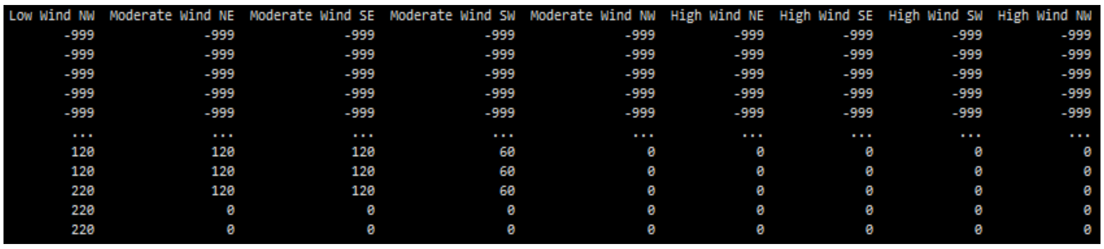
The dataset contains data about 1,814 cyclones, 512 of which have no values missing for any features. From these 512, the best are selected for model training, as will be discussed.

Missing values are represented by 'NaN' or '-999', as shown in the below screenshot (split) of the first and final 5 records of the raw data.



```
             ID     Name      Date  Time Event Status Latitude Longitude  Maximum Wind  Minimum Pressure  Low Wind NE  Low Wind SE  Low Wind SW
0      AL011851  UNNAMED  18510625     0   NaN     HU    28.0N     94.8W            80              -999         -999         -999         -999
1      AL011851  UNNAMED  18510625   600   NaN     HU    28.0N     95.4W            80              -999         -999         -999         -999
2      AL011851  UNNAMED  18510625  1200   NaN     HU    28.0N     96.0W            80              -999         -999         -999         -999
3      AL011851  UNNAMED  18510625  1800   NaN     HU    28.1N     96.5W            80              -999         -999         -999         -999
4      AL011851  UNNAMED  18510625  2100     L     HU    28.2N     96.8W            80              -999         -999         -999         -999
...         ...      ...       ...   ...   ...    ...      ...       ...           ...               ...          ...          ...          ...
49100  AL122015     KATE  20151112  1200   NaN     EX    41.3N     50.4W            55               981          220          220          180
49101  AL122015     KATE  20151112  1800   NaN     EX    41.9N     49.9W            55               983          220          220          180
49102  AL122015     KATE  20151113     0   NaN     EX    41.5N     49.2W            50               985          540          520          200
49103  AL122015     KATE  20151113   600   NaN     EX    40.8N     47.5W            45               985          620          460          180
49104  AL122015     KATE  20151113  1200   NaN     EX    40.7N     45.4W            45               987          710          400          150

[49105 rows x 22 columns]
```



```
Low Wind NW  Moderate Wind NE  Moderate Wind SE  Moderate Wind SW  Moderate Wind NW  High Wind NE  High Wind SE  High Wind SW  High Wind NW
       -999              -999              -999              -999              -999          -999          -999          -999          -999
       -999              -999              -999              -999              -999          -999          -999          -999          -999
       -999              -999              -999              -999              -999          -999          -999          -999          -999
       -999              -999              -999              -999              -999          -999          -999          -999          -999
       -999              -999              -999              -999              -999          -999          -999          -999          -999
        ...               ...               ...               ...               ...           ...           ...           ...           ...
        120               120               120                60                 0             0             0             0             0
        120               120               120                60                 0             0             0             0             0
        220               120               120                60                 0             0             0             0             0
        220                 0                 0                 0                 0             0             0             0             0
        220                 0                 0                 0                 0             0             0             0             0
```

Figure 14: Raw cyclone data

Column and feature descriptions are as follows [37]:

- ID: the ID of the cyclone. E.g. 'AL122015': AL = basin (Atlantic); 12 = storm index for the year; 2015 = year
- Name: the name given to the cyclone, e.g. Katrina
- Date/time: the record's date and time
- Event: record identifier (e.g. L = landfall, signifying the centre of a system crossing a coastline)
- Status: status of a system (e.g. HU for tropical cyclone of hurricane intensity)
- Latitude/longitude: coordinates on Earth's surface
- Maximum wind: max. sustained wind (knots)
- Minimum pressure: min. pressure (millibars)
- [Low/Moderate/High] wind [NE/NW/SE/SW]: 34/50/64 knot wind maximum radius in specified quadrant (e.g. Low Wind NE = maximum radius of 34 knot winds in north-east quadrant).

3.1. Data Cleaning

The first step of data cleaning is to remove redundant columns. The columns to keep are specified in *utils/constants.py*.

Originally, the maximum wind and minimum pressure features were excluded, then included after experimentation found that adding them enhanced general model performance. This is likely because they provide contexts to location data, which would otherwise be hidden, obscuring the trends of cyclone movement. In other words, they enable the model to do latent representation learning – learning of variables that cannot be measured directly, which therefore must be inferred from empirical measurements. Immeasurable variables in this case may pertain to the complex idiosyncrasies of cyclone movement.

Next, latitude/longitude values are converted from strings to floats to allow for processing (e.g. "$4.7S$" $\rightarrow -4.7$), and cyclones with missing values are excluded to improve data quality.

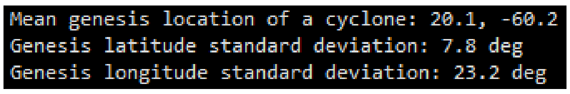
3.2. Exploratory Data Analysis

The data now requires analysis to elicit better understanding. There are 512 cyclones in the dataset with complete columns of values, but only a certain subset of these has enough feature variance to sufficiently contribute to model learning (for instance, cyclones that have travelled long enough distances). This EDA helps find the best subset of cyclone data frames with which to train the model – also because training with 512, each with many records, would be computationally expensive.

Three cyclone trajectories with some of the longest distances in the dataset look as follows (note that latitude/longitude values are normalised). These cyclones also have complex trajectories: Bertha's trajectory starts fairly straight, but has a characteristic hairpin bend halfway through; Fran also starts somewhat straight but has a sharp clockwise turn at the very end of its trajectory; and Ivan has a large loop in its trajectory. Thus, after manually exploring/plotting cyclone tracks, these were determined to be the best to make up the test set.

Since the model must learn data of a certain history size in order to predict a certain target size (e.g. using the past 48 hours to forecast the next 12 hours), EDA crucially helps to determine appropriate values of the sizes of the historical window and the window to forecast. These values will be discussed in 3.3, as they concern feature selection.

The EDA function (*data/exploratory_data_analysis.py*) first prints cyclone genesis information (*main.py* section 3):



```
Mean genesis location of a cyclone: 20.1, -60.2
Genesis latitude standard deviation: 7.8 deg
Genesis longitude standard deviation: 23.2 deg
```

Figure 15: Cyclone genesis info

These are the mean genesis coordinates plotted using Google Maps:

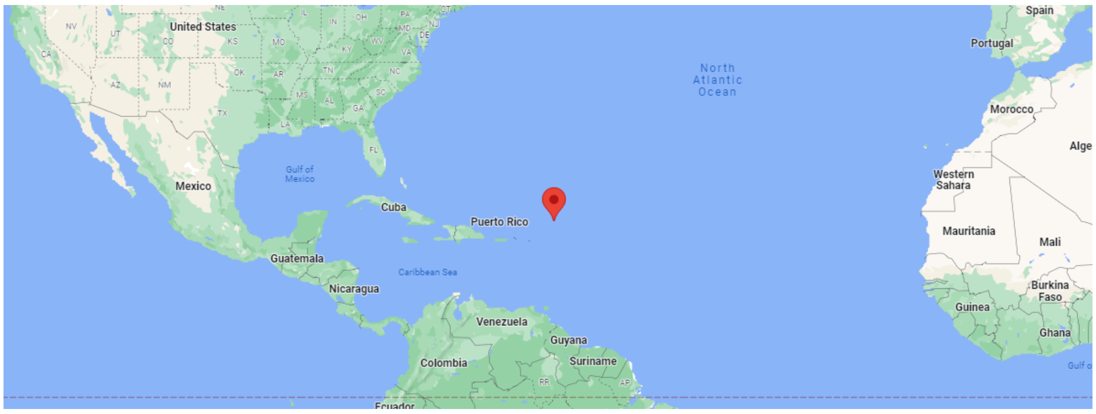
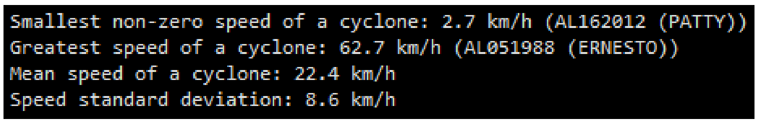
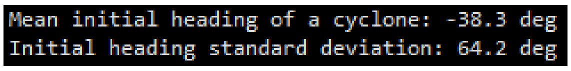


Figure 16: Mean Atlantic cyclogenesis location

Next, velocity and initial heading data are printed:



```
Smallest non-zero speed of a cyclone: 2.7 km/h (AL162012 (PATTY))
Greatest speed of a cyclone: 62.7 km/h (AL051988 (ERNESTO))
Mean speed of a cyclone: 22.4 km/h
Speed standard deviation: 8.6 km/h
```



```
Mean initial heading of a cyclone: -38.3 deg
Initial heading standard deviation: 64.2 deg
```

Figure 17: Cyclone velocity and initial heading info

These three pieces of information can be used to determine the average location of landfall of cyclones in the dataset. This distance from the mean genesis location, together with the known mean speed, will aid selection of the history size and target size parameters.

Other than these data, EDA produced bar graphs of other cyclone features such as minimum pressure across the whole dataset:

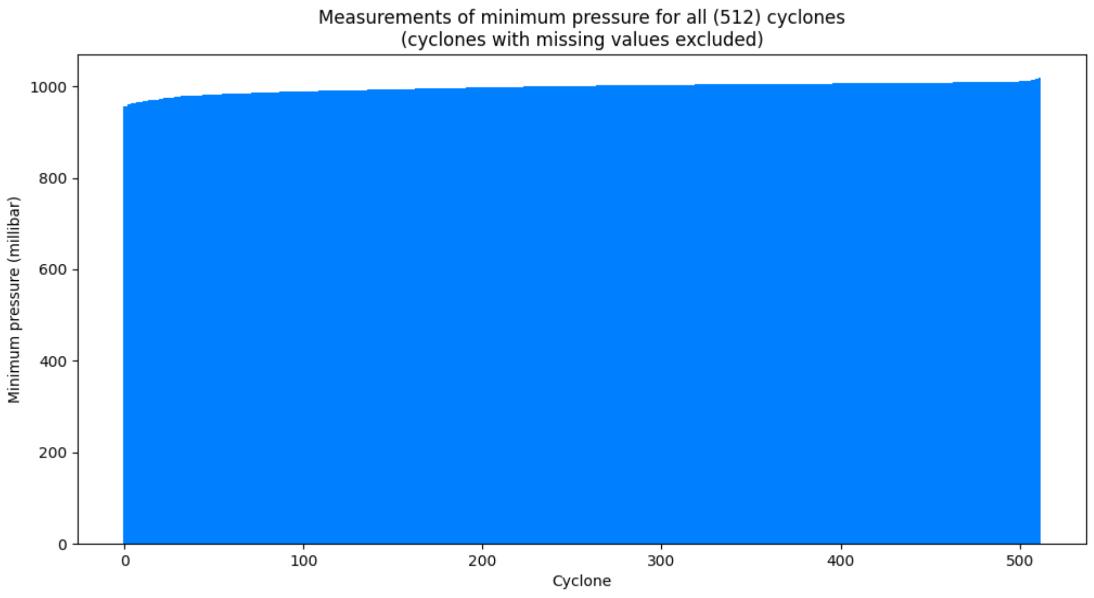


Figure 18: Minimum pressure values per cyclone (pre-augmentation)

As is evident, there is a low variance in these pressure values.

The other crucial feature, maximum sustained wind, has a slightly higher variance:

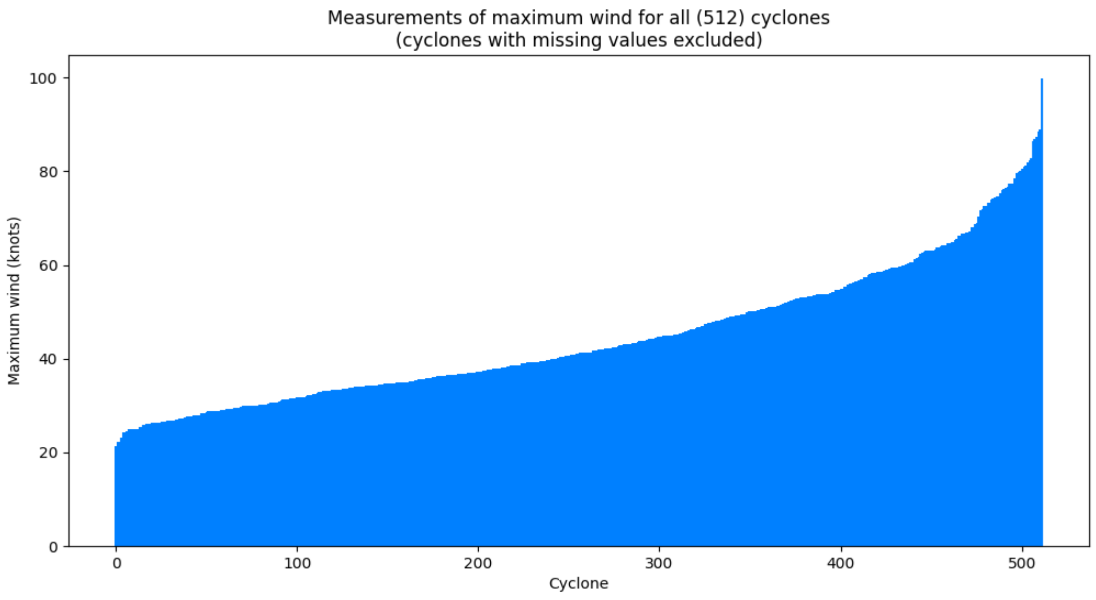


Figure 19: Maximum wind values per cyclone (pre-augmentation)

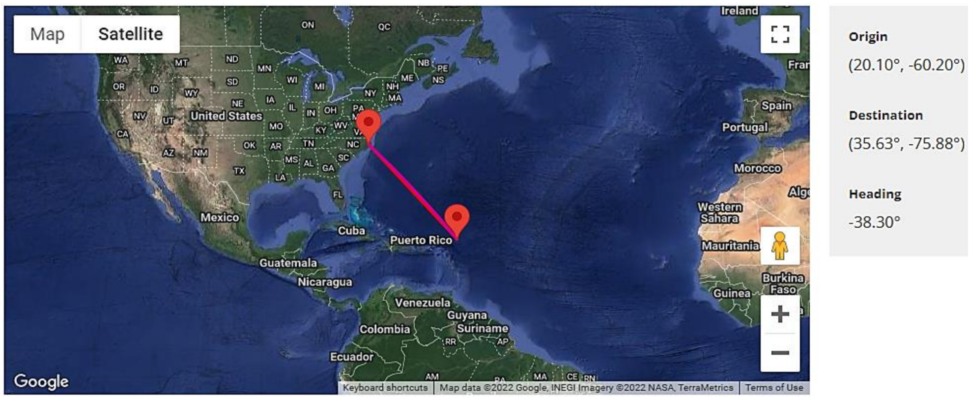
These graphs are important to note, because after the data augmentation process described in 3.4.1, their shape should remain similar. Preserving the shape of the data of these features is crucial, otherwise the model would be learning the wrong data.

3.3. Feature Selection

The selection of the 'sliding window' size is closely tied to feature selection, as it controls how many cyclones of the set of 512 will be used. So, this will be discussed as a precursor. Using the values from the EDA, the mean landfall location can be calculated:

- Mean genesis location of a cyclone = 20.1 N, 60.2 W
- Mean cyclone speed = 22.4km/h
- Mean initial heading = -38.3°

This heading can be plotted online [38]:



Origin
(20.10°, -60.20°)
Destination
(35.63°, -75.88°)
Heading
-38.30°

Figure 20: Visualisation of mean initial heading

Following this heading, the coastline of North Carolina would be crossed. From the mean genesis location, this is a Great Circle distance (distance along Earth's surface) of 2308km [39]. At a mean speed of 22.4km/h, a cyclone would take around 103 hours to reach this coast.

The largest evacuation in US history was ordered for Hurricane Rita's landfall in Texas in 2005 [40], and took 24 hours to move ~3.7 million people [41], [42]. The populations of Texas and North Carolina in 2020 were 29.1m and 10.4m, respectively [43]. Hence, NC would take just over one-third of the time to evacuate compared to Texas (8.6h). To give more time for local government evacuation plans, for instance, this can be increased to 12h. This can be the target size T of the model (given a size H of a window of historical data, predict the next T data).

Since the mean time until landfall is 103 hours, H could be 103 – 12 = 91h. However, this would be a high-risk choice, as it would involve waiting for the cyclone to traverse much distance before making predictions. Due to the high variance of initial headings, the cyclone could make earlier landfall in Florida for instance, by which it would be futile to forecast.

Hence, a history size H of 48h is selected, as this still provides a good $H:T$ ratio of 4. Partial autocorrelation (PACF) plots can corroborate this selection. These visualise the feature importance of lag variables: for instance, for an LSTM forecasting summer temperature, a lag value of around 1 year would be most useful, as the temperatures of that time would have the highest correlation. Training with temperature data from the previous winter would be useless.

The PACF plots for latitude, longitude, max. wind, and min. pressure up to the previous 24h are as follows. Evidently, these features have vanishingly small correlations with themselves, especially the further one looks back. This simply means that the value of H can be arbitrary, so long as it is sufficiently bigger than T (too small would mean that predictions would have large variations and be unreliable).

So, since all model versions use a certain H hours of data to predict the next T, appropriate cyclone data frames are needed for training: ones that have lasted at least $H + T$ hours. 336 hours (2 weeks) is an arbitrary but generous selection of this value, as it allows the *convert_data_to_history_and_target* function in *data/train_data_prep.py* to generate many (x,y) samples to feed to the model for training (*main.py* section 4).

The dataset contains 46 cyclones which have this minimum duration. The augmented versions of their data frames constitute the training, validation, and testing data. As aforementioned, EDA applied to this subset of data frames should result in similar statistics, which it does:

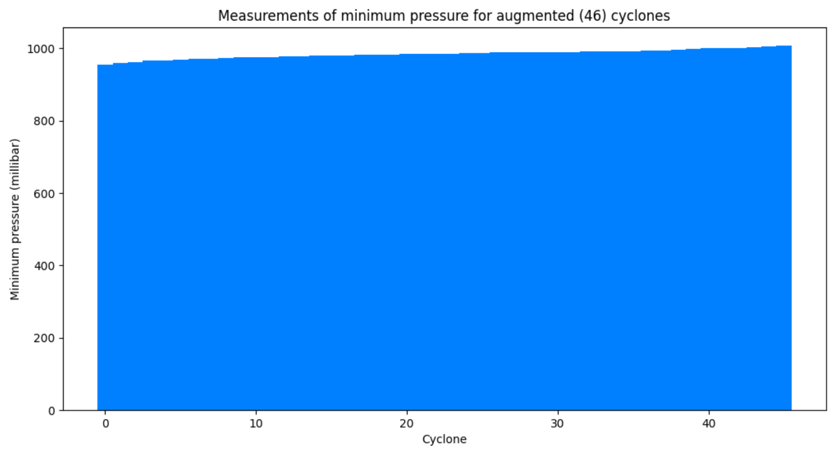


Figure 21: Minimum pressure values per cyclone (post-augmentation)

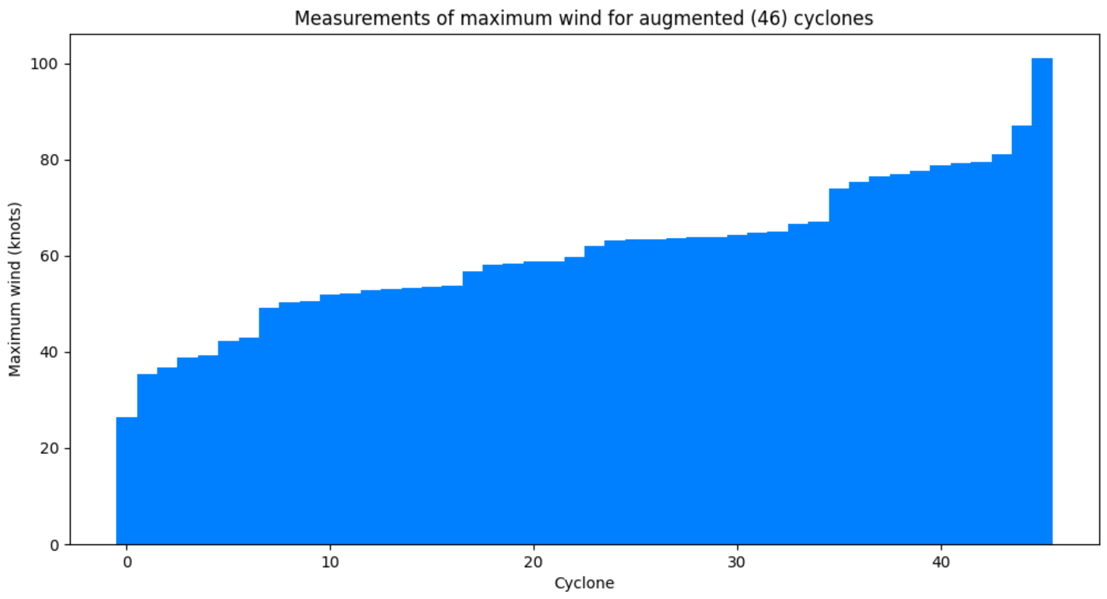


Figure 22: Maximum wind values per cyclone (post-augmentation)

As necessary, the shape of the distributions of these features is preserved. The individual bars are more defined because about 9% of the original set of cyclone data frames is being used.

As aforementioned, originally only latitude and longitude features were fed to the model for training. With experimentation with manual feature selection, the optimal features with which to train the model were found to be these in conjunction with max. wind and min. pressure:

Table 3: Model features vs performance

	Model v4 (best version) performance using…	
Test cyclone	**Only latitude/longitude features**	**Lat/long features + max. wind and min. pressure**
Bertha	Latitude MAE: 0.525° Longitude MAE: 1.947°	Latitude MAE: 0.394° Longitude MAE: 1.453°
Fran	Latitude MAE: 0.678° Longitude MAE: 1.637°	Latitude MAE: 0.514° Longitude MAE: 1.259°
Ivan	Latitude MAE: 1.910° Longitude MAE: 0.702°	Latitude MAE: 1.787° Longitude MAE: 0.634°

Where MAE signifies Mean Absolute Error. Changing the predictive features is simple, as it is a matter of adding or removing feature names to the constant FEATURES_TO_PREDICT in *utils/constants.py*. Other features, such as low wind NE, were not useful to add due to the severe scarcity of values in the dataset.

3.4. Data Pre-processing

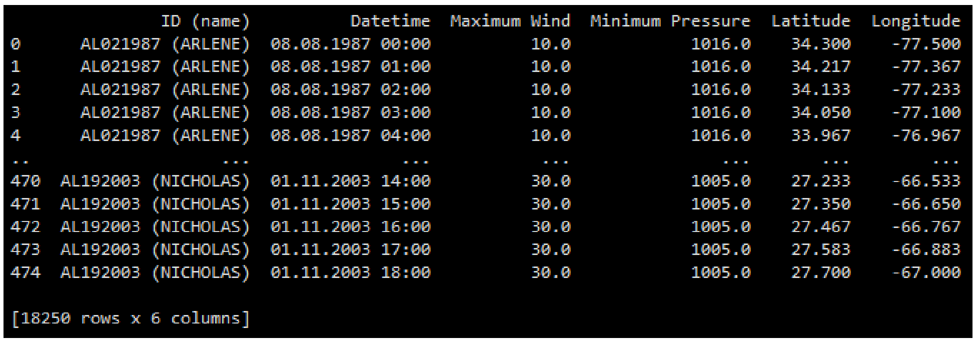
Augmentation (*main.py* section 5) and feature-wise min-max scaling (*main.py* section 9) are required before feature selection and model training/testing, respectively.

3.4.1. Augmentation

Augmentation increases the amount of data by adding synthesised data made from existing values. Here, it is used to increase the 'granularity' of the data – since cyclone records are spaced six hours apart, augmentation is used to add values such that there is a record every hour. This is beneficial to training/test data generation: in *data/train_data_prep.py*, the function *convert_data_to_history_and_target* converts a cyclone data frame to *x* (historical data) and *y* (future data to predict) arrays, via the aforementioned sliding window method. With augmented data, the window can slide one hour at a time instead of six when generating training/test data, thus allowing for more (*x*,*y*) pairs.

There is no need to add noise to the data, as it is intrinsically noisy. This is partly due to the complex and virtually indecipherable fluid dynamics of cyclones, together with the existence of noise produced by the very instruments used to measure cyclones [44].

The dataset with augmented values looks as follows. Note how the records are now spaced hourly rather than six-hourly.



```
                ID (name)          Datetime  Maximum Wind  Minimum Pressure  Latitude  Longitude
0      AL021987 (ARLENE)  08.08.1987 00:00          10.0            1016.0    34.300    -77.500
1      AL021987 (ARLENE)  08.08.1987 01:00          10.0            1016.0    34.217    -77.367
2      AL021987 (ARLENE)  08.08.1987 02:00          10.0            1016.0    34.133    -77.233
3      AL021987 (ARLENE)  08.08.1987 03:00          10.0            1016.0    34.050    -77.100
4      AL021987 (ARLENE)  08.08.1987 04:00          10.0            1016.0    33.967    -76.967
..                    ...               ...           ...               ...       ...        ...
470  AL192003 (NICHOLAS)  01.11.2003 14:00          30.0            1005.0    27.233    -66.533
471  AL192003 (NICHOLAS)  01.11.2003 15:00          30.0            1005.0    27.350    -66.650
472  AL192003 (NICHOLAS)  01.11.2003 16:00          30.0            1005.0    27.467    -66.767
473  AL192003 (NICHOLAS)  01.11.2003 17:00          30.0            1005.0    27.583    -66.883
474  AL192003 (NICHOLAS)  01.11.2003 18:00          30.0            1005.0    27.700    -67.000

[18250 rows x 6 columns]
```

Figure 23: Dataset with augmented values

This is an example track plotted with its augmented equivalent:

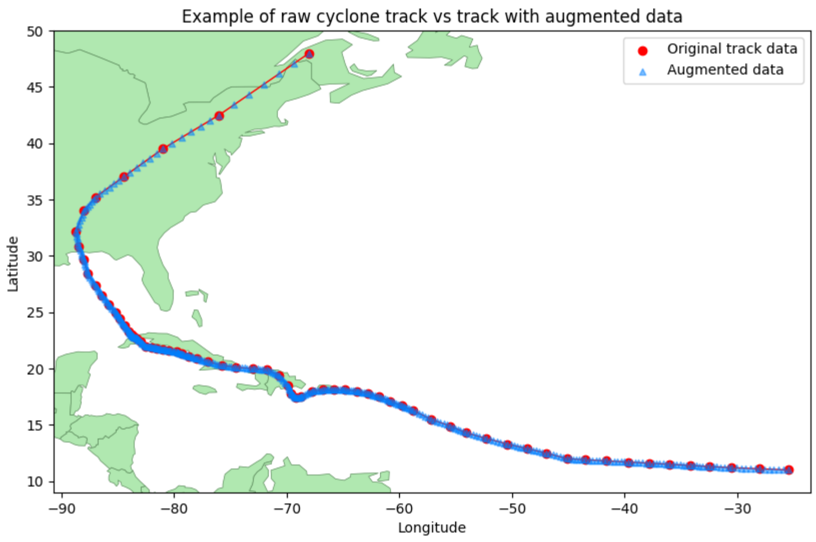


Figure 24: Raw cyclone track vs augmented equivalent

Together, these 46 augmented tracks form the training, validation, and test data (note the aforementioned recurvature):

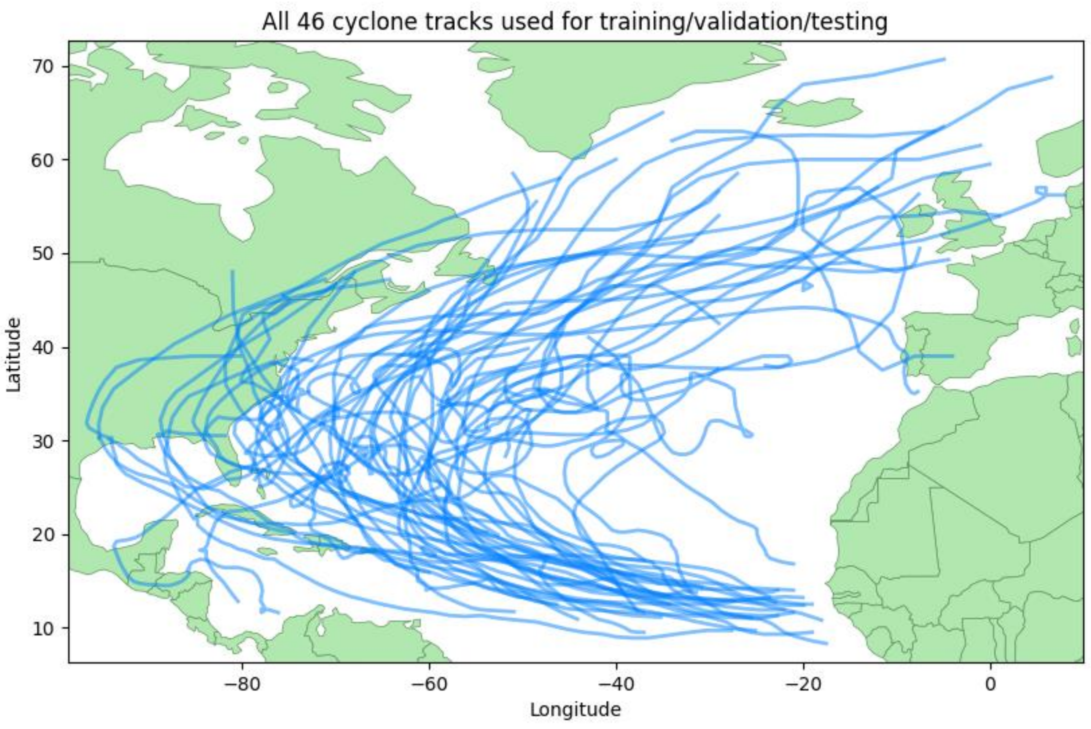


Figure 25: All cyclone tracks

3.4.2. Normalisation

Feature scaling is the final data processing step before model training. Here, numerical columns are brought into the range [0, 1] because features measured at different scales do not contribute equally to model fitting, thus allowing the model to be biased. For instance, many classifiers calculate the Euclidean distance between data: if one feature has a broad range of values, the distance would be dominated by that feature. Therefore, feature ranges should be normalised to ensure equal contributions.

The formula for min-max normalisation is given below, implemented in *main.py* section 9.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Equation 4: Formula for min-max scaling

4 Model Implementation

This section details model development. The model experienced eight iterations, where a sufficiently large difference between consecutive iterations warrants a new version number. Table 4 outlines the high-level changes between iterations/versions; the following subsections detail the reasoning behind the changes.

Table 4: High-level description of model iterations

Iteration	Changes made	Version
1	First experimental version with ConvLSTM2D layers	1
2	• Change activation of ConvLSTM2D layers to exponential linear unit (ELU) • Halve number of filters in first ConvLSTM2D layer • Halve learning rate • Add dropout after ConvLSTM2D layers	1
3	Move from ConvLSTM2D paradigm to ConvLSTM1D paradigm	2
4	• Replace final ConvLSTM1D layer with LSTM • Remove dropout layers • Change final dense layer activation to linear	2
5	• Leave convolutional paradigm completely: replace all ConvLSTM layers with LSTM • Remove BatchNormalization layer • Add a dense layer after each LSTM layer	3
6	• Experiment with increased history size (from 48h to 96h) • Extract input to its own layer for clarity • All activations changed to leaky ReLU (LReLU) • Add Bidirectional wrapper to LSTM layers (i.e. implement bi-LSTM) • Add 3 middle sections of hidden layers, each comprising a bi-LSTM cell with LReLU activation • Remove dense layers except final TimeDistributed one	4
7	• Decrease history size back to 48h • Try RMSprop optimiser to confirm that Adam is better	4
8	Double, quadruple, then multiply by 8 the number of units in all layers to determine the best magnitude	4

The evaluation of each model iteration's performance is the influencing factor on how the next iteration should be designed, and so forth until a desirable forecast accuracy is achieved.

4.1. General Architecture of each Version

The Keras API for TensorFlow offers multiple ways to construct neural networks. The implementation uses the *Sequential* class, as it is the easiest to use: models are created layer-by-layer simply via chained *model.add(layer_type)* commands.

Each model version is an encoder-decoder LSTM, an approach to sequence-to-sequence prediction problems that is proven highly effective [45]. This architecture is comprised of two models: one for reading an input sequence and encoding it into a fixed-length vector, and a second for decoding the fixed-length vector and outputting a predicted sequence [46]. This figure [45] illustrates this general architecture:

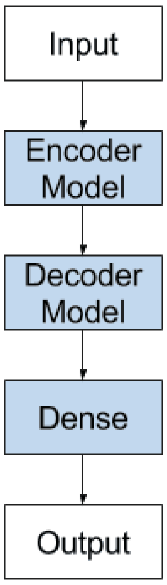


Figure 26: Architecture of encoder-decoder LSTMs

The encoder/decoder models themselves can comprise many different types of layers, such as ConvLSTM2D, ConvLSTM1D, plain LSTM, or bidirectional LSTM.

4.1.1. Optimiser

All model versions, however, use the same optimiser: optimisers are methods used to change the attributes of ML models such as weights or learning rate in order to reduce loss. The optimiser of choice here is Adaptive Moment Estimation (Adam): according to [47], this method "is computationally efficient, is invariant to rescaling of gradients, and is well suited for problems that are large in terms of data or parameters".

To corroborate this choice, the final model version is trained with Adam and RMSprop, another very popular choice. Forecast results (4.5.2) show that Adam is superior.

4.1.2. Loss Function

When *model.compile* is called post-construction, Keras configures the model for training. This function takes an optimiser and a loss function as parameters: mean squared error (MSE) is used as the latter, and is preferred over MAE as more penalty is applied if the prediction differs much from the true values.

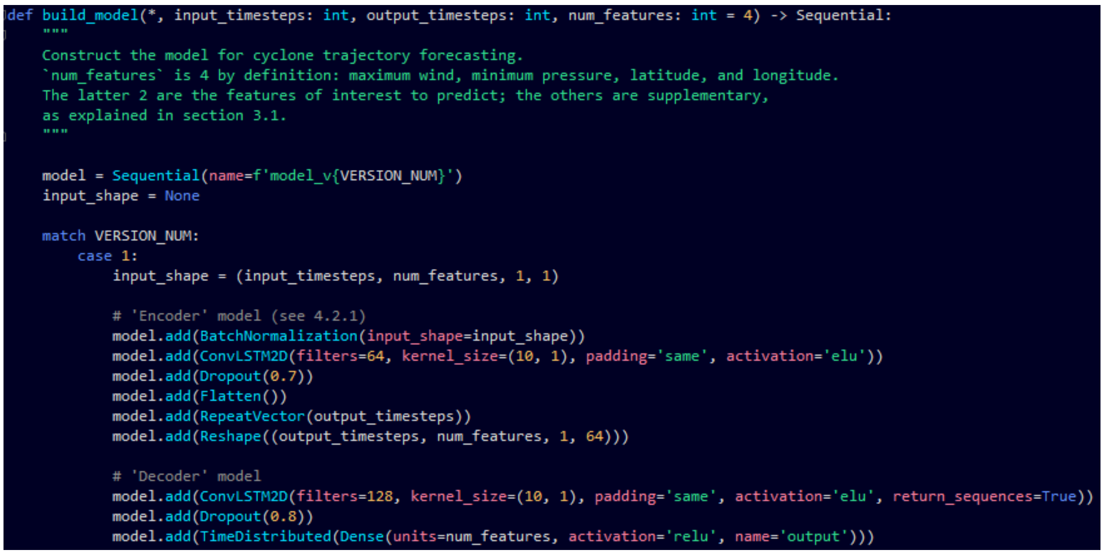
4.2. Version 1

4.2.1. Iteration 1

This version is an experimental *ConvLSTM2D* network. This layer type is typically chosen to process images, but is used here to see if it can learn cyclone data, given the correct shape: ConvLSTMs usually take 5D tensors with shape (*num_samples*, *num_time_steps*, *channels*, *rows*, *columns*).

Hence, a BatchNormalization input layer is used with shape (None, 48, 4, 1, 1). 'None' is used simply to ensure the shape is 5D; 48 = time steps; 4 = features to predict as aforementioned; rows and columns can be kept as 1, as it is not an image being fed in. Batch normalisation was used for this version, as the step of normalising data pre-training was not yet implemented. A diagram of this iteration architecture is found here.

Note that the encoder/decoder models are separated by a paragraph as shown below:



```python
def build_model(*, input_timesteps: int, output_timesteps: int, num_features: int = 4) -> Sequential:
    """
    Construct the model for cyclone trajectory forecasting.
    `num_features` is 4 by definition: maximum wind, minimum pressure, latitude, and longitude.
    The latter 2 are the features of interest to predict; the others are supplementary,
    as explained in section 3.1.
    """

    model = Sequential(name=f'model_v{VERSION_NUM}')
    input_shape = None

    match VERSION_NUM:
        case 1:
            input_shape = (input_timesteps, num_features, 1, 1)

            # 'Encoder' model (see 4.2.1)
            model.add(BatchNormalization(input_shape=input_shape))
            model.add(ConvLSTM2D(filters=64, kernel_size=(10, 1), padding='same', activation='elu'))
            model.add(Dropout(0.7))
            model.add(Flatten())
            model.add(RepeatVector(output_timesteps))
            model.add(Reshape((output_timesteps, num_features, 1, 64)))

            # 'Decoder' model
            model.add(ConvLSTM2D(filters=128, kernel_size=(10, 1), padding='same', activation='elu', return_sequences=True))
            model.add(Dropout(0.8))
            model.add(TimeDistributed(Dense(units=num_features, activation='relu', name='output')))
```

Figure 27: model_builder.py example construction

The ConvLSTM2D layers initially used Rectified Linear Unit (ReLU) activation, purely as it is a popular choice for CNN-type nets, and because further experimentation would find better activation functions.

A *RepeatVector* layer and a *Reshape* layer are required to join the encoder and decoder, to ensure correct shape of data flowing through the layers. The second ConvLSTM2D layer has a *return_sequences* parameter set to *True*, as it must output a sequence over time (as opposed to a state).

As with all iterations, a final dense layer is used, in this case with ReLU activation. Each iteration's dense layer has a *TimeDistributed* wrapper, which essentially allows the application of a layer (dense in this case) to every temporal slice of an input (in this case the returned sequences of the previous ConvLSTM2D layer).

Training this model resulted in gradient explosion (loss values grew extremely large and thus are impossible to graph). This is likely because the first ConvLSTM2D layer had too many filters (128), or because of the high learning rate (0.01).

4.2.2. Iteration 2

For this iteration, the ConvLSTM2D activations were changed to exponential linear unit (ELU), to alleviate the exploding gradient problem. Also, for negative inputs, the outputs are non-zero, unlike ReLU. This avoids ReLU's 'dead neuron' problem [48].

The number of filters in the first ConvLSTM2D layer was halved to 64, together with the learning rate (halved to 0.005); dropout layers are used after ConvLSTM2D layers to help reduce gradient explosion, with rates of 0.7 and 0.8. Implementing dropout means to randomly remove some hidden neurons and their connections during training (this acts as a regulariser, as it aids minimisation of the loss function together with minimising underfitting/overfitting).

During this iteration it was discovered that adding more layers (ConvLSTM2D/dropout pairs) made virtually no difference to performance. The architecture for this iteration is found here.

Over one million trainable parameters meant that training this model took approximately two hours; the resultant loss graph is as follows. Although it plateaus at what seems like a low MSE, this version's best forecast of the test set is nevertheless poor. This is likely due to ConvLSTM2D layers being unnecessarily complex, or altogether unsuited to cyclone data.

4.3. Version 2

4.3.1. Iteration 3

This iteration simply moved from ConvLSTM2D to ConvLSTM1D layers, but maintained the same parameters as the best iteration of v1:

- 64 filters in first ConvLSTM1D layer
- ELU activations in ConvLSTM1D layers
- Use of dropout, with the same rates
- Learning rate of 0.005.

However, this resulted in worse loss than iteration 2.

4.3.2. Iteration 4

Here, the last ConvLSTM1D layer was replaced with a 'plain' LSTM layer, to further the model's simplicity. Dropout layers were removed, as for sufficiently large or complex datasets: "regularisation confers little reduction in generalisation error; in these cases, the computational cost of using dropout and larger models may outweigh the benefit of regularisation" [49].

The activation of the dense layer was changed from ReLU to linear, in order to test an alternative to ELU (i.e. another function that produces non-zero outputs for negative inputs).

The best-found architecture of this version is here, together with its loss graph and best test forecast. This is the first iteration whose training loss was lower than the validation loss, as opposed to previous iterations. For these, which used dropout, validation loss may have been lower because dropout penalises variance of models by randomly selecting neurons in a certain layer to freeze during training. Since dropout is applicable solely during training and affects training loss, it can produce cases where validation loss is lower.

4.4. Version 3

4.4.1. Iteration 5

As pre-training data normalisation was implemented prior to this iteration, the BatchNormalization layer was removed. Secondly, only LSTM layers were used (the convolutional paradigm was abandoned), but dense layers were added throughout after LSTM layers, to maintain complexity (architecture).

Making this relatively small change had already reduced loss compared to the final iteration of version 2, as shown here.

However, the forecast was still poor, most likely due to overfitting on training/validation data (if these losses approach or equal zero, then by definition loss cannot be zero on test data).

4.5. Version 4

4.5.1. Iteration 6

Here, it was tested whether or not a larger history size would affect the final loss compared to v3, by increasing it from 48h to 96h in *utils/model_params.py*. However, there was negligible change.

After this, the input was extracted to its own layer for clarity, and all activations were changed to leaky ReLU (LReLU), yet another alternative to ELU.

Finally, bi-LSTM layers were introduced into the three middle sections, with removal of dense layers to yield more simplicity. This resulted in some good forecasts, for instance this one, due to the fact that LSTM layers may be more adept to sequential problems than dense layers. This may be furthered by use of the bidirectional wrapper, which allows processing of sequences in both directions, as discussed. Note that forecasts are now non-normalised, allowing for more familiarity with MAE values.

4.5.2. Iteration 7

Iteration 7 saw the decrease of history size back to 48h, and experimenting with the RMSprop optimiser to confirm that Adam is better.

This example forecast shows that this is clearly the case.

4.5.3. Iteration 8

The final iteration experimented with doubling, quadrupling, then multiplying by 8 the number of units in all layers to find the best magnitude. It was determined that doubling was the best. This final version resulted in 2,144,260 trainable parameters, resulting in ~5h of training.

An example forecast using 8 times the units is shown here; and one with only double the units here. The total MAE of the former is about 19% more than that of the latter; thus, the latter constitutes the final model version (architecture).

The final loss graph is clearly superior to precedent ones: it has the lowest ultimate loss values; and is smooth, meaning that the model is able to successfully deal with the high variances of the dataset [50] (which were discovered during EDA).

5 Evaluation and Results

In *main.py* section 8, the user is asked to train a model or load an existing one to evaluate. If the former is selected, a model of the version number specified in *utils/constants.py* is built by *model_builder.py*. It is then trained with the appropriate parameters given the model version, in *utils/model_params.py*.

During training, the Keras *EarlyStopping* class is used to end the training process when a monitored metric has stopped improving. See comments in *model_training.py* for a detailed explanation.

If the 'load' option is selected, the best model of the specified version number is loaded and evaluated in *main.py* section 10 (training, in section 9, is skipped).

As testing was interleaved and discussed with and throughout the model implementation, only the best-found result may be interpreted here for the sake of conciseness (section 4 saw the model evaluated with multiple combinations of layer types, activation functions, etc.).

The best forecast found has latitude and longitude MAEs of 0.680 and 0.937, respectively. Adding these two MAE values together, we have a total mean absolute error of 1.617°. Using the equation $l = r\theta$ (arc length of a circle given radius r and angle θ in radians), the error can be found as a Great Circle distance (along Earth's surface):

$$\theta = 1.617 \times \frac{\pi}{180} = 0.02822 \text{ rad}$$

Since the radius of Earth is 6371km [51]:

$$l = r\theta = 6371 \times 0.02822$$
$$= 179.8\text{km}$$

This means that, for a 12-hour forecast, there would be a positional predictive error of around 112 miles or approximately four times the size of Raleigh [52], North Carolina's capital city.

Compared to some of the best models found during literature reading:

Table 5: Model performance vs existing models

Model type	Creators	MAE	Relative size of MAE compared to model v4
CNN	[53]	115km = 71.9 miles (evaluated on Atlantic cyclones)	0.642
LSTM	[54]	1.7398° (averaged)	1.076

the model developed here achieved an MAE approximately 2.5 times higher and 1.5 times lower, respectively. These are reasonable values, as both models in the above table had a team of creators developing them, and were sufficiently complex. With more time and resources, the model developed here could have been developed such as to rival this complexity, by exploring more combinations of layer types and hyperparameter values – the numbers of which grow exponentially the more layers are added.

6 Discursive Conclusions

This paper has suggested a bidirectional long short-term memory network to predict cyclone trajectories twelve hours long. Results show that the final model performs efficiently in predicting the paths of these atmospheric phenomena based on different features like wind speed, minimum central pressure, and geographic location.

The final learning curve has low final loss values (MSEs of 0.001° and 0.002° for latitude and longitude, respectively), and shows that the model is able to negotiate the high variances of the dataset.

However, the model does have limitations which give way to future work that may be conducted, should this model be officially deployed. These will now be divulged.

6.1. Limitations and Possible Revisions

6.1.1. Model

Firstly, the LSTM paradigm has some drawbacks:

- Relatively much time and many resources are required to train these models to prepare them for real-world applications; they need high memory bandwidth due to the complexity of each cell. Thus, LSTMs are relatively inefficient models when it comes to hardware. A solution for this would be simply to train them on appropriately powerful hardware such as dedicated GPUs or cloud servers.
- Overfitting is an issue to which LSTMs are fairly prone, and applying the dropout method to suppress this problem is a notoriously difficult task. A way to automate selection of optimal dropout parameters (and any model parameters in fact) could be to use tools such as scikit-learn's GridSearchCV. This is a class that tries all possible hyperparameter values automatically, and returns the model with the best ones.

Secondly, the aforementioned black-box nature of many ML models applies here: how can one be sure that this model is discovering the true physical laws behind cyclonic phenomena, rather than mere patterns in the data? This issue pertains to the immense challenge of creating explicit, *white*-box ML models, a requirement which is out of the scope of this project – but nevertheless a crucial issue to consider and conduct more research on (applied to cyclone prediction contexts), given more time.

Another issue is the lack of confidence intervals to accompany forecasts. Should this model be deployed, it would need a range of estimates for its predictions, as with any life-critical ML application. A possible amendment for this could be a Bayesian treatment of the LSTM. In general, a Bayesian treatment of NNs treats all sources of hyperparameters in a probabilistic way, and potentially even network architectures.

Any well-defined NN can be modelled in a Bayesian manner. To advance this revision even further, relevant physical laws such as the laws of fluid dynamics may be incorporated as Bayesian priors into the LSTM construction – this would involve working in a team with physicists or mathematicians.

Finally, an ensemble technique could be explored, to see if the 'wisdom of the crowd' of conjunctive models out-performs individual models.

6.1.2. Data

The selection of cyclone data frames with at least 336 hours impedes the volume of data with which the model is trained. The 46 cyclones that have this duration comprise only around 9% of the set of data frames with complete column values. Hence, the value of 336 hours may be further modified, to strike an optimal balance of training data volume and variance (e.g. the smaller this value, the more cyclones with which to train, but the less variance of data in individual shorter-duration cyclones which may curtail learning).

Regarding the augmentation method, this should not necessarily be linear. For instance, at points where a cyclone turns, augmentation should produce a curve rather than 'sharpening' the existing turn point. Here, Bézier curves may be used:

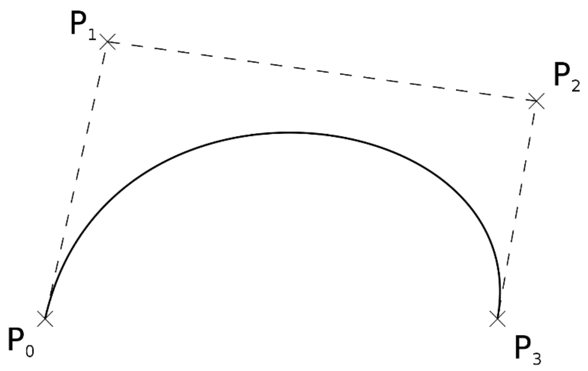


Figure 28: Use of Bézier curves in augmentation

These are smooth, continuous parametric curves defined by a set of control points by means of a formula. Such curves approximate real-world shapes that have no mathematical representation, or whose representation is too complicated. If P_0, P_1, P_2, P_3 represent existing location points in a cyclone data frame, then the resultant Bézier curve can be the augmented cyclone track. This would be more realistic than the current implementation.

6.1.3. Training

Training wise, k-fold cross-validation may be used: this is a technique in which a dataset is divided into k non-overlapping folds. Each of these folds is given an opportunity to be used as a test set, while all other folds are used collectively for training. A total of k models are fit and evaluated on the k test sets, and the mean performance is reported. However, this method would take a lot of time to execute, especially with large cyclone datasets. Powerful hardware would also be useful here.

Finally, a metaheuristic method such as the genetic algorithm (GA) or simulated annealing (SA) may be used to determine optimal model hyperparameters. This is because these two methods are immensely popular for solving combinatorial optimisation problems, or constraint satisfaction problems. In this case, a GA or SA approach could find the best combination of learning rate, activation functions, layer types, number of units and so on, which produces the most effective model. This would be an alternative to GridSearchCV.

6.2. Reflections

A problem encountered took place during the splitting of the dataset via the sliding window method. The data was originally split horizontally (the window moved 'left-to-right'), although vertical splitting was discovered to be correct afterwards.

Other difficulties simply revolved around bug fixing, such as obtaining the correct order of input shape for the initial layers, e.g. (None, 48, 4, 1, 1) as opposed to (1, 1, 4, 48, None). Locating and amending these minor issues occurred only on a timescale of hours.

Should this project be continued or restarted, an important consideration to make is use of a formal ML pipeline. These are means of automating ML workflows by enabling data to be transformed and correlated into a model that can then be analysed to achieve outputs. This type of ML pipeline makes the process of inputting data into the ML model fully automated.

Another type of pipeline is the art of dividing ML workflows into independent, reusable, modular parts that can then be pipelined together to create models. This type of ML pipeline renders model creation more efficient and simplified, minimising redundant work.

A final reflection concerns the application of the model to other cyclone datasets (of cyclones from other geographic locations), as the governing physical laws are the same. For instance, the Indian Ocean is a hotspot for cyclone activity. As it is in the southern hemisphere, the only difference would be the reversal of the Coriolis force. With more time, a dataset describing cyclones around this ocean could be used to evaluate the model.

6.3. Conclusion

Should the model be deployed, it could see much improvement with the aforementioned revisions. Furthermore, as cyclones are occurring more frequently with the increase of global ocean surface temperatures, this means that more cyclone data can be collected. This in turn signifies the possibility of creating larger datasets which could further hone model accuracy.

Another point regarding data use, specifically data recency, is to do with the effect of global warming on the feature values themselves. For example, training with cyclone data from the 1950s may have a detrimental effect on model performance, as cyclones from this long ago would have been quite different from ones that occur today (e.g. they may have intensified much more slowly due to cooler oceans), and thus be useless. Older cyclones may also have more inaccurate data due to worse equipment than is available now, further obscuring their true behaviour.

Overall, this was an eye-opening project to many nuances of machine learning, not only pertaining to cyclone trajectory forecasting but to the field in general. The best model for a given dataset cannot be known; it is the job of a machine learning practitioner to use careful experiments and discover what works best. This project allowed much realisation of this.

References

[1] P. Wang, P. Wang, W. Cong, B. Xue and D. Wang, “Using a 3D convolutional neural network model and gated recurrent unit for tropical cyclone track forecasting,” *Atmospheric Research,* p. 269, 2022.

[2] National Oceanic and Atmospheric Administration, “Frequently Asked Questions,” 1 June 2021. [Online]. Available: https://www.aoml.noaa.gov/hrd-faq/#what-is-a-hurricane. [Accessed 15 April 2022].

[3] S. King, “Forecasters predict a very active hurricane season,” BBC News, 24 May 2022. [Online]. Available: https://www.bbc.co.uk/news/science-environment-61571996. [Accessed 27 May 2022].

[4] SciJinks, “What Are Trade Winds?,” NOAA, 14 July 2022. [Online]. Available: https://scijinks.gov/trade-winds/#:~:text=The%20trade%20winds%20are%20air,used%20by%20sailors%20for%20centuries.. [Accessed 31 July 2022].

[5] NOAA, “What are El Niño and La Niña?,” 2 June 2019. [Online]. Available: https://oceanservice.noaa.gov/facts/ninonina.html. [Accessed 10 May 2022].

[6] S. Seneviratne and N. Nicholls, “Changes in Climate Extremes and their Impacts on the Natural Physical Environment,” March 2018. [Online]. Available: https://www.ipcc.ch/site/assets/uploads/2018/03/SREX-Chap3_FINAL-1.pdf. [Accessed 10 June 2022].

[7] A. Khan, “Why neural networks are gaining so much popularity,” May 2022. [Online]. Available: https://www.linkedin.com/posts/ashbabkhan_why-the-neural-network-is-so-popular-in-the-activity-6932596635617107968-tmBh?utm_source=linkedin_share&utm_medium=member_desktop_web. [Accessed 23 May 2022].

[8] Emergent Garden, “Why Neural Networks can learn (almost) anything,” YouTube, 12 March 2022. [Online]. Available: https://www.youtube.com/watch?v=0QczhVg5HaI. [Accessed 10 July 2022].

[9] J. Brownlee, “What is time series forecasting?,” Machine Learning Mastery, 2 December 2016. [Online]. Available: https://machinelearningmastery.com/time-series-forecasting/. [Accessed 25 April 2022].

[10] D. Gjylapi, E. Proko and A. Hyso, “Recurrent Neural Networks in Time Series Prediction,” *Journal of Multidisciplinary Engineering Science and Technology,* vol. 5, no. 10, p. 8741, 2018.

[11] S. Basodi, C. Ji, H. Zhang and Y. Pan, “Gradient amplification: An efficient way to train deep neural networks,” *Big Data Mining and Analytics,* vol. 3, p. 198, 2020.

[12] C. Costa, “Best Python Libraries for Machine Learning and Deep Learning,” Towards Data Science, 24 March 2020. [Online]. Available:

https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c. [Accessed 1 May 2022].

[13] Python Engineer, “I built the same model with TensorFlow and PyTorch | Which Framework is better?,” YouTube, 24 April 2022. [Online]. Available: https://youtu.be/ay1E1f8VqP8. [Accessed 30 April 2022].

[14] A. Gupta, “Steps to Complete a Machine Learning Project,” Analytics Vidhya, 16 April 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/04/steps-to-complete-a-machine-learning-project/. [Accessed 18 May 2022].

[15] Be Smart, “This Machine Makes 300+ km/h winds (indoor hurricane!),” YouTube, 2 June 2022. [Online]. Available: https://www.youtube.com/watch?v=g4SjaZb1AIM. [Accessed 2 August 2022].

[16] S. Seman, “Meet the Tropics,” PennState College of Earth and Mineral Sciences, 2020. [Online]. Available: https://www.e-education.psu.edu/meteo3/l11_p2.html#:~:text=The%20tropics%20are%20commonly%20defined,to%2030%2Ddegrees%20South%20latitude.. [Accessed 10 August 2022].

[17] W. Gray, “Global view of the origin of tropical disturbances and storms,” *Monthly Weather Review,* vol. 96, no. 10, pp. 669-700, 1968.

[18] W. Gray, “Hurricanes: Their formation, structure and likely role in the tropical circulation,” *Meteorology over the tropical oceans,* vol. 155, p. 218, 1979.

[19] NOAA's Atlantic Oceanographic and Meteorological Laboratory, “Anatomy and Life Cycle of a Hurricane,” U.S. Department of Commerce, 2022. [Online]. Available: https://www.aoml.noaa.gov/hrd-faq/#causes-and-effects. [Accessed 9 August 2022].

[20] National Weather Service, “Tropical Cyclone Structure,” National Oceanic and Atmospheric Administration, [Online]. Available: https://www.weather.gov/jetstream/tc_structure#:~:text=The%20main%20parts%20of%20a,top%20in%20the%20opposite%20direction.. [Accessed 28 July 2022].

[21] H. Willoughby, “Temporal changes of the primary circulation in tropical cyclones,” *Journal of the Atmospheric Sciences,* vol. 47, pp. 242-264, 1990.

[22] R. Tuleya, “Tropical storm development and decay: Sensitivity to surface boundary conditions,” *Monthly Weather Review,* vol. 122, pp. 291-304, 1994.

[23] P. S. Prasad and A. Senthilrajan, “Leaf Features Extraction for Plant Classification using CNN,” *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT),* vol. 2, no. 2, p. 149, 2021.

[24] T. Mitchell, Machine Learning, New York: McGraw Hill, 1997.

[25] JavaTpoint, “Types of Machine Learning,” 2021. [Online]. Available: https://www.javatpoint.com/types-of-machine-learning. [Accessed 6 May 2022].

[26] R. Chen, W. Zhang and X. Wang, “Machine Learning in Tropical Cyclone Forecast Modeling: A Review,” *Atmosphere,* vol. 11, 2020.

[27] S. Kim, J.-S. Kang, M. Lee and S.-K. Song, “DeepTC: ConvLSTM Network for Trajectory Prediction of Tropical Cyclone using Spatiotemporal Atmospheric Simulation Data,” *32nd Conference on Neural Information Processing Systems (NeurIPS),* 2018.

[28] K. Sookyung, H. Kim, J. Lee, S. Yoon, S. E. Kahou and K. Kashinath, “Deep-hurricane-tracker: Tracking and forecasting extreme climate events,” *IEEE Winter Conference on Applications of Computer Vision (WACV),* pp. 1761-1769, 2019.

[29] R. Lee and L. James, “Tropical cyclone identification and tracking system using integrated neural oscillatory elastic graph matching and hybrid RBF network track mining techniques,” *IEEE Transactions on Neural Networks,* vol. 11, no. 3, pp. 680-689, 2000.

[30] M. Ali, C. Kishtawal and S. Jain, “Predicting cyclone tracks in the north Indian Ocean: An artificial neural network approach,” *Geophysical research letters,* vol. 34, no. 4, 2007.

[31] W. Yuanfei, W. Zhang and W. Fu, “Back Propogation (BP)-neural network for tropical cyclone track forecast,” *19th International Conference on Geoinformatics, IEEE,* pp. 1-4, 2011.

[32] S. Alemany, J. Beltran, A. Perez and S. Ganzfried, “Predicting Hurricane Trajectories Using a Recurrent Neural Network,” *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19),* vol. 33, no. 1, pp. 468-475, 2019.

[33] M. M. Kordmahalleh, M. G. Sefidmazgi and A. Homaifar, “A Sparse Recurrent Neural Network for Trajectory Prediction of Atlantic Hurricanes,” *Proceedings of the Genetic and Evolutionary Computation Conference,* pp. 957-964, 2016.

[34] Y. Zhang, C. Rohitash and J. Gao, “Cyclone track prediction with matrix neural networks,” *International Joint Conference on Neural Networks (IJCNN),* pp. 1-8, 2018.

[35] S. Giffard-Roisin, M. Yang, G. Charpiat, C. K. Bonfanti, B. Kegl and C. Monteleoni, “Tropical cyclone track forecasting using fused deep learning from aligned reanalysis data,” *Frontiers in Big Data,* vol. 1, 2020.

[36] National Hurricane Center, “Hurricanes and Typhoons, 1851-2014,” 2017. [Online]. Available: https://www.kaggle.com/datasets/noaa/hurricane-database. [Accessed 3 February 2022].

[37] C. Landsea, “The revised Atlantic hurricane database (HURDAT2),” April 2022. [Online]. Available: https://www.nhc.noaa.gov/data/hurdat/hurdat2-format-atl-1851-2021.pdf. [Accessed 30 April 2022].

[38] academo, “Geodesics on the Earth,” [Online]. Available: https://academo.org/demos/geodesics/. [Accessed 2 June 2022].

[39] Movable Type Ltd, “Calculate distance, bearing and more between Latitude/Longitude points,” [Online]. Available: https://www.movable-type.co.uk/scripts/latlong.html. [Accessed 4 June 2022].

[40] R. D. Knabb, D. P. Brown and J. R. Rhome, “Tropical Cyclone Report: Hurricane Rita,” 2005. [Online]. Available: https://www.nhc.noaa.gov/data/tcr/AL182005_Rita.pdf. [Accessed 7 June 2022].

[41] P. O'Driscoll, R. Wolf and R. Hampson, “Hurricane Rita,” USA Today, 25 September 2005. [Online]. Available: https://usatoday30.usatoday.com/news/nation/2005-09-25-evacuation-cover_x.htm. [Accessed 7 June 2022].

[42] K. Soika, “Evacuation Planning in Texas: Before and After Hurricane Rita,” *Interim News (House Research Organization, Texas House of Representatives),* vol. 79, no. 2, p. 2, 2006.

[43] U.S. Census Bureau, Population Division, “State Population by Rank,” Infoplease, 2020. [Online]. Available: https://www.infoplease.com/us/states/state-population-by-rank. [Accessed 8 June 2022].

[44] D. Herres, “Understanding noise in electronic instrumentation,” EE World Online, 17 January 2020. [Online]. Available: https://www.testandmeasurementtips.com/understanding-noise-in-electronic-instrumentation/. [Accessed 12 June 2022].

[45] J. Brownlee, “Encoder-Decoder Long Short-Term Memory Networks,” Machine Learning Mastery, 23 August 2017. [Online]. Available: https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/. [Accessed 27 May 2022].

[46] K. Cho, B. v. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,” p. 1, 2014.

[47] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” p. 1, 2014.

[48] N. S. Chauhan, “An Overview of Activation Functions in Deep Learning,” The AI Dream, 26 April 2020. [Online]. Available: https://www.theaidream.com/post/an-overview-of-activation-functions-in-deep-learning. [Accessed 10 July 2022].

[49] I. Goodfellow, Y. Bengio and A. Courville, “Deep Learning,” The MIT Press, 2016, p. 265.

[50] J. Brownlee, “How to use Learning Curves to Diagnose Machine Learning Model Performance,” Machine Learning Mastery, 27 February 2019. [Online]. Available: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/. [Accessed 15 August 2022].

[51] D. R. Lide and H. P. Frederikse, CRC Book of Chemistry and Physics, CRC Press, 2000.

[52] United States Census Bureau, “2019 U.S. Gazetteer Files,” 2019. [Online]. Available: https://www2.census.gov/geo/docs/maps-data/data/gazetteer/2019_Gazetteer/2019_gaz_place_37.txt. [Accessed 28 August 2022].

[53] S. Giffard-Roisin, M. Yang, G. Charpiat, B. Kegl and C. Monteleoni, “Deep Learning for Hurricane Track Forecasting from Aligned Spatio-temporal Climate Datasets,” 2018.

[54] W. Qin, J. Tang, C. Lu and S. Lao, “Trajectory prediction based on long short-term memory network and Kalman filter using hurricanes as an example,” *Computational Geosciences,* vol. 25, no. 3, pp. 1005-1023, 2021.

Appendices

Main Body Word and Page Count

Words: 9,960

Pages: 31

Clarification of Hurricanes, Cyclones, and Typhoons

- A cyclone is a large air mass that rotates around a centre of low atmospheric pressure, anti-clockwise in the Northern Hemisphere and clockwise in the Southern Hemisphere (as viewed from above).
- A *tropical* cyclone is a generic term for a low-pressure system formed over tropical waters (25°N to 25°S) with thunderstorm activity at its centre. Tropical cyclones derive their energy from vertical temperature differences, and have warm cores.
- If it lacks a closed circulation, it is a *tropical disturbance*. If it has a closed circulation but under 39mph maximum sustained surface winds, it is a *tropical depression*. When winds exceed that threshold, it becomes a *tropical storm* and is given a name. Once winds exceed 74mph it is a *hurricane* (in the Atlantic or east Pacific oceans), or a *typhoon* (in the north-west Pacific).

Link to Code Repository

https://github.com/sambarba99/cyclone-trajectory-prediction

Test Set Trajectory Plots

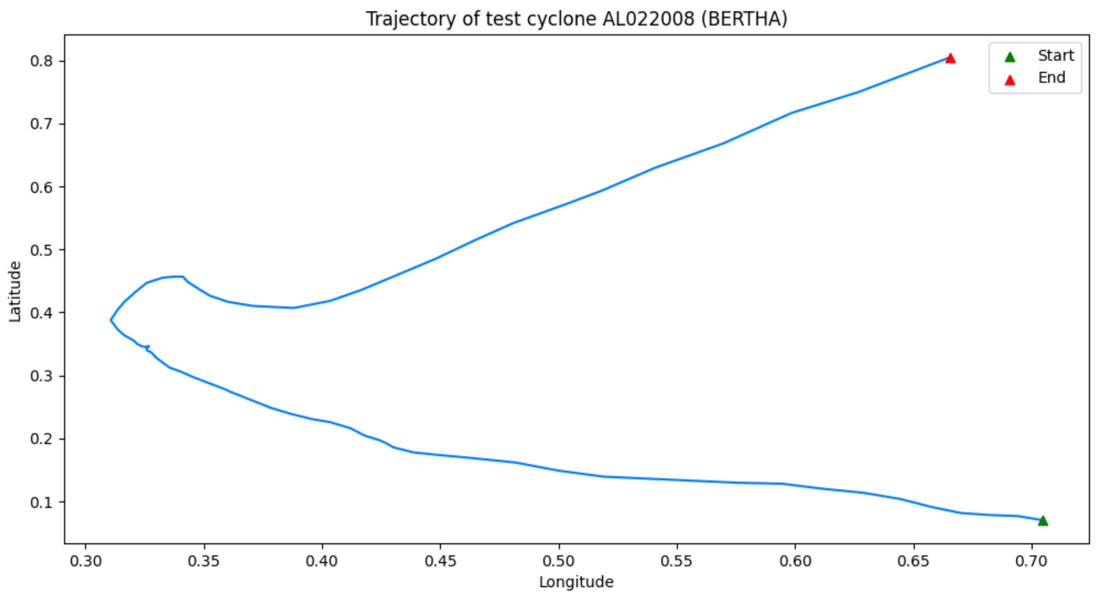


Figure 29: Trajectory of AL022008 (Bertha)

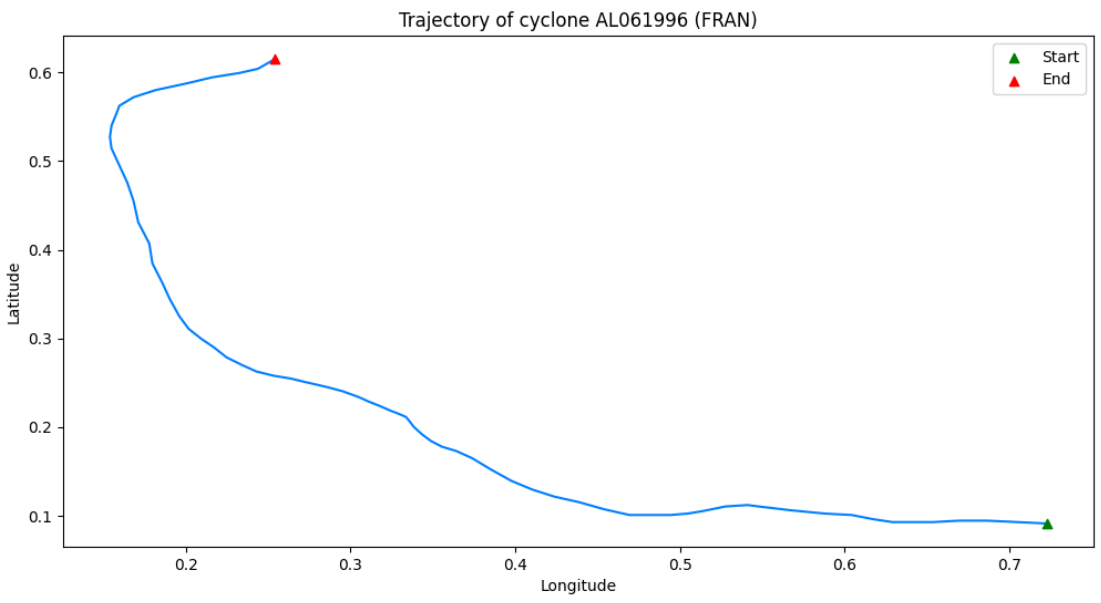


Figure 30: Trajectory of AL061996 (Fran)

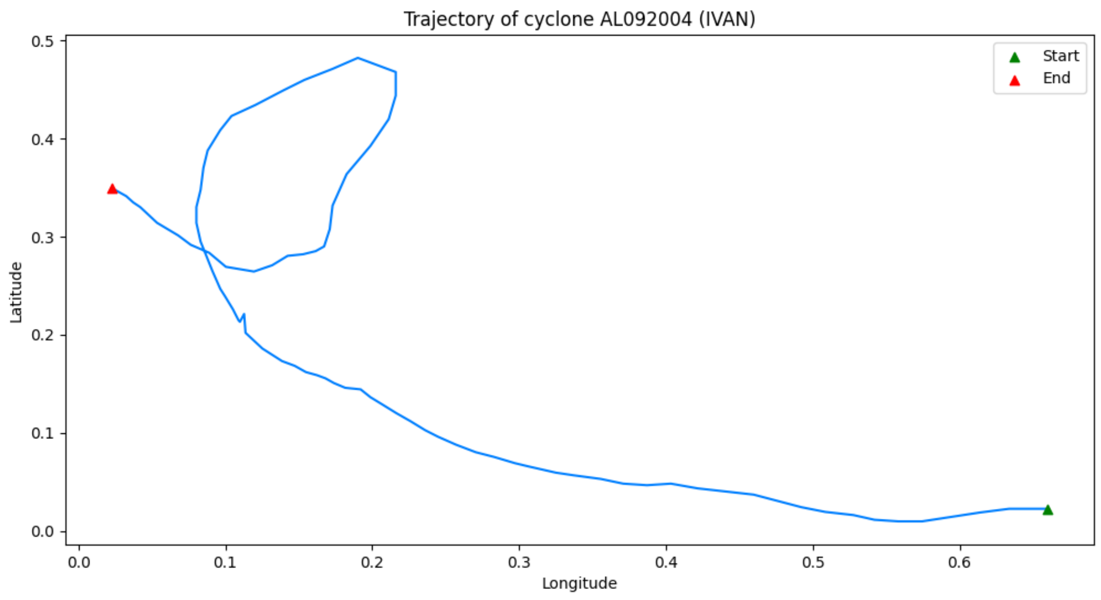


Figure 31: Trajectory of AL092004 (Ivan)

PACF Plots

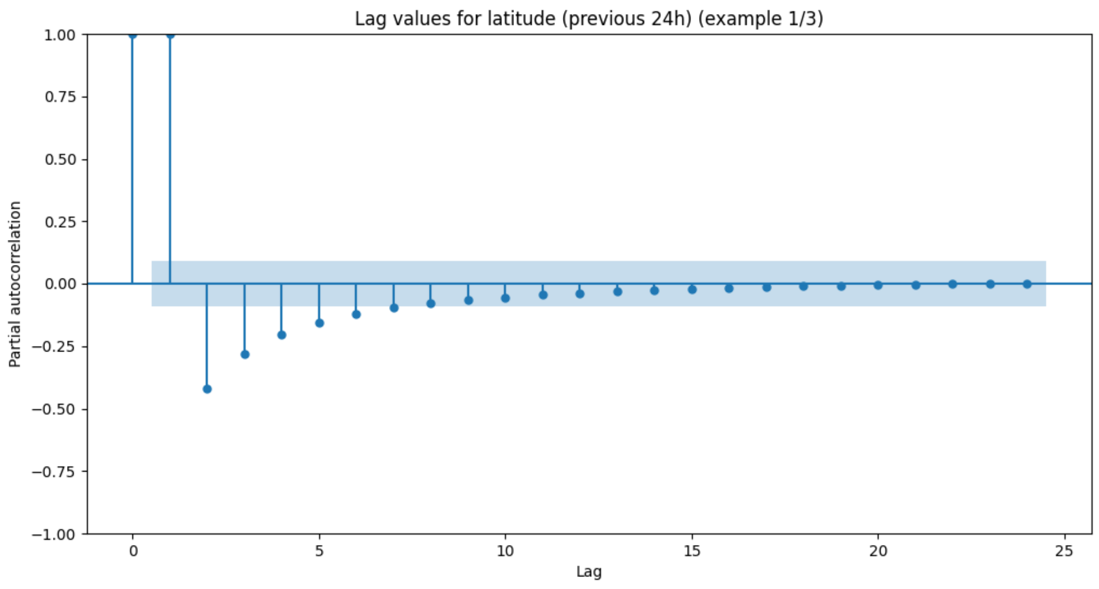


Figure 32: Lag values for latitude

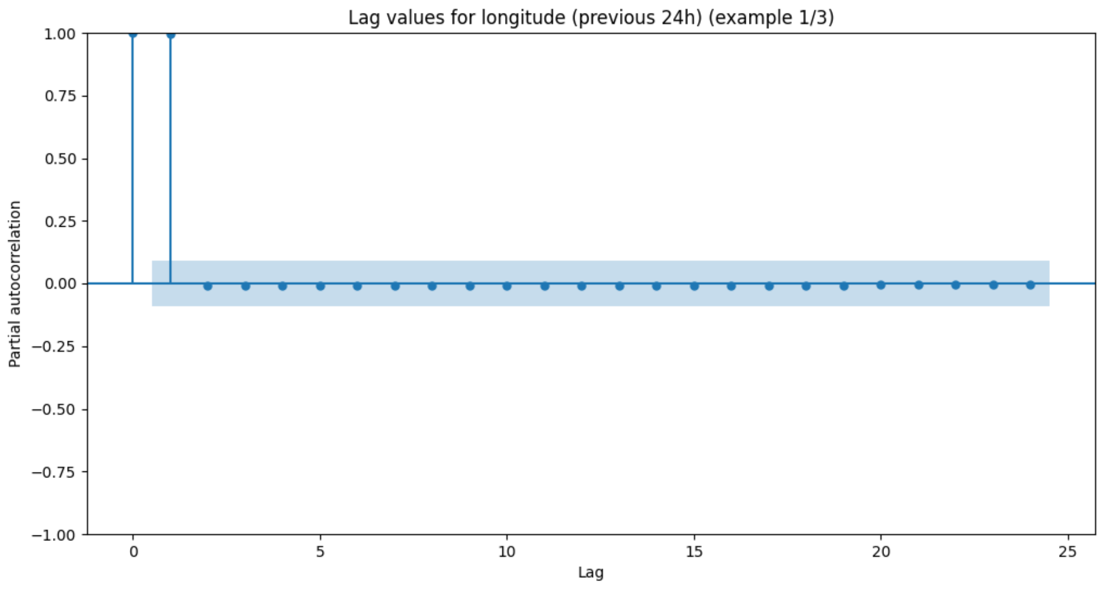


Figure 33: Lag values for longitude

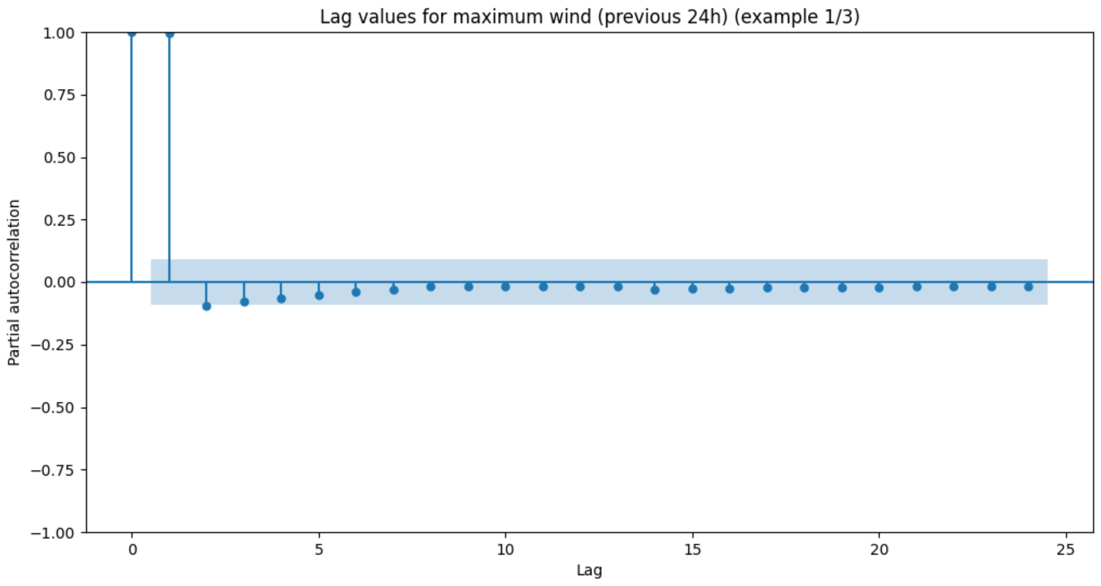


Figure 34: Lag values for maximum wind

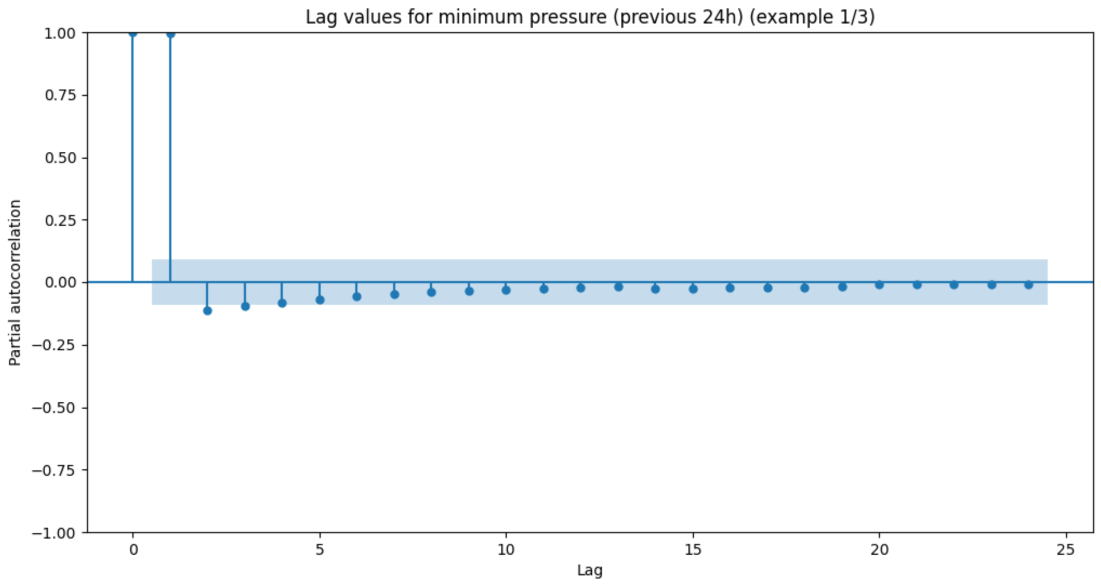


Figure 35: Lag values for minimum pressure

Iteration 1 Architecture

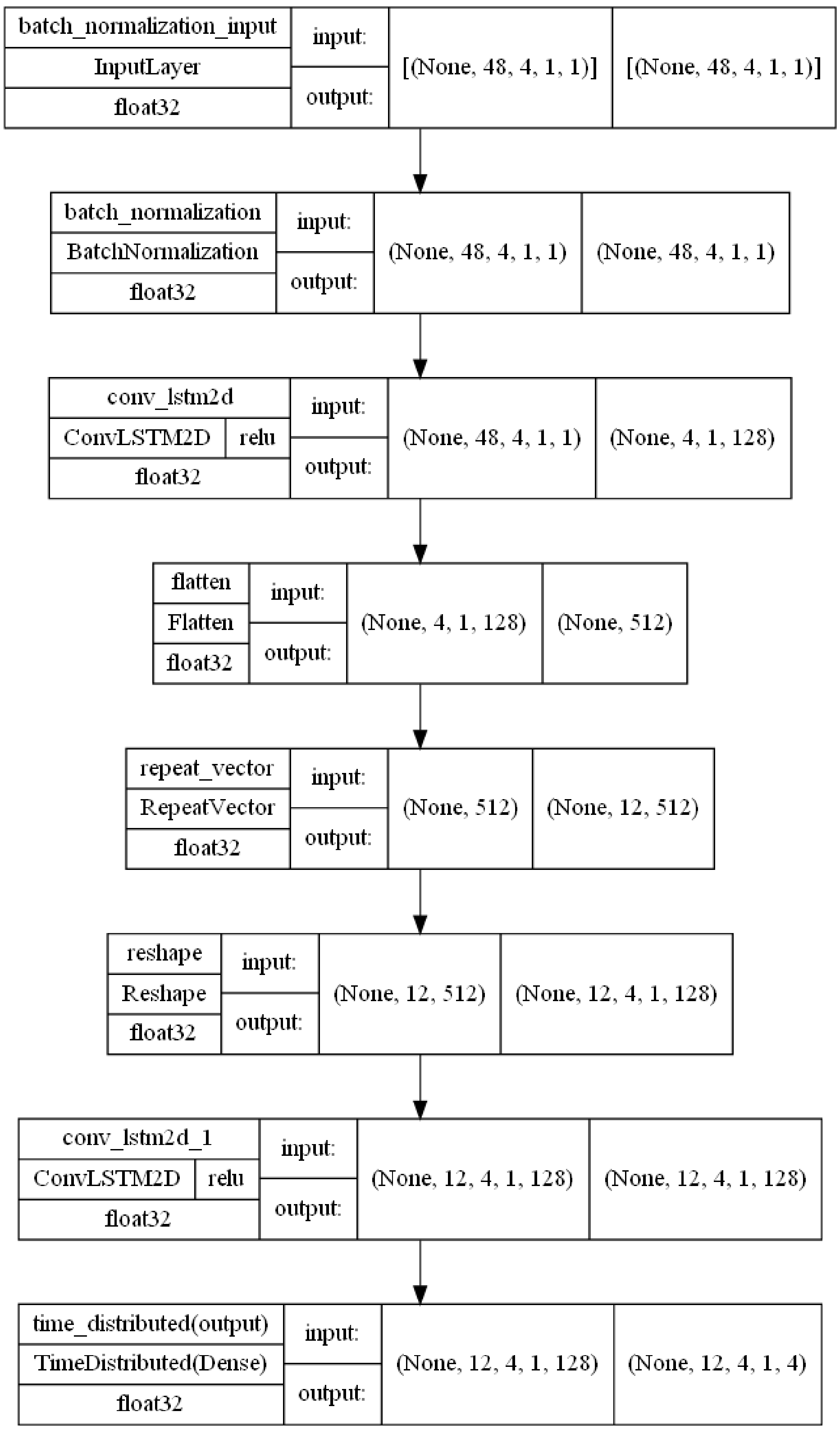
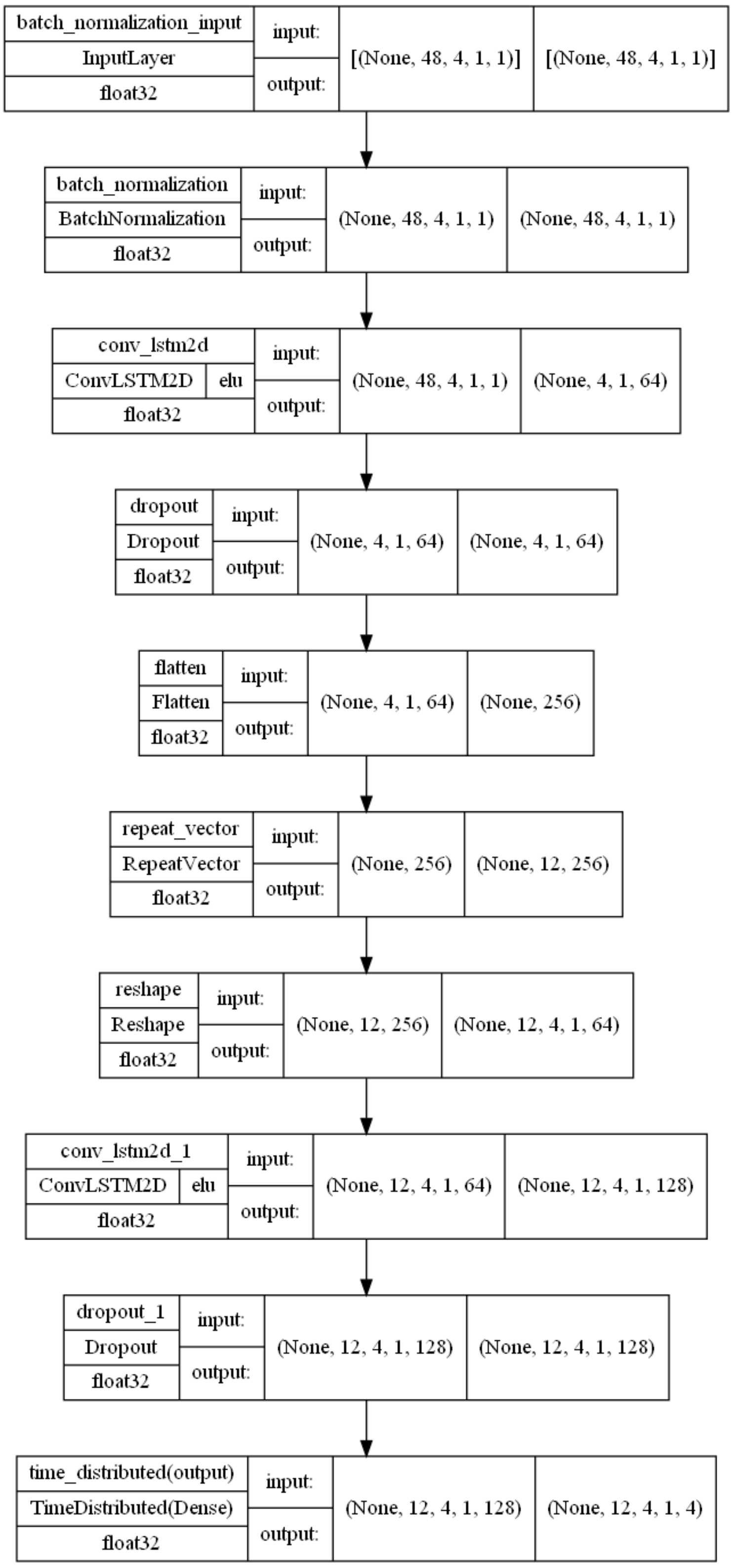


Figure 36: Iteration 1 (v1) architecture

Iteration 2 Architecture



batch_normalization_input | InputLayer | float32 | input: | output: | [(None, 48, 4, 1, 1)] | [(None, 48, 4, 1, 1)]
batch_normalization | BatchNormalization | float32 | input: | output: | (None, 48, 4, 1, 1) | (None, 48, 4, 1, 1)
conv_lstm2d | ConvLSTM2D | elu | float32 | input: | output: | (None, 48, 4, 1, 1) | (None, 4, 1, 64)
dropout | Dropout | float32 | input: | output: | (None, 4, 1, 64) | (None, 4, 1, 64)
flatten | Flatten | float32 | input: | output: | (None, 4, 1, 64) | (None, 256)
repeat_vector | RepeatVector | float32 | input: | output: | (None, 256) | (None, 12, 256)
reshape | Reshape | float32 | input: | output: | (None, 12, 256) | (None, 12, 4, 1, 64)
conv_lstm2d_1 | ConvLSTM2D | elu | float32 | input: | output: | (None, 12, 4, 1, 64) | (None, 12, 4, 1, 128)
dropout_1 | Dropout | float32 | input: | output: | (None, 12, 4, 1, 128) | (None, 12, 4, 1, 128)
time_distributed(output) | TimeDistributed(Dense) | float32 | input: | output: | (None, 12, 4, 1, 128) | (None, 12, 4, 1, 4)

Figure 37: Iteration 2 (v1) architecture

Iteration 2 Loss

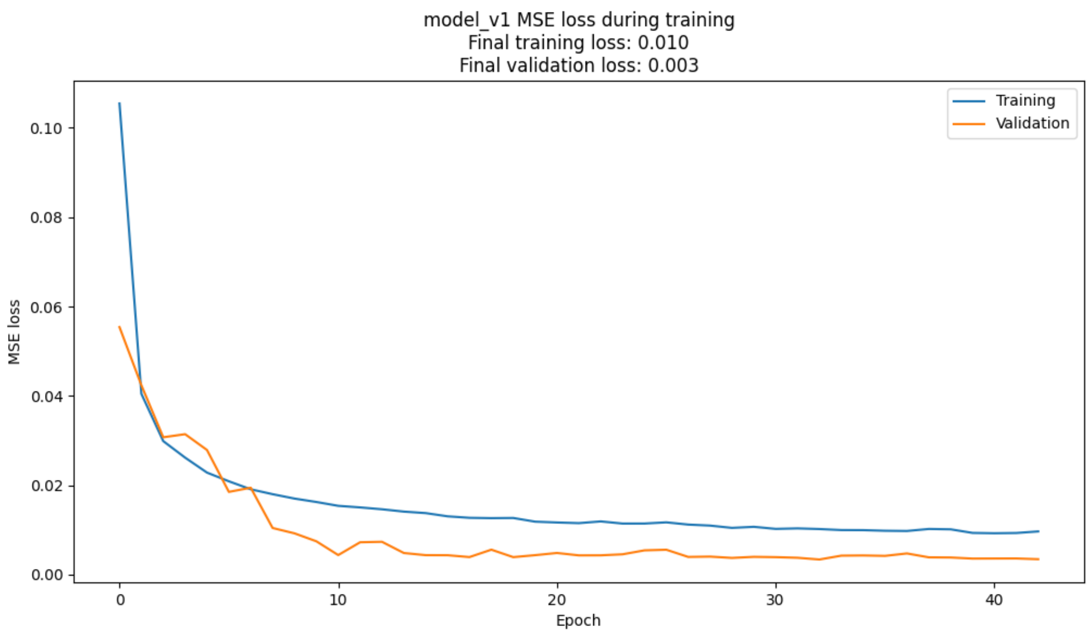


Figure 38: Iteration 2 (v1) loss

Iteration 2 Best Forecast (normalised)

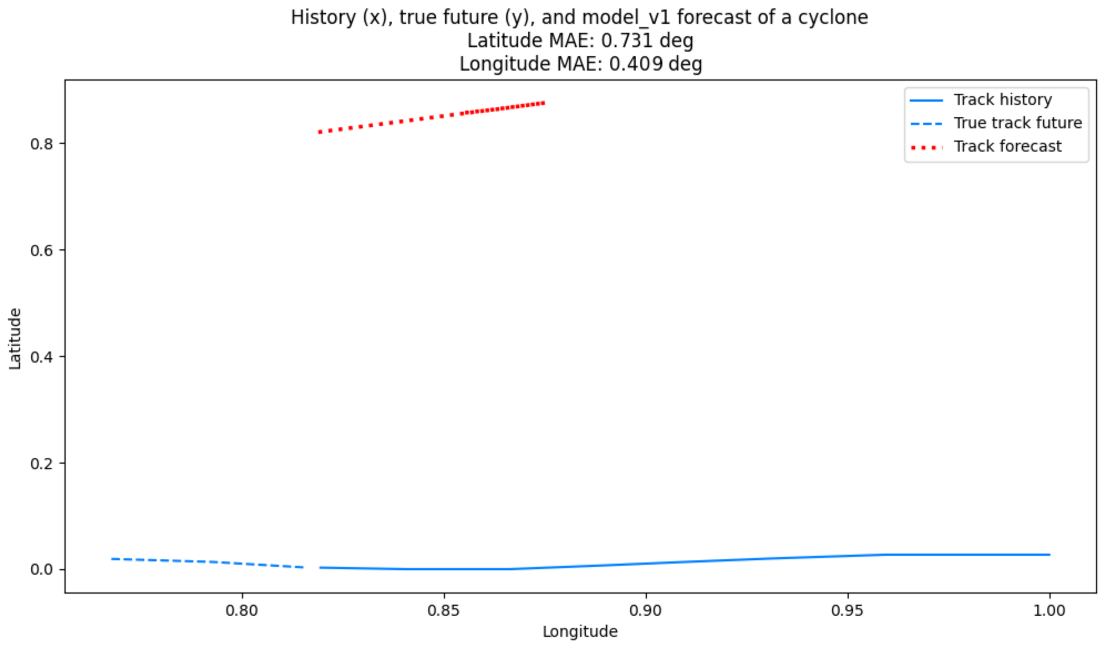


Figure 39: Iteration 2 (v1) best forecast

Iteration 3 Loss

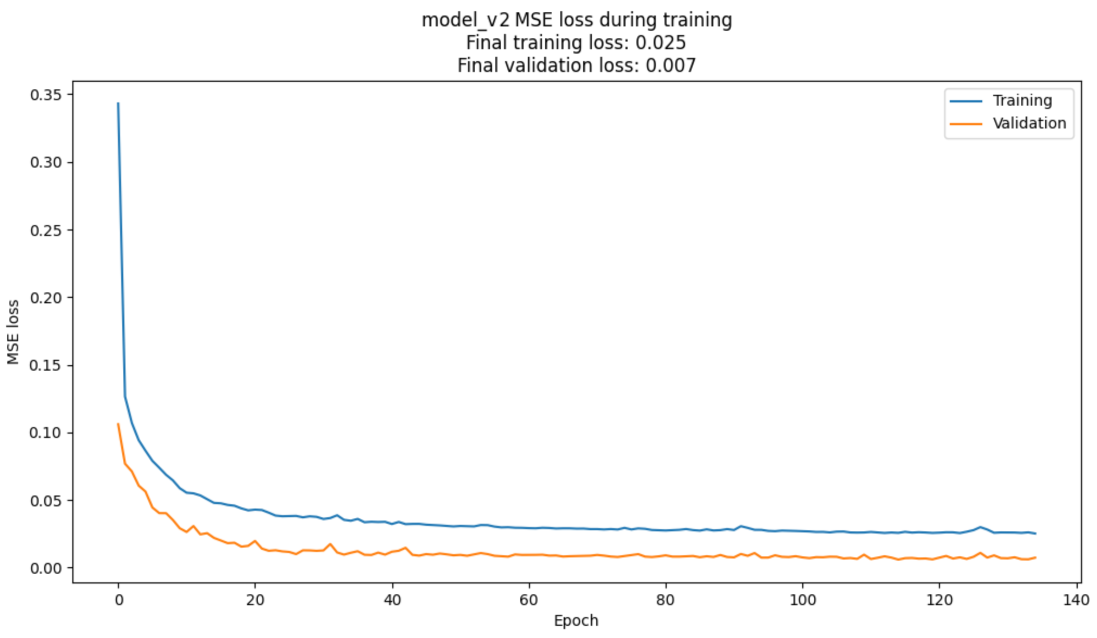


Figure 40: Iteration 3 (v2) loss

Iteration 4 Architecture

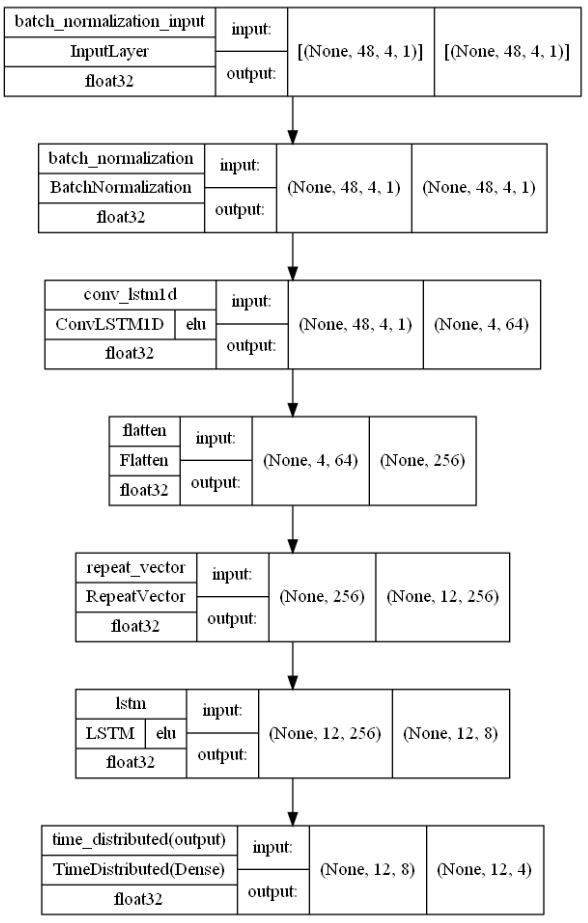


Figure 41: Iteration 4 (v2) architecture

Iteration 4 Loss

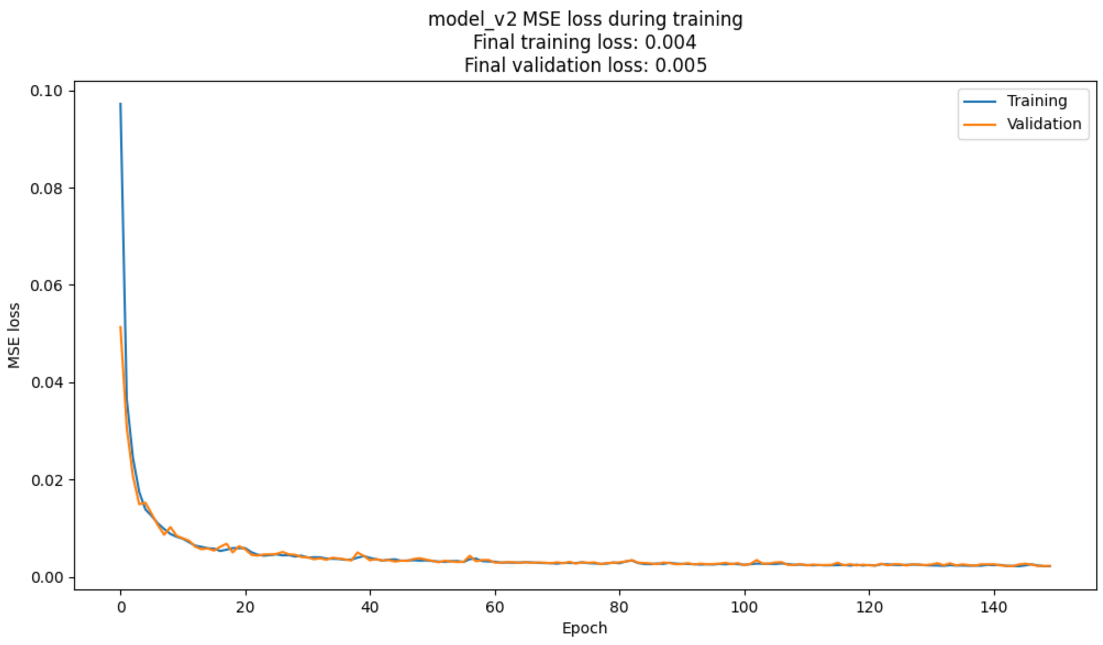


Figure 42: Iteration 4 (v2) loss

Iteration 4 Best Forecast (normalised)

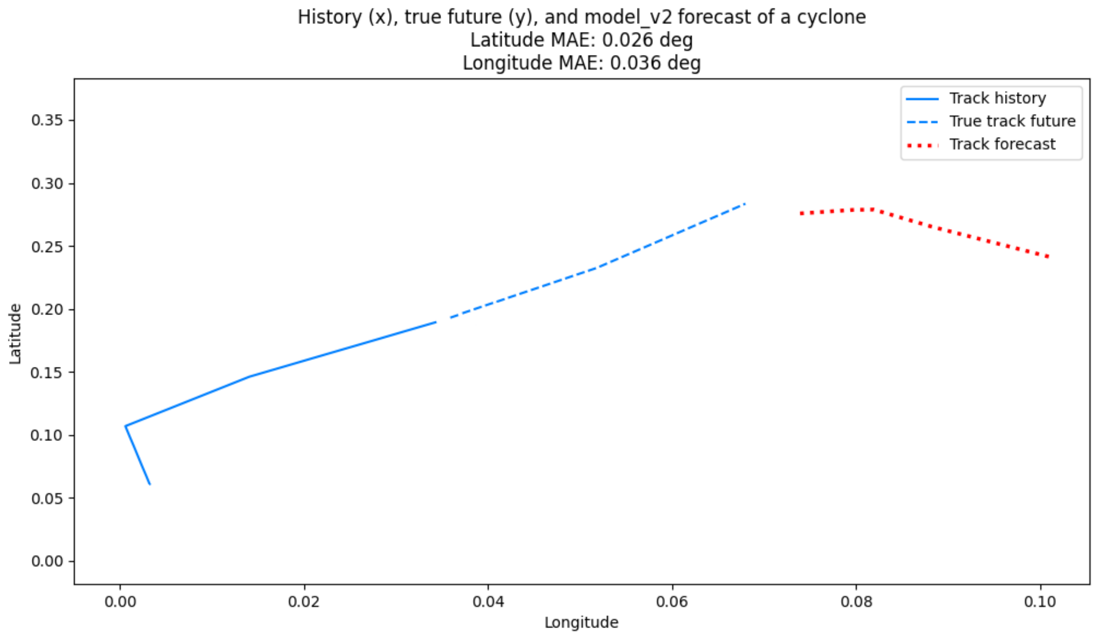


Figure 43: Iteration 4 (v2) best forecast

Iteration 5 Architecture

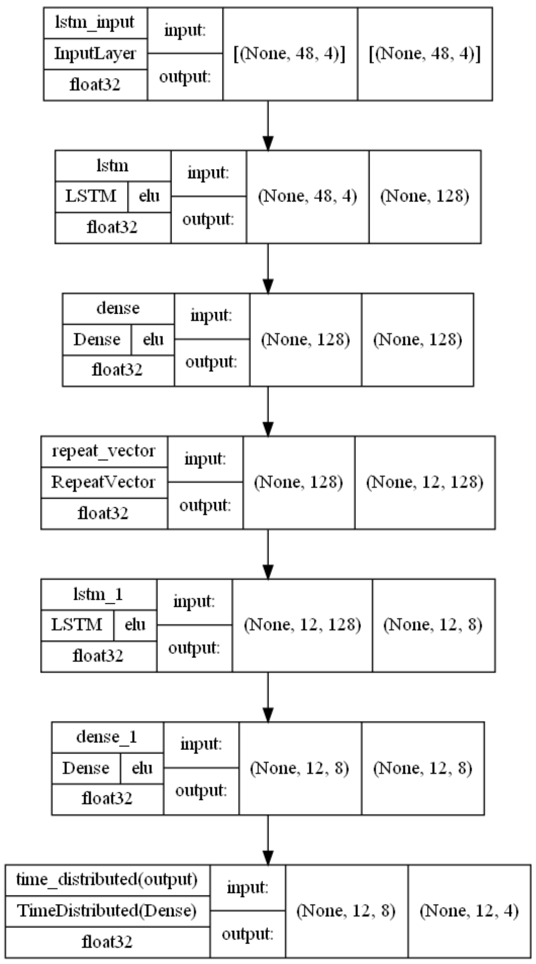

lstm_input | InputLayer | float32 | input: | output: | [(None, 48, 4)] | [(None, 48, 4)]
lstm | LSTM | elu | float32 | input: | output: | (None, 48, 4) | (None, 128)
dense | Dense | elu | float32 | input: | output: | (None, 128) | (None, 128)
repeat_vector | RepeatVector | float32 | input: | output: | (None, 128) | (None, 12, 128)
lstm_1 | LSTM | elu | float32 | input: | output: | (None, 12, 128) | (None, 12, 8)
dense_1 | Dense | elu | float32 | input: | output: | (None, 12, 8) | (None, 12, 8)
time_distributed(output) | TimeDistributed(Dense) | float32 | input: | output: | (None, 12, 8) | (None, 12, 4)

Figure 44: Iteration 5 (v3) architecture

Iteration 5 Loss

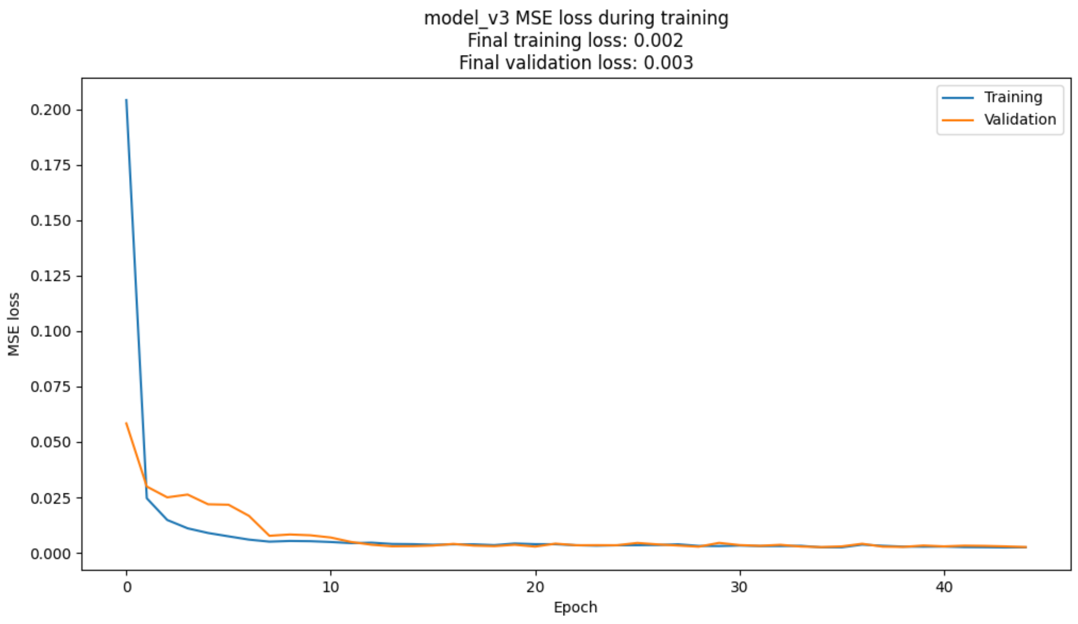


Figure 45: Iteration 5 (v3) loss

Iteration 5 Best Forecast (normalised)

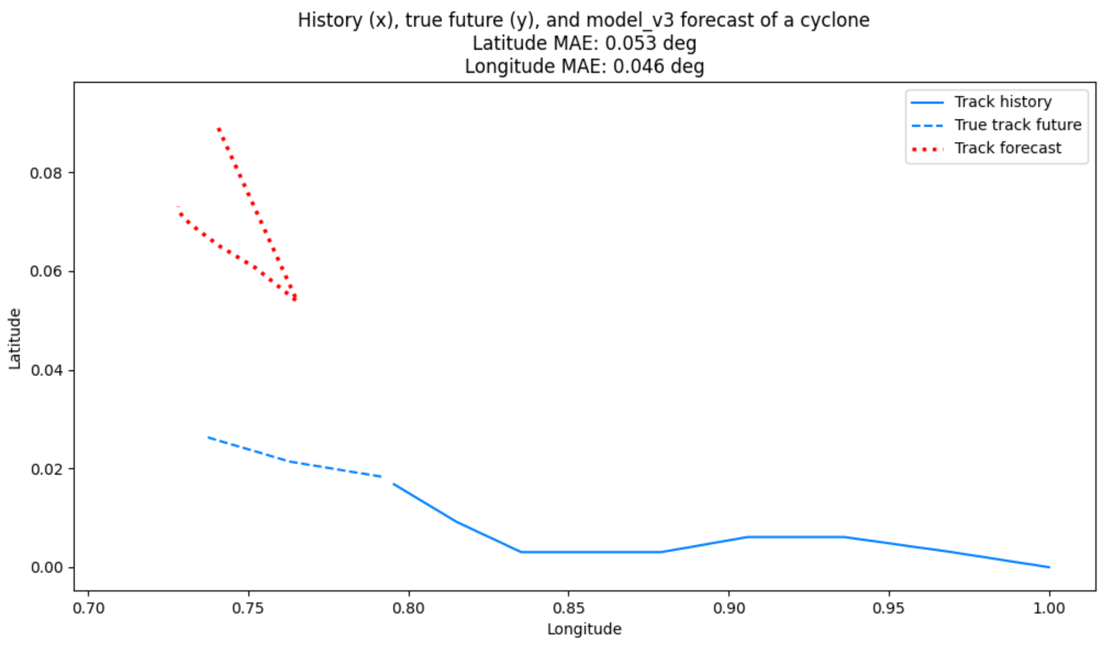


Figure 46: Iteration 5 (v3) best forecast

Iteration 6 Loss

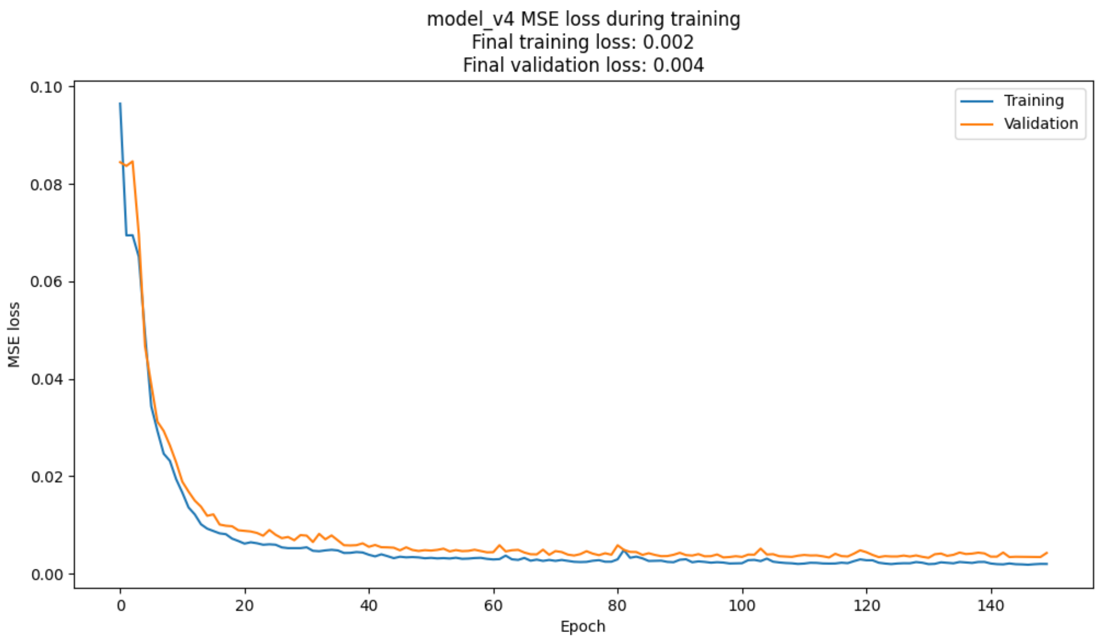


Figure 47: Iteration 6 (v4) loss

Iteration 6 Best Forecast

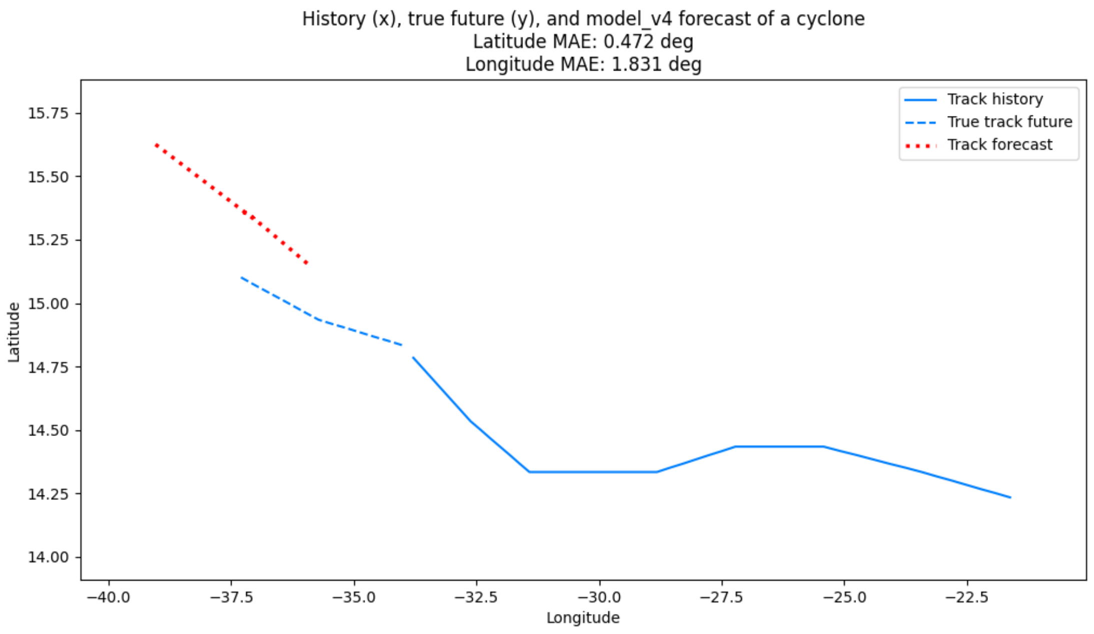


Figure 48: Iteration 6 (v4) best forecast

Iteration 7 Best Forecast

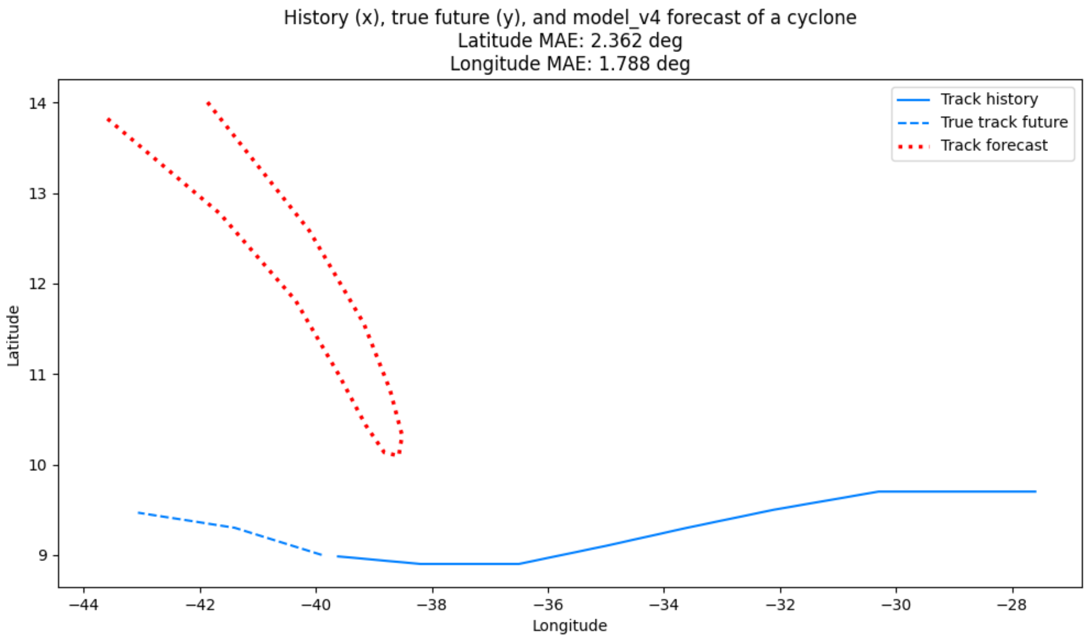


Figure 49: Iteration 7 (v4) best forecast

Iteration 8 Best Forecast (8x units)

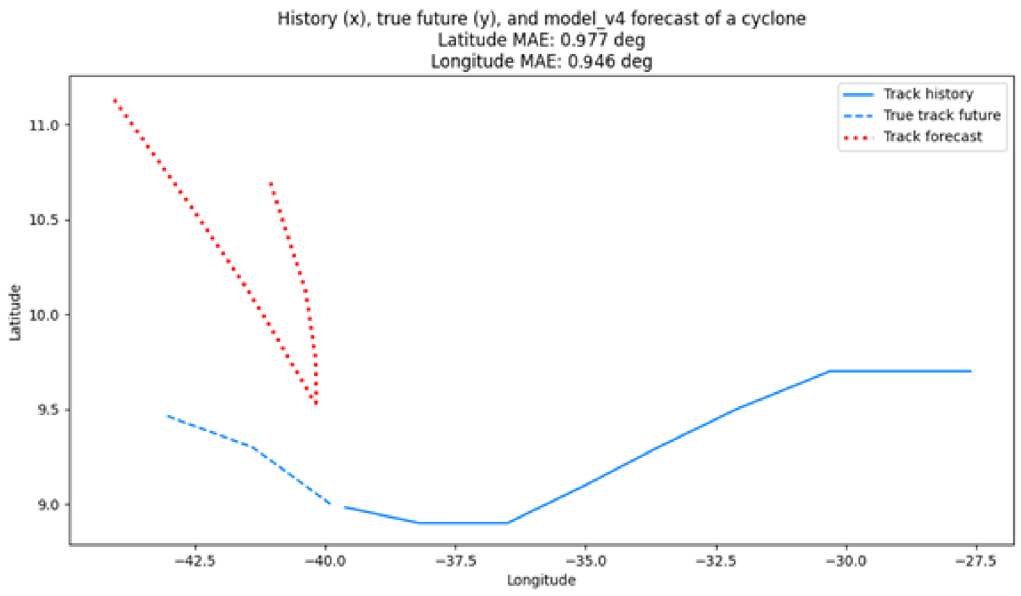


Figure 50: Iteration 8 (v4) (8x units) best forecast

Iteration 8 Best Forecast (2x units)

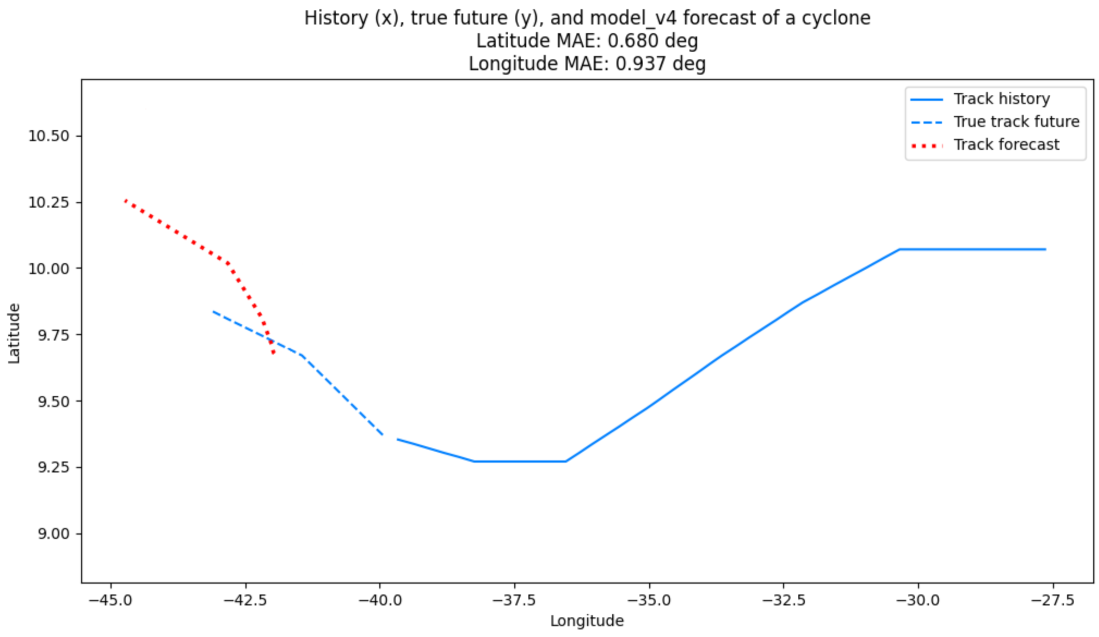


Figure 51: Iteration 8 (v4) (2x units) best forecast

Iteration 8 Architecture

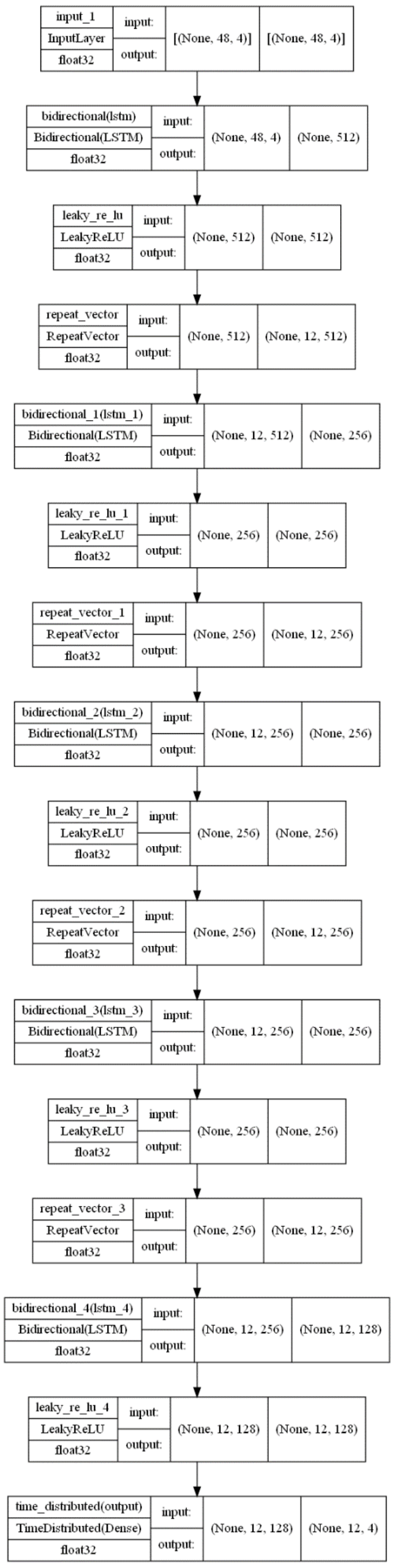


Figure 52: Iteration 8 (v4) architecture

Iteration 8 Loss

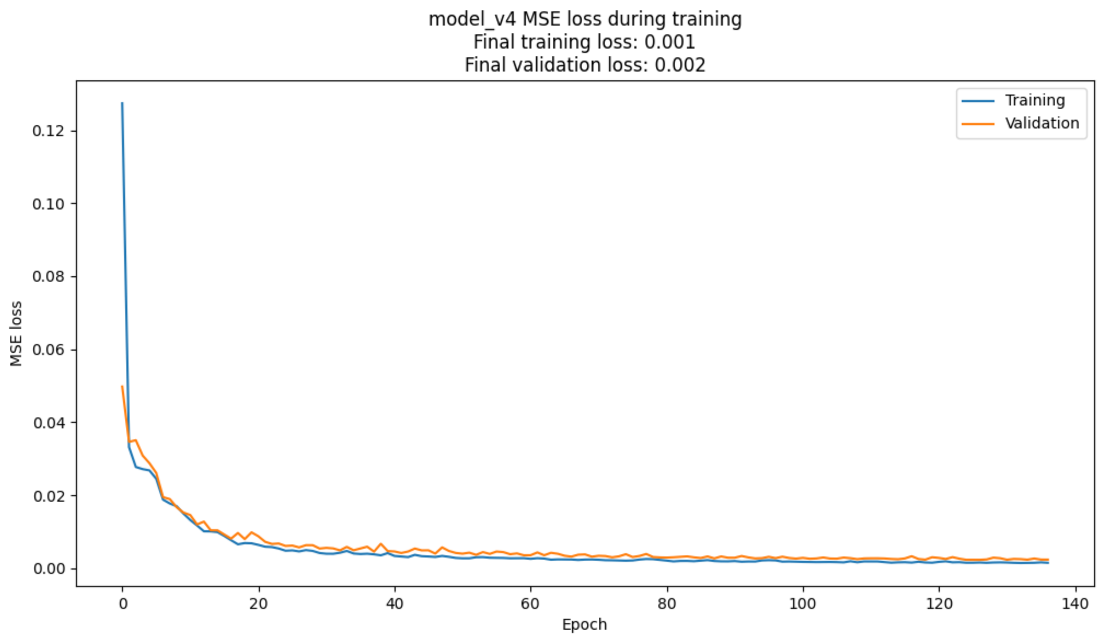


Figure 53: Iteration 8 (v4) loss