

CLIMATE RELATED NATURAL DISASTERS: A CRUCIAL CHALLENGE FOR PORT RESILIENCE. A
NEURAL NETWORK APPLICATION

By

NOMIKOU-LAZAROU EIRINI

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

Data Science

DEREE - The American College of Greece

2023

THESIS APPROVAL

“Climate related natural disasters: a crucial challenge for port resilience. a neural network application” a thesis prepared by Nomikou-Lazarou Eirini in partial fulfillment of the requirements for the Master of Science degree in Data Science was presented July 18, 2023, and was approved and accepted by the thesis advisor, internal examiner and the School of Graduate and Professional Education.

APPROVALS: _____

Dr. Dimitrios Milioris, Thesis Advisor

Dr. Dimitrios Vogiatzis, Committee Member

APPROVED BY: _____

Dr. Areti Krepapa

Dean, School of Graduate and Professional Education

1 Acknowledgements

I am so very grateful to my Professor and Supervisor Dr. Dimitrios Milioris for his invaluable patience, feedback and support, and for also having generously provided his knowledge and expertise. I would also like to express my deep gratitude to Professor Dr. Ioannis Christou who kindly offered his insightful advice on the subject. Last but not the least, I must thank my parents Sotiria Lazarou and Aris-Stefanos Nomikos, for their endless love and support.

2 Abstract

Current consumption habits are enabled due to the various commercial ports around the world. Goods are transported and traded only due to the existence of ports since the ancient days. However, any port disruptions jeopardize the ordinary consumption patterns. A well know suspect of port operations is climate change. Climate change shifts weather patterns causing more severe and more frequent weather events very often responsible for disturbance of port operations and marine roots. In this context, we investigate how Deep Learning Neural Networks (DLNN), in contrast to the traditional Numerical Weather Prediction (NWP) processes, could offer more accurate weather predictions in port regions preventing major economic losses. This Thesis presents the relative state-of-the-art literature on deep learning weather prediction and constructs 5 days forecasts for the ten biggest US commercial ports for 2023.

TABLE OF CONTENTS

1	Acknowledgements	1
2	Abstract	3
3	Introduction	6
4	Literature Review	8
4.1	Climate Change link to extreme weather events	8
4.2	Climate Change link to Port management and operation disruptions	11
4.3	Neural Network forecasting algorithms for time series	15
4.3.1	Artificial Neural Networks (ANNs)	16
4.3.2	DNN Deep Neural Networks	21
4.3.3	Simple RNN Recurrent Neural Networks	22
4.3.4	Problems with Recurrent Neural Networks	27
4.3.5	LSTM Long Short-Term Memory RNN's	29
4.3.6	Gated Recurrent Unit (GRU)	31
4.3.7	Transformer models with Attention mechanism	32
4.4	Neural Network models for weather prediction	34
5	Research Design and Methodology	37
5.1	Data Description	37
5.2	Data Cleaning	38
5.3	Model Application	43
5.4	Model evaluation	47
6	Conclusion	50
7	Appendix A	52
8	Appendix B	73

3 Introduction

Production intensification especially after the early 20th century, has been a critical factor for climate change. On this basis, production intensification increases exponentially greenhouse gas emissions (GHG) which by extension trap heat in the atmosphere of earth. Thus, global average temperature tends to increase more than the anticipated annual increase the last 100 years, causing changes to the climate that living beings were adapted to all those previous years. Indeed, climate in earth according to climatologists and paleoclimatologists, has been through many changes from desert climate to arctic climate, while beyond doubt climate constituting a space (geographical) related phenomenon. The disturbance factor that enters the usual climate earth cycle from warmer climate to colder climate and vice versa is that given the excessive GHG emissions, the anticipated climate change is accelerated leading to faster species extinctions and extreme weather patterns in only a few years, named as anthropogenic climate change. Focusing on the extreme weather events, many human economic activities already face severe disruptions, since the plan under which they were formed supposed a milder climate with less unexpected weather incidents. Port management and operations along with port supply chain, constitutes a human economic activity that is profoundly affected by extreme weather events. Given this background, a great volume of research effort continuously evolves towards the understanding of weather patterns and weather incident prediction to achieve better port operation adaptation and more efficient shipment scheduling, limiting extensive economic losses due to weather-related delays, and weather-related operational shutdowns in ports. Given this framework, the Thesis structure is as follows. Section 2.1, presents the concept of climate change, discussing anthropogenic and natural causes. It further investigates literature that links natural disasters and extreme weather events to shifting climate. Section 2.2, extends the intensification impact of extreme weather events due to changing climate, and explores their effect on ports. Moreover, this section inspects within literature the economic impact of port damages and disruptions caused by extreme weather events and possible adaptation measures that could increase port resilience to the reality of changing climate. Section 2.3, presents at first the notion of neural networks zooming into the most crucial architectures and

describing their functionalities while Section 2.4 presents the state of the art literature for neural network algorithms used in weather prediction applications exhibiting the evolution of weather forecasting field interest from traditional Numerical Weather Prediction procedures to the currently used Deep Learning Neural Networks. Following, in regards to the empirical work done on this Thesis, the aim of this study is to develop a neural network algorithm that is able to predict future weather patterns around port regions affecting port operations and causing economic damages. In these lines, Section 3.1 portrays the data used for this analysis that are comprised of event type incidents and their respective duration in various American States. Next, Section 3.2 analyzes in detail the data preprocessing steps taken and the made assumptions while Section 3.3 describes investigates various LSTM architectures and presents the results of the best performing models per state and per season. Finally, Section 3.4 evaluates the model performance over the tests sets and based on their performance the best models are used to create forecasts for a time span of five days beyond the test set. General results, implications, and future work is presented in Section 4. Appendix A and Appendix include code for a complete step evaluation of one port in a four season period, along with tables with all model performance metrics, tables and graphs with prediction accuracy over the test set and forecasts for days beyond the test set for the best models.

4 Literature Review

4.1 Climate Change link to extreme weather events

Evidence indicate that indeed increasing in frequency and intensity weather related events disrupt the normal operational activities of ports around the globe resulting to increasing economic damages. However, it is meaningful to briefly introduce evidence in literature that identify links between climate change and weather events. In this framework, the next paragraphs introduce basic concepts such as the differentiation between climate and weather events, the notion of climate change and it's possible causes, the consequences of climate change on our surrounding environment, as well as the concept of "extreme" natural events. Beyond doubt, the extreme weather incidents announced the last years by the various natural catastrophe observatories around the world, establish a relation regarding climate change, a relation that has become an urgent and political topic ([Khurana et al., 2022](#),).

This discussion, has been a controversial topic as for it's causes e.g. the percentage of the intensification of natural catastrophes resulted by human behavior or/and resulted by a natural cycle more difficult for scientists to observe it due to human's limited life span. Climate pattern differentiations in comparison to prior decades, are caused by two usually opposite but sometimes supplementary forces. On one hand, are the internal forces to the climate system regarding the natural climate cycles. On the other hand, are the external forces for example, changes in the sun's radiation due to the length of sun cycle, volcanic activity, but also anthropogenic forces resulting from excessive emissions of greenhouse gases when burning fossil fuels, oil, land use etc. Regarding greenhouse gasses, the mechanism behind global warming is that gasses released by human activity, in the atmosphere trap the heat existing in the atmosphere that is responsible for preventing radiation from escaping into space. If the anthropogenic greenhouse gasses didn't exist, it is considered that radiation would escape space, avoiding the earth's overheating and balance earths climate to a situation where fauna and flora have used to live and survive on this planet. In addition, greenhouse gasses display certain persistence,

in the sense of remaining in the earth's atmosphere for multiple decades, denoting that climate change is bound to continue for the years to come even if we stopped emitting today. Shifts in temperature, promote a variety of secondary consequences on hydrological systems, terrestrial and marine ecosystems (Van Aalst, 2006,). Such influence, involves global mean sea level rise, extensive retreat of glaciers, reduction of lands covered in snow, melting of permafrost areas, distortion of plant and animal ranges, early flower bloom, bird breeding seasons and development of insect population, coral bleaching, extreme weather events etc.

Nevertheless, apart from the frequently discussed adverse consequences of climate change there are also some positive outcomes. It is true that since climate change effects are highly correlated with spatial longitude and latitude, in some regions, increase in average temperature will improve and enhance agriculture as well as reducing the winter heating energy needs. It is a tragic irony that regions which participated the minimum to greenhouse gasses concentration, will experience the maximum ramifications.

On this framework, natural scientists state that external forces affecting the global average climate conditions, lead to changes in extremes (Seneviratne et al., 2012,). In this stage, is meaningful to define climate extremes. And this is crucial because "extreme" is a feature dependent on space and time, it is not an absolute property of an event. Accordingly, "climate extremes" are typically contingent on the probability of occurrence of a specific number of events or on the surpassing of a prior set threshold. Regarding the assumption of threshold, different thresholds are applied, however, conventionally probabilities of occurrence 10%, 5%, 1% or less across a specific time interval are used to describe the probability of extreme natural events.

It is useful to group the extreme climate incidents in relation to natural disasters, into the following three categories, firstly we could observe extremes in climate variables such as temperature, precipitation and wind, secondly detect weather and climate events which impact on the occurrence of extremes in weather or climate variables such as monsoons, tropical cyclones, heavy storms e.t.c. and thirdly, monitor consequences on the natural environment regarding droughts, floods, extreme sea level, waves, landslides sand storms, dust storms and so forth.

Of course, the researcher may examine a specific natural catastrophic event aiming to understand its economics and social impacts, however nature has taught us that the only adjective that we cannot name her is simple and linear. Having this in mind a very useful literature has been developed around the idea of compound events (Brink et al., 2005,) and (Svensson and Jones, 2002,). In particular, the idea behind compound events is that several mild weather or natural events aggregated, and essentially their combination is the extreme event.

An other meaningful remark around extreme natural incidents, is the distinction between

extreme weather incidents and extreme climate incidents. The first concept is referred to fluctuating weather patterns, observed within small time frames e.g. lasting from some hours up to several weeks, in comparison to the second concept referring to longer time scales presenting more extreme natural events with higher persistence in time.

Usually, in scientific literature, extreme indices regard "moderate extremes", in contrast exorbitant extremes are examined by employing the Extreme Value Theorem, a method aiming to derive the probability distribution of rare incidents based on the right and left tails of the general probability distribution, mainly below 1-5 % of the total sample (Coles et al., 2001,).

In order to uncover the extremely significant link between unusual weather patterns and average condition trends researchers utilize empirical analysis comparing present and historical data, theoretical analysis applying simulations in global and regional climate models (GCMs, RCMs), along with detecting patterns in insurance claims driven by catastrophes from extreme natural incidents (Pielke, 2005,) and (Pielke Jr et al., 2005,).

Specifically, middle and high northern latitudes (Tropic of Cancer, Tropic of Capricorn and in the Arctic and Antarctic Circle) are anticipated to display increased extreme precipitations and by extension an shorter return period for intense rainfalls leading to increased floods and landslides. Middle latitude zones had typically four seasons (regarding the 19th, 20th, 21th, centuries) but new data identifying the real effects of climate change, show evidence of season deviation phenomena. Low latitude regions, close to the equator, are anticipated on the contrary to have long dry periods, which will give rise to significant risk of extended droughts and fires, especially during the summer season.

An other determinant factor that could play an important role on climate change is the North Atlantic Oscillation. According to (Anderson and Bausch, 2006,), when it is at it's negative phase it may trigger dryer winter weather conditions resulting to less recharging of rivers, and by extension to more intense summer droughts. Analysis studying the Atlantic Oscillation decade per decade, since 1850, show that the increased frequency and intensity of Atlantic tropical cyclones over the past few years, might be a result of a combination of the natural cycle of Atlantic Oscillations and climate change. However from 1995 and the later years, it is identified that climate change has higher impact on the observed increased catastrophes.

In fact, climate models evidence that losses from winter storms, are anticipated to double until 2085 forcing some European regions to suffer from the impacts of climate change (Höppe and Grimm, 2008,). It can be concluded that climate change shifts acknowledged hazard risks, but on the same time just as well increases the degree of uncertainty of intensity, frequency and new phenomena of natural catastrophic disasters.

Having stated the basic concepts of climate change and its relation to natural disasters and extreme weather phenomena, we have set the ground for analyzing more the consequences caused in human activity. Specifically, for the scope of this study, we will try to briefly present evidence in literature indicating port disruptions due to unanticipated weather events such as intense storm events, fog and heavy wind.

4.2 Climate Change link to Port management and operation disruptions

Although climate change is a phenomenon widely claimed to be part of a bigger climate cycle of earth's interchangeable climatic conditions within the thousands of years, increased anthropogenic emissions of CO_2 accelerate its process and evolution. Climate change, is a continuously evolving condition that among other impacts such as disruption of biodiversity and development of diseases in all species, increases the frequency of extreme weather events and triggers the intensification per incident including sea rise level, tropical storms and typhoons, shift of the absolute average ocean temperature and extreme storm incidents. Furthermore, heavy rain exceeding quay well drainage capacity causing flooding, rising ocean temperature causing the deterioration of water quality within harbors and augmentation of wave height surmounting breakwater design levels (Yang and Ge, 2020,) have a significant impact upon man made infrastructures such as coastal settlements and ports. Focusing specifically on the last type of structured space, ports play a key role in the international transportation of goods. While land and air transportation are crucial transportation alternatives, marine transportation has enabled the current consumption patterns, allowing for millions of containers filled with commercial goods to be exchanged among continents in relatively short time while being also economically more efficient than the other means of transport. Observing the image in a global scale, it is estimated that 80% to 90%¹ of the total magnitude of traded goods are transported via the sea, while the estimations regarding marine trade anticipate a 4% rate of change for the next three years (León-Mateos et al., 2021,). Especially in the case of the United States (US), the marine transportation ecosystem offers more than 23 million job positions, and maintains more than 99% of the international marine commerce, while according to the American Association of Port Authorities, the indirect and direct impact of ports is estimated to create more than \$4.5 trillions every year (Wendler-Bosco and Nicholson, 2020,). Certainly, in the US are located some of the biggest commercial ports on earth e.g. the Port of New York and New Jersey in New York, the Port of Los Angeles in California, the Port of Houston in Texas etc. However mostly after 2005, is observed a growing literature of studies that investigate the US port resilience to climate change related extreme weather events (Panahi et al., 2020,), and highlight the intensification and expanding duration of weather events having a great

¹<https://unctad.org/publication/review-maritime-transport-2021>

negative impact among other social and economic factors, in ports as well. In particular, the Superstorm Sandy in 2012, was registered as one of the most severe natural disasters that have hit the US, leading to the loss of more than 200 human lives and caused close to \$70 billion in infrastructure and economic damages ². The aforementioned superstorm, among the other damages, disrupted the operation of the Port of New York and New Jersey for more than 8 days resulting in huge economic losses (Smythe, 2013,).

Explicitly, according to United Nations Conference on Trade and Development some of the most important potential climate change impacts on ports³ in relation to extreme weather conditions (e.g. hurricanes, storms, floods, increased precipitation and wind) include:

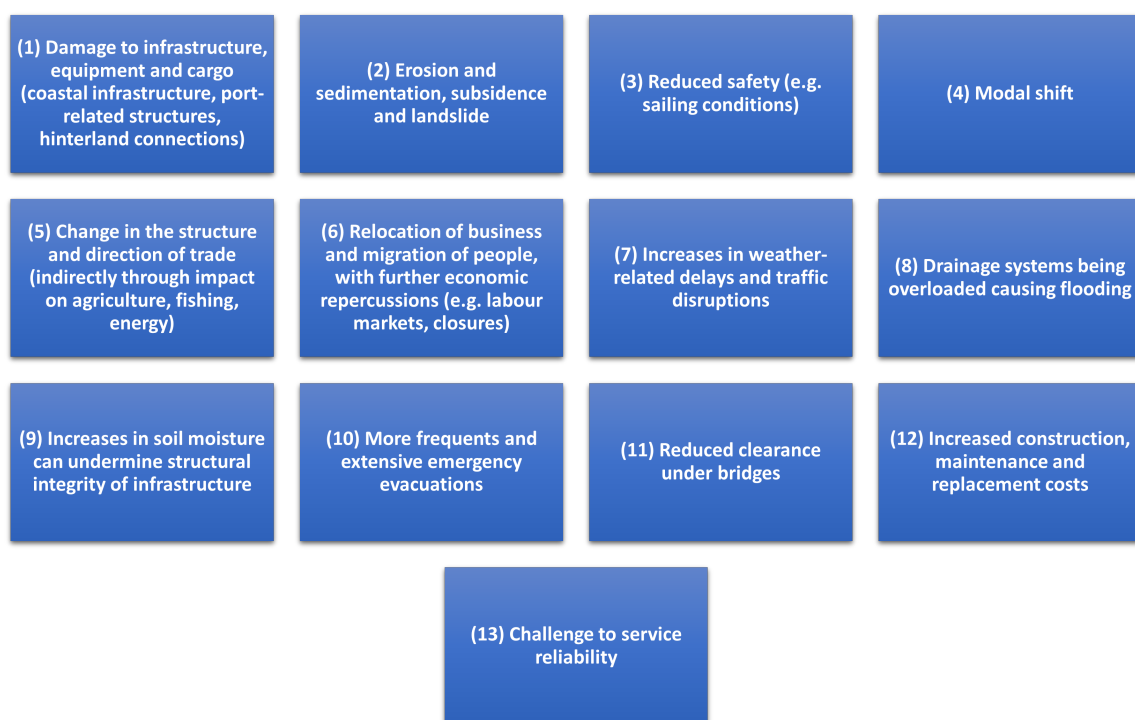


Figure 1: Impacts of extreme weather conditions on ports based on UNCTAD

Assessing the additional impacts of climate change impacts on ports regarding rising of sea levels in the long-run (Lenton et al., 2009,) indicate that if by 2050 sea level has risen by half a meter, the value of exposed assets in 136 port mega-cities will reach a magnitude of \$28 trillions. Following, extreme weather and climate conditions have a strong impact on the overall operation activities of ports, disturbing the normal operation programmed schedules. Specifically, it is shown that, intense storms and shift in sea level rise apart from flooding in various regions of the port often results in increased downtime disturbing the supply systems

²https://www.nhc.noaa.gov/data/tcr/AL182012_sandy.pdf

³https://unctad.org/system/files/official-document/dtltlb2011d2_en.pdf

(Esteban et al., 2014,). Elaborating, regarding the global supply chain, extreme weather events consist crucial risk factors causing disruptions, delays, systems risk, forecast risk, intellectual property risk, procurement risk, receivables risk, inventory risk and capacity risk (Hofmann, 2013,). Additionally, the occurrence of an extreme weather event may cause electric power shut downs for extended periods of time and in combination with the fact that human behavior may become unpredictable after a disaster, could result to an unclear or mistaken flow of information limiting effective communication much needed in such conditions (Wendler-Bosco and Nicholson, 2020,). Observing this complex problem, (Izagirre et al., 2021,) estimate the future multi-hazard risk in global port operations by defining the risk function for ports is a function of hazard, exposure and vulnerability and introduce changes only in the hazard component to observe the risk of ports in 2100 under the assumption that no significant emission reductions have taken place.

Moreover, (Rose and Wei, 2013,) identifies disruptions in normal port operations have a compound impact. This compound impact is elaborated in three main levels, the first accounts for the port level that bears disruptions of imports and exports along with shift in port activities, the second monitor the macroeconomic level which includes intermediate good shortfalls, final goods shortfalls and final demand decrease, while the third level accounts for the total impacts referring to the effects in the surrounding region of the port and the permanent loss of port business.

Apart from the port infrastructure losses and their sub-consequences in the whole supply chain and economic stability, extreme weather events have a great impact on the vessels themselves (Wendler-Bosco and Nicholson, 2020,) while is also observed a decrease in the container throughput (Cao and Lam, 2019,). In combination to the material losses, reputation losses regarding to the effectiveness of the port could lead to less demand for the port that has not taken the required adaptation measures (Cao and Lam, 2019,), (Wendler-Bosco and Nicholson, 2020,), (Athanasatos et al., 2014,). In this context, it is meaningful to state that in general identifying the climate related risks for a specific port is a very complex endeavor, according to (Gharehgozli et al., 2017,) consists a "wicked problem", and this is the reason why investment by ports in disaster prevention is still limited in comparison to the real world needs. In these lines, the measures that increase the port's resilience (Cao and Lam, 2019,) in climate change related events account for improvement of port structures, improvement in port facilities and other non-structural measures displayed in more detail in the following table.

(Alderson et al., 2015,) focusing in the notion or port resilience, highlight the concept of operational resilience, which is a term describing the ability of a system to absorb the impact of an event/ shock "without losing its operational capacity". Extending this concept, (León-Mateos et al., 2021,), create the Port Resilience Index (PRI) an indicator that measures the capacity of a port to absorb and recover from the damages of an natural disaster event offering a quantification of the total port resilience, the view of improving port's adaptability and the

The development and improvement of port structures	Improvement of port facilities	Non-structural measures
Changing or reinforcing structures such as revetments, wave-breakers, and rails, etc.	Raising wharf and breakwater height to cope with high waves	Improvement of cargo handling facilities to reduce material or equipment losses caused by strong winds or waves
Strengthening port facilities and equipment, such as shore cranes, to cope with strong winds	Steepening quay slopes to improve drainage capability	Promotion of inland water cycling in and out of the breakwater to reduce deterioration of water quality
Replacement or securing pipes or manhole covers to mitigate increased groundwater levels caused by buoyancy uplift and install groundwater drainage pumps		Harbor deepening and waterway dredging
Strengthening existing pier design criteria in response to the impacts of climate change		Ensuring safety and incorporation of alternative transport routes and logistics solutions

Figure 2: Adaptation measures that increase port resilience against climate change

current performance of the port in terms of resilience against climate change related events.

Specifically, UNCTAD indicatively identifies a variety of measures to limit the adverse effects of extreme storm events, presented in the table below ⁴.

Potential adaptation measures for port limiting the effects of extreme weather events
(1) Integration of emergency evacuation procedures into operations
(2) Setting up of barriers and protection structures
(3) Relocation of infrastructure
(4) Ensuring functioning of alternatives routes
(5) Greater monitoring of infrastructure conditions
(6) Restriction of development and settlement in lowlying areas
(7) Construction of slope-retention structures
(8) Preparation for service delays or cancellations
(9) Adjustments to speed and frequency of service
(10) Strengthening of foundations, raising dock and wharf levels
(11) Smart technologies for abnormal events detection
(12) New design for sturdier ships
(13) Development of new design standards for hydraulic structures such as drainage channels
(14) Better land use planning in flood prone areas
(15) Construction of storm retention basins for flush flooding

Figure 3: Adaptation measures that increase port resilience against extreme storm events

Summarizing, anthropogenic climate change has very strong impacts on the "as we know it" natural and artificial system. For this reason, mitigation measures are taken from many parts of the society such as governments, companies, NGO's etc. aiming to reduce a further disturbance of climate. In parallel, the development of an extended toolbox of adaptation measures that

⁴https://www.nhc.noaa.gov/data/tcr/AL182012_sandy.pdf

recognize that even in the case of net zero emissions in the close future, existing emissions in the atmosphere are already causing and will cause more shifts in weather and climate patterns is of utmost importance. Adaptation measures are also taken from various agents of the society and economy, such as governments forming national adaptation plans⁵ and individuals who wish to protect their properties from extreme weather events e.g. housing, companies and factories infrastructures, farming landscapes etc. Among the other domains that are affected heavily from climate change related events are commercial ports. Ports, from the ancient times consisted the structural nodes around and due to which entire civilizations were developed and thrived. Today, given that commercial port infrastructures are hubs that facilitate all international trade functions, literature sheds light in the fact that in reality they are highly vulnerable to climate change incidents, and the absence of appropriate adaptation measures could jeopardize the global supply chain and established social stability in general in the case of operation disruptions. Measures that investigate port disruption management and resilience in general is a relatively new field of research however the inherent uncertainty of accurately predicting climate and weather systems, restricts the undertaken port adaptation measures needing expensive investments. Nonetheless, the cost of natural disaster prevention is estimated to be less than the cost of treatment and this is getting more visible as previously considered unsystematic climate change related adverse events become more systematic. In these lines, are developed various statistical and machine learning algorithms aiming to capture the patterns of weather related data so as to facilitate data driven choices and measures to prevent possible huge economic losses in port authorities and other involved parties. In that regard, initiatives such as the World Port Climate Initiative (WPCI), the World Port Climate Declaration (WPCD)(Ng et al., 2018,) and the World Ports Sustainability Program (WPSP)⁶ address the consequences of climate change in the marine sector aiming to enhance and coordinate future sustainability efforts of ports worldwide and foster international cooperation with partners in the supply chain. Given this context, in the next chapters, we briefly present a view in the literature of neural network algorithms that investigate robust patterns in weather data structures.

4.3 Neural Network forecasting algorithms for time series

Weather data related or not to climate change, present high seasonality, meaning a periodic behaviour that describes the overall pattern of historical values normally anticipating to have an impact on future values as well. Of course, climate science explores the cyclicity of weather incidents happening and puts significant efforts to decompose the "change" part of the "normal" pattern of weather/climate events. In essence, the following question is addressed: "Newspapers wrote today that last weeks cataclysmic flood was unprecedented, and I am wondering did this event resulted from climate change?". There is a variety of available tools able to

⁵<https://www.epa.gov/greeningepa/climate-change-adaptation-plans>

⁶<https://sustainableworldports.org/>

recognise patterns in data structures linking today's events to an "old" trend but are also able to distinguish new "shocks" disturbing climate as we know it today. Traditionally, for time series with seasonal variations, various seasonal adjustments methods are used. In more detail, Seasonal autoregressive integrated moving average (ARIMA) models are used after achieving stationarity via seasonally differencing the initial data. In this framework, seasonal unit root models and periodic models were developed to tackle the difficulty in discriminating seasonal from non seasonal fluctuations in time series data. Tests such as the Dickey-Fuller and the Phillips-Perron are available, however they are not able to accurately identify a unit root and a near unit root process. A significant limit of the aforementioned models is that they are heavily dependent on their initial assumptions. Assumptions that are set without knowing the true hidden mechanism of the data/climate/weather generating procedure. On this grounds, the introduction of Artificial Neural Networks (ANN) on time series forecasting, overpasses this limitation, since neural networks (NN) do not need strong specific assumptions to be made about the model, because the NN model "learns" while "observing" and managing the fed data properties that make NN a promising new tool treating the peculiarities of weather and climate data. ANN can be used in various real world prediction problems, dealing with linear and non linear processes and are able to identify nonlinear trend and seasonality patterns in data. Particularly, NN with detrending and deseasonalization perform considerably better than seasonal ARIMA models in out-of-sample projections according to (Zhang and Qi, 2005,). In the following paragraphs, we set the base for time series weather forecasting with artificial neural network architectures by presenting basic NN architectures along with their conceptual framework which by extension we will use to construct the port weather forecasting models aimed at this thesis.

4.3.1 Artificial Neural Networks (ANNs)

The construction of ANNs is a scientific attempt to artificially mimic the way the human nervous system makes decisions. Observing closely, scientists uncovered that the human brain is a large connection of interconnected neurons. Neuron is considered a brain cell that collects, processes, and disseminates electrical signals. Additionally, neurons are connected via synapses and they get activated depending on the conditions of the neighboring neurons. In these concept lines, a variety of simpler and more complex ANN architectures have been developed, including different mechanisms of transmitting the information.

Initially, we observe a simple ANN with one hidden layer of two nodes. While the procedure of the forward pass, the input x is assigned the weight w_1 and inserts the node h_1 . Within the node h_1 the product of x and w_1 is transformed by the activation function. The result of this transformation, is the output of node h_1 (ah_1) which subsequently multiplied by w_3 , inserts the output node o_1 . The same procedure follows x traversing the node h_2 with output ah_2 . The ah_1 and ah_2 multiplied respectively with weights w_3 and w_4 form a sum which

enters $o1$ where again given the type of the activation function is transformed. The result of this transformation forms the final output which is an estimation of y . For every x of the input layer, the estimated y 's create the loss vector which, depending on the type of the loss function calculates the difference between the true y and the estimated y . There are many loss function choices such as the Mean Squared Error (MSE), the Mean Absolute Error (MAE), the Binary Cross-Entropy, the Categorical Cross-Entropy Loss and other custom loss functions such as the Kullback-Leibler divergence loss, the Hinge Loss etc. Now having formed the loss vector in the forward pass, the neural network moves in the opposite direction conducting back propagation. In this step, the network using the Chain Rule calculates the gradient vector of the loss functions for every input x with respect to the weights. For each input x a gradient loss vector is calculated, and summing each gradient for each input x the final loss vector is formed. The dimensions of the total gradient loss vector is 1 times the number of weight in the network. This way, the weights are optimized and the network repeats the forward propagation given the updated weights and back propagation steps as many times as the epochs indicated in the ANN architecture.

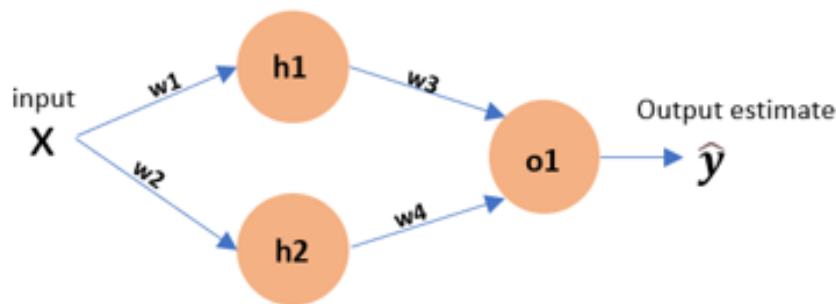


Figure 4: Simple ANN with one hidden layer.

Zooming into the hidden layer, a hidden layer may consist of single or multiple parallel nodes. Each node is also called perceptron or neuron, artificially signifying the services of a human brain cell. Zooming more, the perceptron accepts the sum of the output of the previous layers times the respective weight. If the ANN architecture imposes a bias term then this bias is also added in the sum forming a dot product of the input and the bias weight. The bias weight is also optimized as all other weights do in the back propagation process. Including a bias term has the effect of shifting the activation function by a constant amount $w_{0j} \cdot a_i$ allowing the model to adjust the output of each node. This, helps the model learn complex relations between inputs and outputs. The bias term is added to the weighted sum before being passed through the activation function, however can some times lead to overfitting. The sum of weights and inputs are transformed by the activation function forming the a_j output for each specific node.

There is a variety of activation functions available and the next paragraph briefly describes

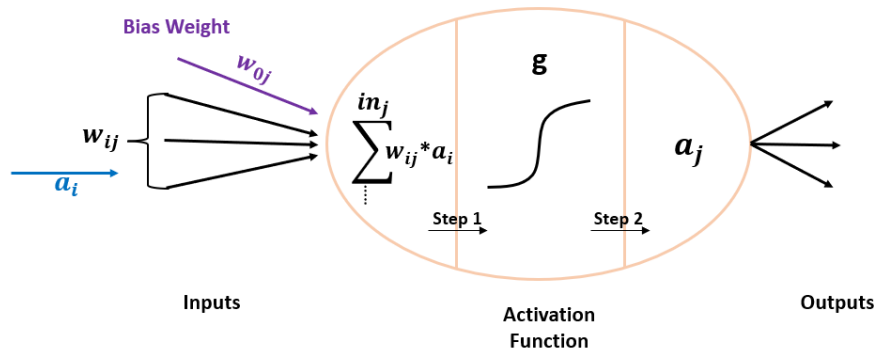


Figure 5: The Neuron of a simple ANN.

the most frequently used ones⁷. The purpose of an activation function is to introduce non-linearity to the output of the neuron. This allows the model to be trained into more complex relations. On this framework, the Linear activation function does exactly the opposite. Does not introduce non-linearity, the model does not learn something new while it combines and outputs in a sum form all the input dot products of weights and outputs of previous layers. Often, the linear activation function is imposed in the output layer where all the network collapses into one single layer that includes the actual estimated values. The Rectified Linear Unit activation function (ReLU) introduces non-linearity in the node output transforming the node output sum to zero if the output is zero or smaller than zero, or it maintains the output if the output sum is greater than zero. An other frequently used activation function is the Sigmoid activation function imposes the input sum to shrink in the range (0,1). In that case, 0.5 is considered the threshold meaning that if the sum is greater than 0.5 the output of the activation will be 1 and if is less than 0.5 the output will be 0. Interpreting this result, an output of 1 indicates that the neuron is activated transferring the signal and participating in the learning of the model, while if the output is 0 indicates that the neuron is not activated.

⁷Source: <https://iq.opengenus.org/content/images/2021/11/Comparion-of-Activation-Functions-1-.png>

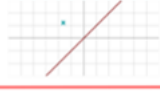

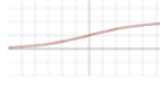
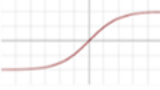

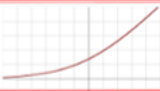

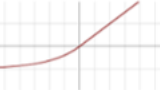
ACTIVATION FUNCTION	PLOT	EQUATION	DERIVATIVE	RANGE
Linear		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$
Binary Step		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Sigmoid		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$
Hyperbolic Tangent(tanh)		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$
Rectified Linear Unit(ReLU)		$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x > 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$
Softplus		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, 1)$
Leaky ReLU		$f(x) = \begin{cases} 0.01x & \text{if } x < 0 \\ x & \text{if } x > 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$(-1, 1)$
Exponential Linear Unit(ELU)		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$	$f'(x) = \begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$[0, \infty)$

Figure 6: Basic activation functions for ANNs.

Further, another classic choice is the Softmax activation function. This function, instead of outputting actual values, it maps the input to a probability distribution⁸ over the output classes where the sum of all derived probabilities equals to 1. The figure below displays all the output probabilities over the entire input space. Given two input classes for simplicity in a classification problem, we observe two Sigmoid functions spreading across opposite directions. The cyan and magenta are in this case a 2D Gaussian distributions of the output. Increasing the input of a certain class, we increase the output of that class moving along a Sigmoid curve also moving along a decreasing Sigmoid curve for the other class.

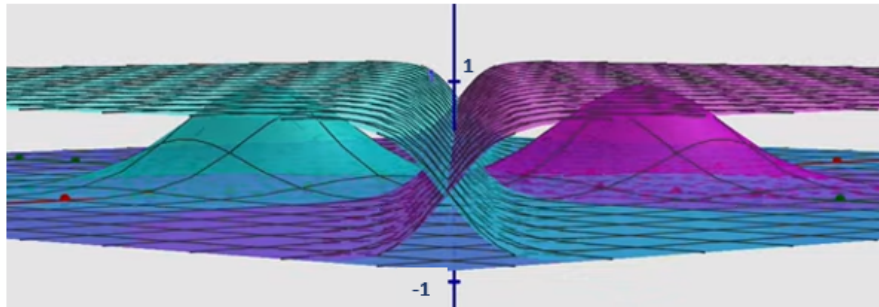


Figure 7: Visualization of Softmax activation function.

Of course this is only a glance at the analysis and concepts of activation functions could be elaborated in depth, however this is beyond the scope of this study. Now zooming out of the perceptron again, it is important to mention that a ANN hidden layer may be constructed to have different activation functions in each node and multiple parallel nodes.

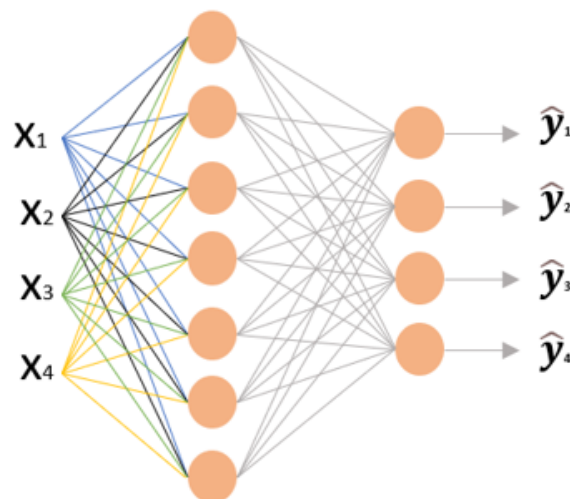


Figure 8: Simple ANN with one hidden layer with multiple inputs, parallel hidden nodes in the same hidden layer and multiple outputs.

⁸Source: <https://github.com/elliottwaite/softmax-logit-paths>

4.3.2 DNN Deep Neural Networks

As an extension of the "shallower" ANN with one or two hidden layers, is the Deep Neural Networks (DNN) architecture having three or more hidden layers and has been applied to a variety of machine learning problems such as feature extraction, feature reduction, and classification.

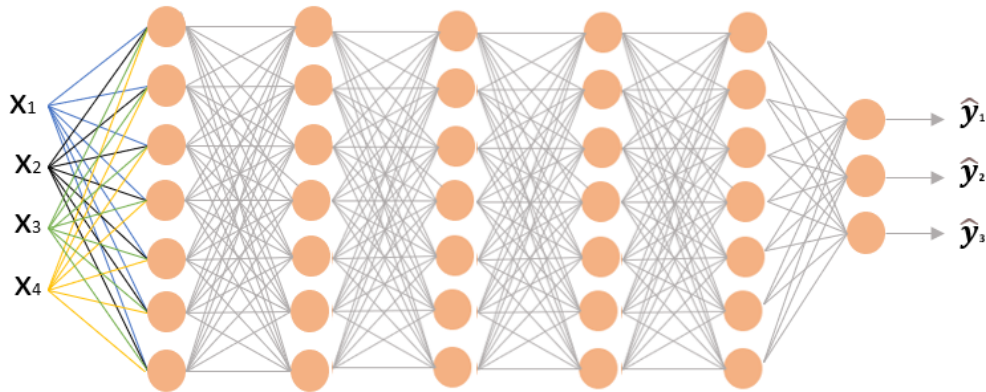


Figure 9: DNN with multiple inputs, multiple hidden layers and multiple outputs.

At this point, it is significant to briefly describe Deep Belief Networks (DBNs). DBNs are ANNs that consist of multiple hidden layers and are used for unsupervised learning techniques. Observing the figure below, Layer 1 and Layer 2 of the network contain neurons with undirected connections which form an associative memory, copying the biological neuron architecture (Awad and Khanna, 2015,). However initially back-propagation for weight optimization did not produce an accurate machine learning model. The problem lies within the assumption that the network layers are independent from each other. In essence, according to the Berkson's paradox, given two independent events the occurrence of one event negates or "explains away" the occurrence of the other in a way that one could obtain a negative correlation between them. In this framework training DBNs was a difficult task. The solution to this problem was introduced by (Hinton et al., 2006,) that proposed a greedy training algorithm that trains the restricted Boltzmann machines (RBM) units independently before adjusting the weights using an up-down algorithm to avoid underfitting.

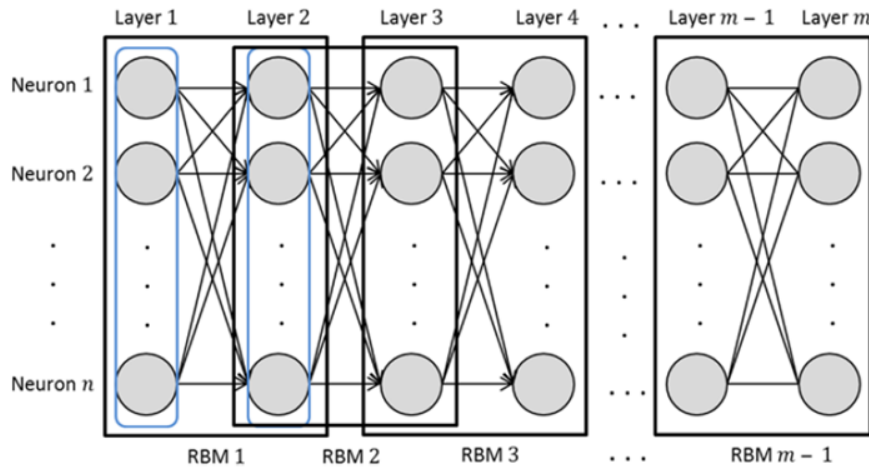


Figure 10: DNN with multiple inputs, multiple hidden layers and multiple outputs.

4.3.3 Simple RNN Recurrent Neural Networks

Recurrent Neural Networks are a class of the neural network algorithms that are mostly used in tasks where sequence data need to be modeled. Text analysis, time series data, sales forecasting, stock forecasting are some examples where RNNs are widely used offering prediction results with high accuracy. Focusing on time series tasks, traditional statistical techniques such as Autoregressive Integrated Moving Average (ARIMA), SARIMA accounting for seasonality in data etc. use some window of the past values to predict the future value. In contrast, an RNN model aiming to predict the new value, will take into account the total amount of past instances in the data making sure that the sequence information is maintained. In this framework some time window blocks could also be considered, but this is dependent on the RNN architecture discussed below. More analytically, forward propagation through time is a key concept in simple RNN structure.

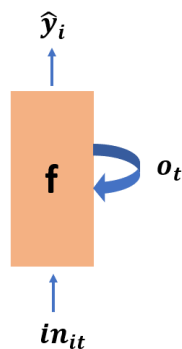


Figure 11: Forward propagation RNN general architecture.

Observing the figure above, the input in_{it} of any dimensionality enters the hidden layer part in orange of the RNN. The hidden layers may be of any number and having single or multiple number of neurons each. Next, \hat{y} is the output of the RNN which is the estimation of the future

value we want to predict. Along with the \hat{y} , we obtain o_t which is an output with respect to time. More analytically, assuming that the input in the RNN is the number of occurrences of extreme weather event incidents for $t = 1$, for $t = 2$ the estimation of the next count of instances \hat{y}_2 will incorporate the information of the forecast of the previous time step o_{t1} , this is done for every input inserted in the RNN training the model, and this way the sequence information is kept. Now, unfolding the general architecture of the described RNN we zoom into the orange part of the above figure.

The expanded RNN architecture is visualized in the figure below.

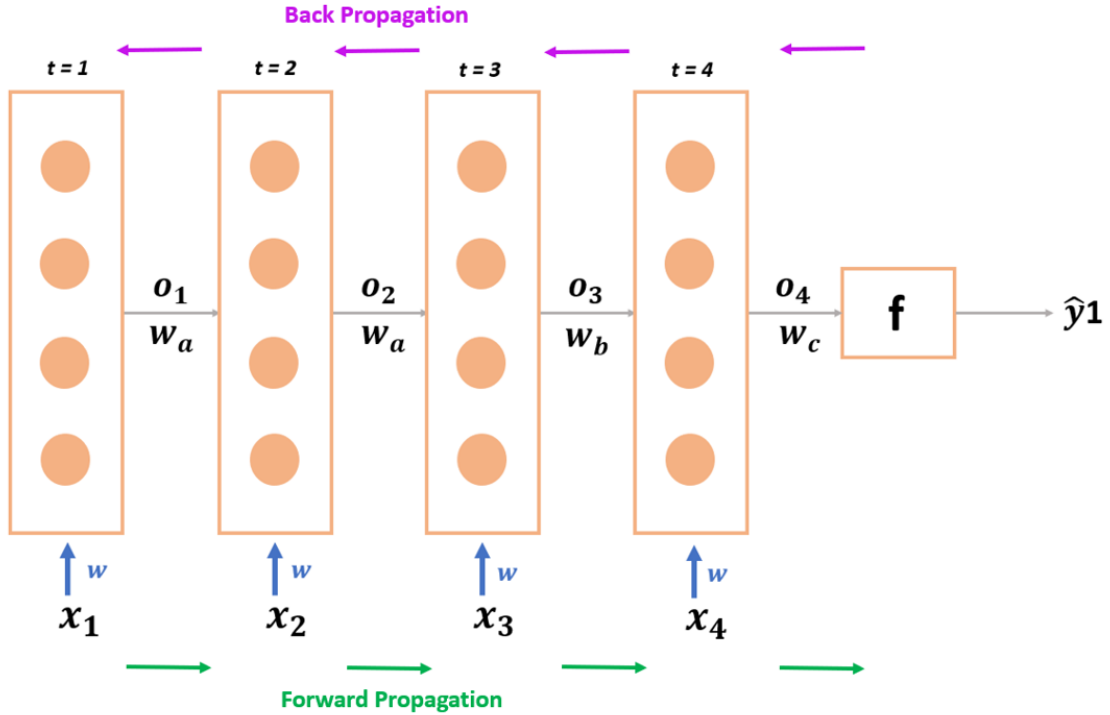


Figure 12: Expanded RNN architecture.

The first element x_1 of the four element time series is multiplied with the weight vector and then it is passed to the first layer starting from the left side. In this particular example, the hidden layer has four neurons hence the output of the first layer o_1 is the sum of the first input times each respective weight dedicated to each specific neuron passed through the respective activation function. Having computed the o_1 output, according to the RNN process, the same output should be fed again to the same hidden neuron while in parallel for time $t = 2$ the next element of the time series will get passed to the second hidden layer. As the second hidden layer accepts the second element x_2 with initial weight vector, is also accepts the output of the previous layer o_1 multiplied by a different weight w_a . In essence, the output of the second layer o_2 is the sum of x_2 times the weight vector plus the o_1 times the w_a , always passed through the dedicated activation function. Since o_2 is dependent on x_2 and o_1 this allows RNN to keep

sequence information. At this point, we should note that in all four layers the same weight vector w is assigned as the forward propagation takes place. The weight vector w will only be updated in the backward propagation process conducting weight optimization. For time $t = 3$, the output of the second layer is fed again to the same neurons, so o_2 inserts layer 3 multiplied with the same weight w_a . Accordingly, the third element of the time series $w_{1,3}$ is also getting passed to layer 3. The output o_3 , will be the sum of x_3 times the initial weight vector plus the output o_2 times w_a passed through the dedicated activation function. For the last hidden layer, the output o_4 will be x_4 times the weight vector plus the output o_3 times w_b a different weight. Finally the o_4 times w_c will be passed to the activation function of the output layer giving the prediction. Then the loss function is computed subtracting the actual value from the predicted value. The main goal of doing back propagation is to reduce the output of the loss function. Focusing now on the back propagation procedure, using Chain Rule the derivative for each weight is calculated. To update the weight w_c for instance, the new w_c weight will be equal to the old weight w_c minus the derivative of the loss function with respect to w_c . Returning back to layer 3, to update the weight vector w the first step is to calculate the derivative of the loss function with respect to w . To do so, with the help of Chain Rule, the previous expression equals with the dot product of the derivative of the loss function with respect to \hat{y}_1 , the derivative of \hat{y}_1 with respect to o_4 and the derivative of o_4 with respect to finally w vector. We note that even though in forward propagation the weight vector w applied is always the same, in the pack propagation pass, the vector will updated 4 times in the way it was described above. Thus, all weights will be updated for the first epoch. This front and back pass will be repeated as many iterations as it is imposed by the RNN architecture. Finally, after some number of iterations the RNN will have optimized the weights and this way will be able to reach the global minimum where the training process of the RNN will stop.

RNNs may have various structures⁹ in terms of input and output dimensions. In the case of *one – to – one* structure, the feedforward RNN model accepts a single input and it outputs a single output estimation with no temporal dependencies between them. However in the case that the input data is for example built by the user to signify per year observations, a typical format for time series, the predicted output after the training of the model, indicates the forecast for that specific variable for the next year.

⁹<https://stanford.edu/shervine/teaching/cs-230/>



Figure 13: RNN one to one.

Next, in the case of a *one – to – many* RNN, the model structure accepts a single input and it predicts a sequence of outputs. The initial input is fed into the RNN, and the output of each time step is fed as the input for the next time step. *One – to – many* RNN structure, is often used in tasks such as Music Generation. In that case, the RNN model is trained to take a single input (a sequence of musical notes) and generate a longer sequence of musical notes that follow a similar pattern. In this case, the RNN is trained to predict the next note in the sequence based on the previous notes, and this process is repeated until the desired length of the sequence is generated.

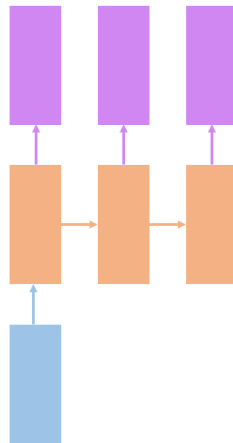


Figure 14: RNN one to many.

Following, when the RNN has a *many–to–one* structure, it accepts a sequence of inputs and it predicts only one output. This structure, is commonly used in natural language processing tasks, speech recognition and sentiment analysis task where the model should take into account various words and elements of a text and produce a sentiment score. This can be done using various types of many-to-one RNN architectures, such as the basic RNN, LSTM, or GRU, RNN structures that will be discussed in the next paragraphs.

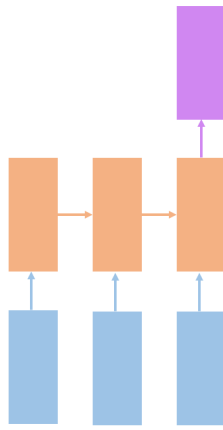


Figure 15: RNN many to one.

Further, *many – to – many* is an RNN model structure that accepts a sequence of inputs and predicts a sequence of outputs. This is a useful structure, in the case where the user identifies a correspondence between the elements of the input sequence and the elements of the output sequence. The input sequence is inserted into the RNN one element at a time, and the output sequence is generated in parallel. In the synchronous approach, the input and output sequences have the same length, and each element of the input sequence is processed in parallel with its corresponding element in the output sequence. This approach is commonly used in tasks such as speech recognition and machine translation. In the asynchronous approach, the input and output sequences can have different lengths, and the RNN continues to produce output after the input sequence has ended. This approach is commonly used in tasks such as video classification, where the RNN is trained to recognize actions or events in a sequence of frames.

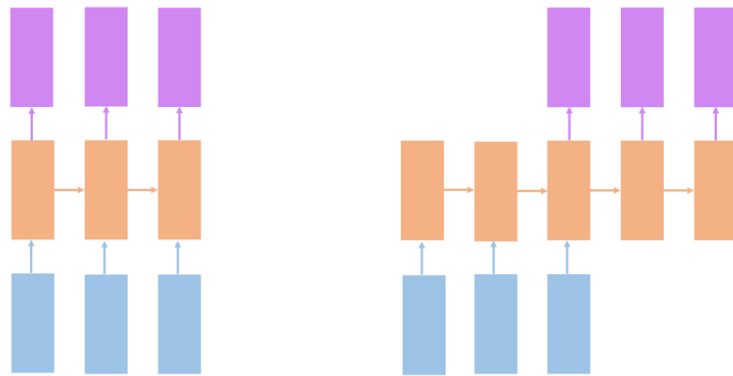


Figure 16: RNN many to many structures.

The left figure displays a model with more hidden layers than the number of input and output sequence elements, while the right figure displays a model with as many hidden layers as the number of input and output sequence elements.

4.3.4 Problems with Recurrent Neural Networks

Although simple RNN's are easy to interpret and built, there are some important characteristics in their architecture that are problematic in terms of their ability to accurately predict unseen information. In the next paragraph these issues are discussed and visualized in order to understand why in many time series tasks are often used LSTM's or other more complex neural network algorithms. As it was stated in the Chapter 2.3.3, RNN's given the input conduct forward propagation and backward propagation over time training the model. In the backward passing, the derivative of the weights is updated continuously optimizing the assigned weights with respect to time. Hence, while updating the weights, the information passes through the designates activation function in each neuron. In case the activation function is a Sigmoid, through the Chain Rule, the derivative will always be between 0 and 1. In the first back propagation step, this may not be a problem, however updating all the weights from right to left, leads to a very small value close to zero for the derivative which means that the initial weights are negligibly updated which in extension means that the model does not learn well. Particularly, observing the Figure 17 (a) the model will never converge to the global minimum point of the gradient descent of the loss function because of the very small learning rate. Alternatively, in case the activation function is ReLU, the derivative will be greater than 1, meaning that towards the updating of the weights the learning rate will be so big causing the problem of exploding gradient descent as displayed in the Figure 17 (b), that again doesn't allow the RNN to converge to the global minimum of the loss function resulting to bad training.

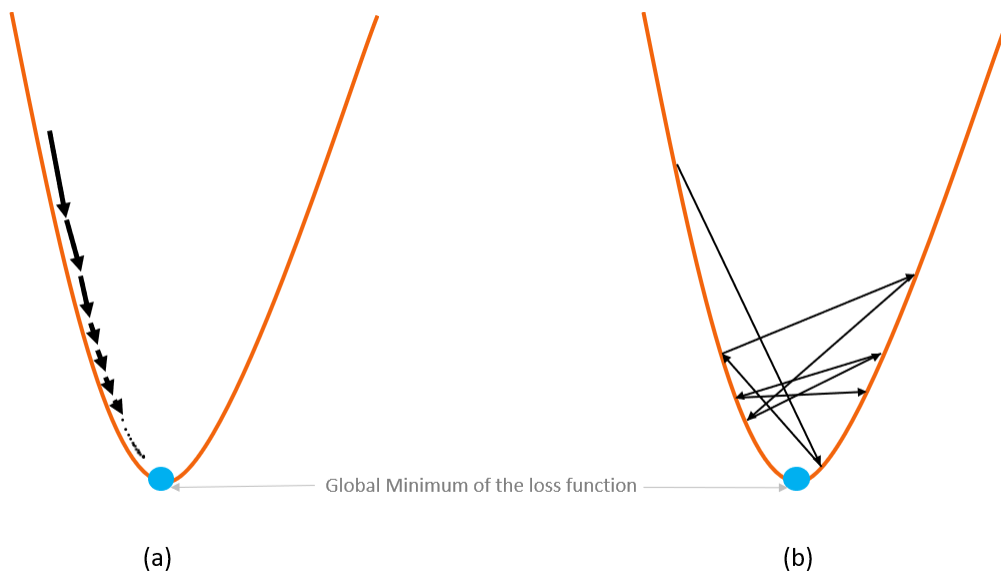


Figure 17: Vanishing and Exploding Gradient Decent at RNN.

Further, the ability to converge to the global minimum has also to do with the "landscape" of the loss function. On this context, the previous examples implies loss functions that are convex. If the loss function is non-convex, then there is not only one minimum the global

minimum but there could be also multiple local minima. Thus, when applying gradient descent as optimization method in a non-convex function it is not certain that the reached minimum is the global minimum, which means that the model will stop the learning process-training without having optimized the weights no matter the number of iterations. Additionally, an other landscape particularity of the loss function could be a "Plateau" region where as the RNN tries to learn by updating the weights, the weight correction (the derivative of the loss function with respect to the weights) will be the same because the region of the objective function is almost parallel to the horizontal axis. In that case, with a very small learning rate the gradient of the loss function will not be able to reach the global minimum.

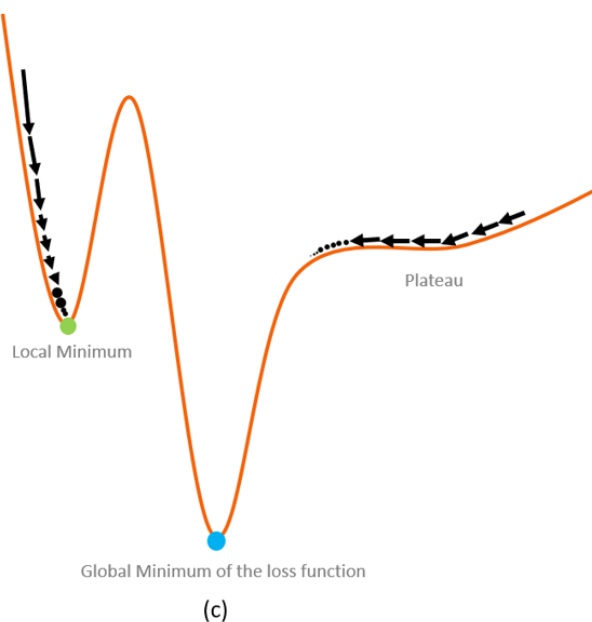


Figure 18: Problems with non-convex loss function in RNNs.

As a consequence, for non-convex loss functions instead of gradient-based optimization algorithms there are various others that are able to overpass the discussed problem. A widely used optimization algorithm is the Adaptive Moment Estimation (ADAM) optimizer which contributes to the computation of learning rates for each weight w by using the first and second moment of the gradient. ADAM optimizer requires less memory and outperforms on large data sets. Additionally, the Stochastic Gradient Descent (SGD) by randomly selecting data samples and updates the parameters according to the loss function. SGD converges faster than the ordinary Gradient Descent and requests also less memory by not accumulating the non optimized intermediate weights. Another optimization algorithm, is the Adaptive Gradient Optimizer (Adagrad) which adapts the learning rate for each parameter based on the previous gradient information during training. There are many more optimization algorithms for non-convex objective functions but this is beyond the scope of this Thesis.

Now to overcome the Vanishing and Exploding Gradient Descent problems caused by the simple RNN architecture, we could use models such as Long Short-Term Memory (LSTMs),

more analytically presented in the next Chapter.

4.3.5 LSTM Long Short-Term Memory RNN's

The visualization below displays a simple LSTM RNN cell ,for x_t the inputs and H_t the outputs for the given time. Note that the mechanism called "Gate" refers to a neural network layer inside the LSTM cell that regulates the flow of information being passes from one time step to the next while performing the various operations described below.

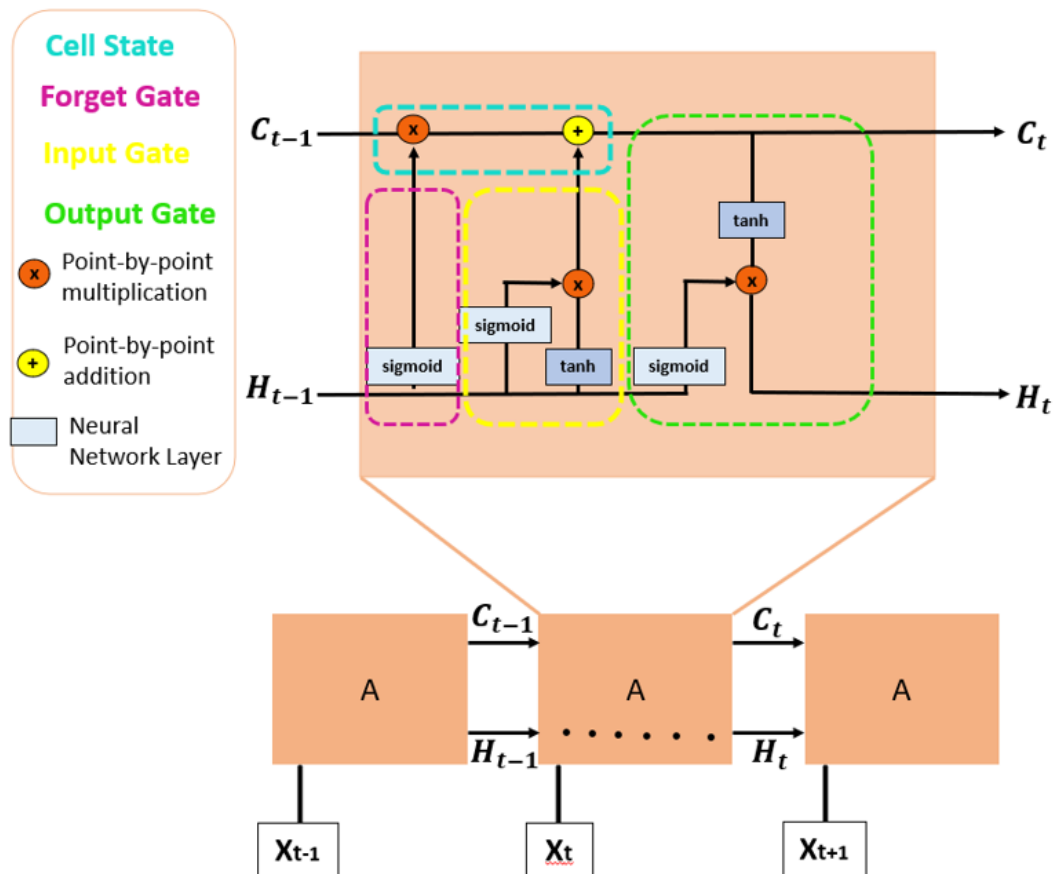


Figure 19: LSTM cell

Zooming in this architecture, in the dashed region is the functions of the "cell state" or "memory cell". This cell is used to remember and forget information. It accepts the previous state output and conducts point-by-point multiplication between the previous state output and the input. This results to a new vector that tells the LSTM cell the specific information it needs to remember in case the context has changed. If the context of the input has not changed, then the cell state remembers all the information from the previous state output. Next, the point-by-point addition adds new information to the "remembered" information vector. The final output of the cell state will be the previous state output for the next cell for time $t + 1$. Regarding the "forget gate" in magenta, it accepts the input x_t and the output of the previous output H_{t-1} , concatenate them, assigning two different weights (plus bias if applied) and pass

them through the Sigmoid activation function which decides at last what information to forget. Next, from the "input gate" in yellow, new information is added to the memory cell. In this step, the input x_t and the output of the previous output H_{t-1} is passed through a Sigmoid activation function squeezing the result between 0 and +1 and the same two inputs are passed through a Tanh activation function which converts the input between -1 and +1. The outputs of the two activation functions are point-by-point multiplied and whatever information is kept is described with 1 in the output vector of the input gate. Then, this information as was described before, is added point-by-point to the memory cell. In essence, the aforementioned operations indicate that any new information from the *Tanh* activation function will be kept if the input context has changed, only then it will be added to the memory cell. This way, the previous state output is updated to the new state output (black arrow on the top of the LSTM cell). The green region, is referred as the "output gate". What ever information was kept in the memory cell, will be converted through the *Tanh* function to a value between -1 and +1, and next will be point-to-point multiplied to the output of the Sigmoid activation function. Conceptually, the output gate enables the LSTM cell to retrieve only the information which have a meaningful context to add to the training of the model and that will be the output H_t that will be passed to the next LSTM cell in time $t + 1$.

4.3.6 Gated Recurrent Unit (GRU)

While LSTM has separately long term memory and short term memory, Gated Recurrent Unit (GRU) is a modified version of LSTM which combines long and short memory into its hidden state. Continuing with the comparison, LSTM has three gates, input, output and forget gate while the GRU cell has only two gates, the "update gate" and the "reset gate". GRU's have less tensor operations and parameters to update hence are more computationally efficient than LSTM's.

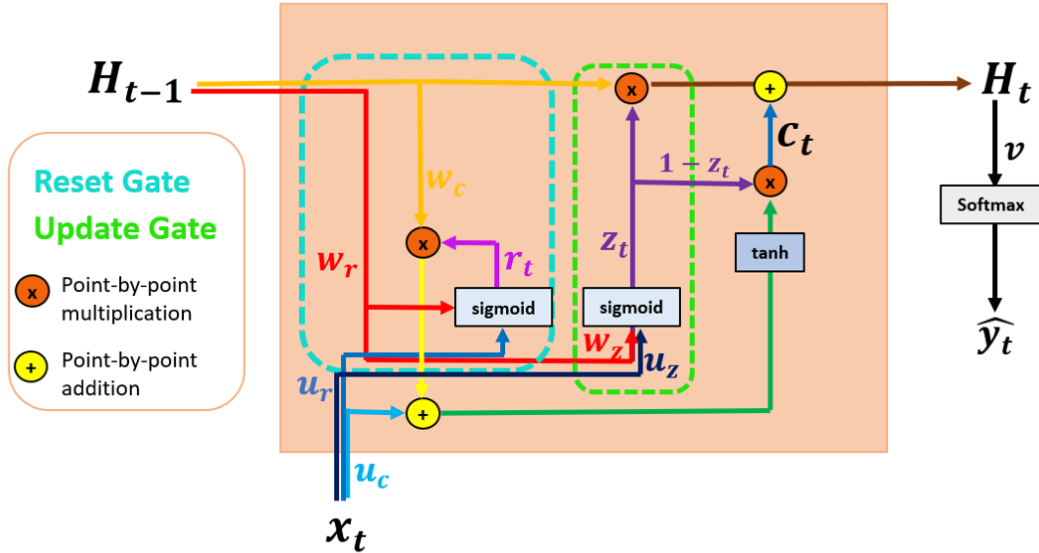


Figure 20: GRU cell

The role of the "reset gate" gate is to control how much information we want to remove from H_{t-1} to H_t . Whereas, the "update gate" controls how much information we want to add to the previous information (from the new input x_t) to H_t . Before the information enters the next hidden layer new weights and biases are assigned in H_{t-1} , x_t , and the activation functions outputs w, u, v respectively as the typical neural network process. The weight $1 - Z_t$ is the complement of Z_t to avoid including an additional gate. In more detail, supposing we are at time t , H_{t-1} is the information passing from the previous GRU cell to the current GRU cell. The output of GRU cell at time t , is H_t which eventually is fed to the next GRU cell at time $t + 1$. The information from H_{t-1} and H_t , are the information stored in the hidden state of GRU. The way the two gates help in changing the hidden state is by adding new information and removing information. To add new information, a new candidate state c_t is created to hold the information which will be added to the previous information H_{t-1} using the plus operation in the yellow disc on the right. To remove information, from the hidden state, z_t which is the update gate, decides what part of the the previous time step H_{t-1} can be taken forward to the next time step H_t . Thus, following the red arrow of H_{t-1} directed in passing through the second Sigmoid and being filtered from z_t , outputs the relevant information from the previous time

step. Finally, information passed to the next GRU cell is $H_t = (1 - Z_t) \cdot c_t + Z_t \cdot H_{t-1}$. Then, the result of the hidden layer H_t is multiplied with the weight matrix u and is passed to the last activation function Softmax which gives the estimation of \hat{y}_t , meaning the input x_t mapped to a probability distribution over the output classes. Indeed, the GRU model architecture as the LSTM's, solves the problem of vanishing gradient descent while conducting back propagation weight optimization but it is more time efficient since it has less parameters to train.

4.3.7 Transformer models with Attention mechanism

In 2017, (Vaswani et al., 2017,) introduced Transformers which is a radical NN model which is an attention based encoder decoder architecture that is mostly used in natural language processing (NLP) tasks such as language translation, language generation, text summarization, question answering and sentiment analysis. Briefly, given sequential or structured input data the attention mechanism allows the model to focus on specific parts of the input sequence instead of the whole input set, based on the relevance of the target task. The reason why this model architecture is considered better than the previously described model is because RNN's take into account historical inputs of a shorter time window hence they suffer from "shorter memory". In the case for example of a language generation task, when the model generates a longer text, vanilla RNN's are not able to access words generated early in time sequence. The same holds for LSTM's and GRU's, however the aforementioned allow for a longer historical time window to reference from and this is the reason they are named long-short memory models. In contrast, the attention mechanism theoretically and given enough compute resources, allows for an infinite time window to reference from. In this context, the transformer model in the previous example of the text generation, is able to reference to the whole text when generating a new word and not only to a shorter window of the word sequence. On a high level, the encoder part (on the left) maps an input sequence into an abstract continuous representation that holds all the learned information of that input, the decoder part of the architecture (on the right) takes that continuous representation and progressively generates a single output while in parallel feeds the previous outputs into the encoder recurrently until an "end of sentence" token is generated. Currently, there is great interest in this type of architecture and generally, the attention mechanism can be applied in RNN's, NLP models and Convolutional neural networks (CNN's) used in image and video tasks, improving the performance on language models.

Having presented briefly the basic NN architectures, we set the basis for the weather models constructed in Chapter 3.4. Among all discussed NN architectures, we choose to focus on LSTM models performing well in time series data aiming to forecast weather patterns in the future.

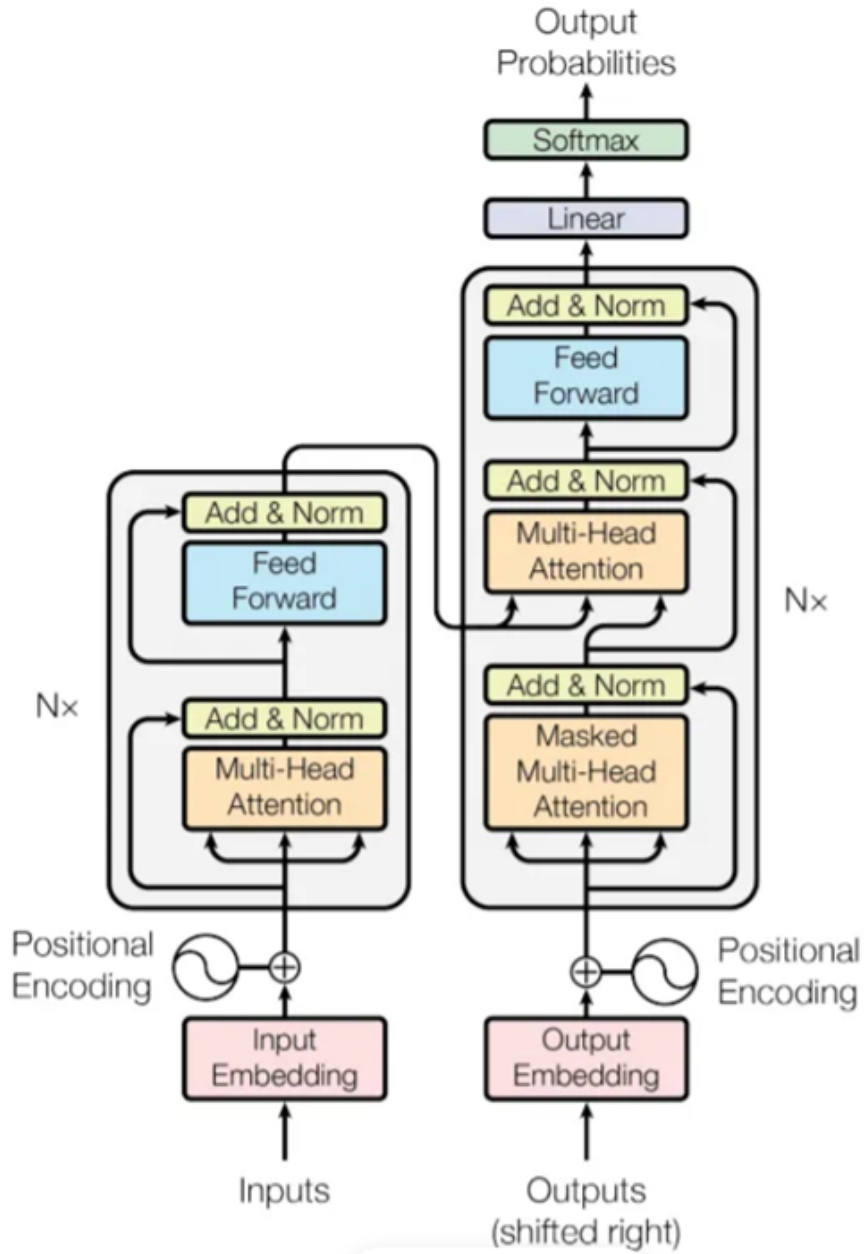


Figure 21: The architecture of Transformer model from the initial paper (Vaswani et al., 2017,)

4.4 Neural Network models for weather prediction

Having described the basic architectures of NN algorithms, it is meaningful to investigate specific applications presented in the relative literature regarding NN and their contribution in the field of weather prediction before structuring the models needed for this Thesis. The models described below do not focus specifically on ports, instead they present a wider framework of weather pattern forecasting useful for many applications amongst which is port resilience on extreme weather events.

Currently, weather prediction is based on the Numerical Weather Prediction (NWP) procedure¹⁰ (Lorenc, 1986,), (Bauer et al., 2015,). However, limitations such as the sensitivity to initial conditions of the model, the partial knowledge of the atmospheric physical mechanisms while modeling, the enormous amount data obtained from sensors and from simulations, the data set uncertainties in combination with spatio-temporal correlations, along with the computationally expensive non-linear equation handling introduced the tool of artificial neural network as alternative model to traditional NWPs (Ren et al., 2021,). Specifically, Deep Learning Neural Networks (DLNN) perform well in situations where the system behaviour is dominated by spatial or temporal context and thus there is a need of time and space feature extraction, a typical example is the weather data forecasting. In this case, Deep Learning Weather Prediction (DLWP) models treat weather data as a multi-dimensional time series problem.

Regarding the data features, different DLWP are applied in different big weather data structures aiming to reveal the linking mechanisms and to capture the weather pattern changes. More precisely, on one hand real type data structures from in situ observations in the form of longitude, latitude, level and time are efficiently handled by Autoencoder. On the other hand, in the case of satellite image data, given the very big amount of image data produced each day, Convolutional Neural Networks (CNN)++ are widely used for image processing and early forecasting and warning of extreme weather events. Last but not least, in the case of long term sequence weather data some times going back hundreds of years ago, are applied long-short term memory (LSTM) models a special case of Recurrent neural networks (RNN) that handles the problem of vanishing and exploding gradient descent of the loss function. LSTMs, are effective on identifying long term sequences since they are equipped with a memory function dedicated to make the model remember event incidents that it had seen in the distant past. In this context LSTMs are suitable for long term weather forecasting and climate simulation data structures.

Against this background, it is meaningful to mention two more NN model groups for weather forecasting. Firstly, the Hybrid architecture Deep neural network (DNN) models. This family

¹⁰<https://www.ncei.noaa.gov/products/weather-climate-models/numerical-weather-prediction>

of models focuses in extracting the spatio-temporal features of the weather data sets. The (Shi et al., 2015,) propose a Convolutional LSTM model aiming to predict the rainfall intensity on a zero to six hours time frame offering early warning information for social protection schemes and airport traffic management. To do so, the traditional LSTM model is equipped with a convolutional layer in the encoding part of the NN incorporating the time and space link of the meteorological data. This approach, increases the accuracy level of the network. Besides, (Shi et al., 2017,) suggest that Convolutional LSTM models are location invariant hence there is a need for a model that could learn the location variant structure of the precipitation data. In this framework, given the requirement for extensive, real-time and high-resolution regional precipitation nowcasting led to the introduction of a Trajectory Gated Recurrent Unit model that uses a sub network to output the state-to-state connection structures before state transitions. Further, this paper constructs a benchmark data set containing radar echo data of 6 years from a Chinese region with the purpose of pushing nowcasting to be trained on real-time learning achieving dynamic forecasting power. A balanced mean squared error and a balanced mean absolute error is proposed to assign higher weight to precipitation with high intensity as more appropriate evaluation metrics for the target forecasting task. Moreover, (Wang et al., 2017,) construct a predictive recurrent neural network based on the concept of a dual memory structure that learns the spatial occurrences and the temporal variations of weather events at the same time (spatio-temporal LSTM) trained on video data. An improvement of the previous model is developed by (Wang et al., 2018,), who built a causal LSTM with cascaded dual memories able to reduce the gradient vanishing complications in deep-in-time predictive models. Specifically, this paper introduces a gradient highway unit providing the gradients with quick routes from future forecasts to back in time older inputs. In a Deep Convolutional Neural Network (CNN) context (Agrawal et al., 2019,) propose the usage of the U-Net architecture initially used for biomedical image segmentation, for precipitation nowcasting. Specifically, this model application treats forecasting as an image to image translation problem, accepts a sequence of radar images and is able to generate a high-resolution 1km by 1km precipitation image for the next hour. In that case, the traditional NWP weather prediction, is transformed to a data driven input-output problem with images. Along similar lines, (Sønderby et al., 2020,) built a CNN with attention mechanism called MetNet and use it in an application, in order to forecast the location of the precipitation events in space. Particularly, MetNet accepts radar and satellite image data and predicts precipitation up to 8 hours into the future, every 2 minutes with latency in the order of seconds and at the same space resolution as U-Net, outperforming NWP as well.

In the previous chapters we presented and discussed the basic NN architectures and their functionalities. Further, we investigated various applications of NN in the literature of weather prediction offering new tools to the traditional Numerical Weather Prediction field. Against this background, the following chapters unfold the steps of port weather data preprocessing and forecasting based on LSTM architectures aiming to investigate the way extreme weather

incidents hinder the normal port operations.

5 Research Design and Methodology

5.1 Data Description

Regarding the data used, this analysis uses open-source data from the National Oceanic and Atmospheric Administration. Specifically, the time series data contain Severe Storms and Extreme Events occurrences in various parts in USA. Extreme weather data cause severe damages to life and property causing increased and often unexpected economic damages. More analytically, the data contain 48 different types of severe events, from localized thunderstorms, tornadoes, and flash floods to regional events such as hurricanes, derechos, and winter storms. These data are collected by the National Weather Service. Weather offices detect events using instruments and visual observations, and they also receive information from storm spotters - people who call in to report severe events. Tornadoes, high wind speeds, and storm cell data are collected with radar. Extreme weather events, in the database are selected based on the storm intensity, causing loss of life, injuries, significant property damage, and/or disruption to commerce, and rare or unusual storm weather phenomena along with significant meteorological events, such as extreme temperature. Data contain numerical values and descriptive text. The data can be found in the following link. <https://www.ncei.noaa.gov/pub/data/swdi/stormevents/csvfiles>.

5.2 Data Cleaning

Conducting the data cleaning process, initially the raw csv files for the years 1995-2022 are loaded containing 1.551.853 rows and 7 columns. The data for each year are stored in separate data frames while for this analysis only the following variables are maintained displayed in the following table.

<i>YEAR</i>	Year of the event
<i>MONTH_NAME</i>	Month label
<i>BEGIN_DAY</i>	Begin day of the event
<i>BEGIN_DAY_TIME</i>	Begin day of the event
<i>END_DAY_TIME</i>	Begin day of the event
<i>STATE</i>	State where the port is located
<i>EVENT_TYPE</i>	Type of the event (e.g., heavy snow, hail, thunderstorm wind, flood)

Figure 22: Variables Description

The reason why we choose the specific time span in because the years after 1995 display an interesting weather landscape and specifically severe storms, in comparison to the previous years available in the NOAA’s data set, translating to continuously increasing billion-dollar disasters¹¹.

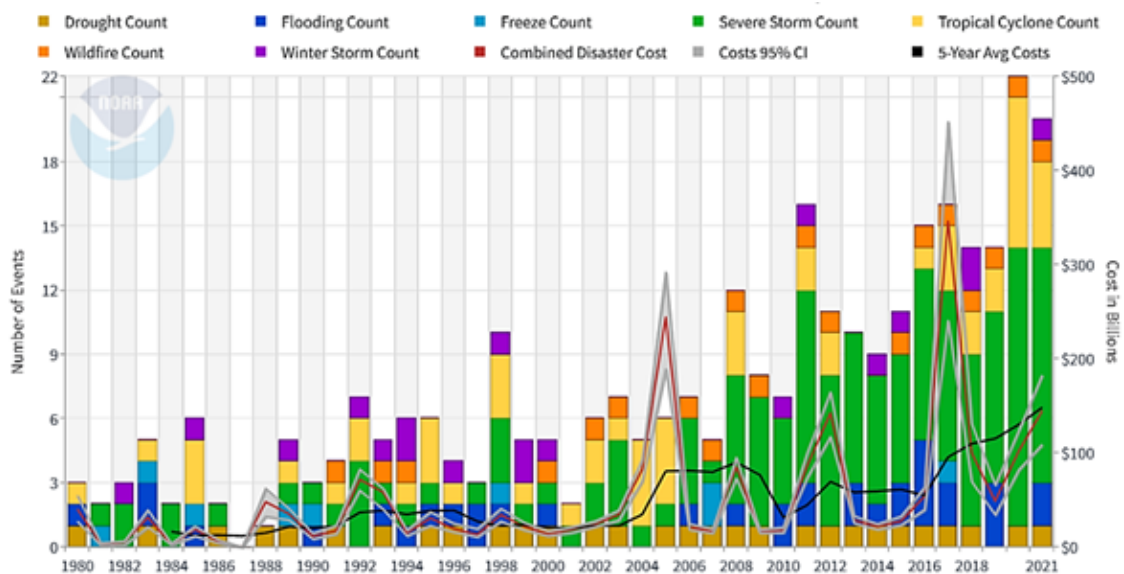


Figure 23: United States Billion-Dollar Disaster Events 1980-2021 (CPI-Adjusted) according to NOAA NCEI.

The time span choice is confirmed by the NOAA historical data, where indeed after 1995 the total number of extreme weather registered incidents in the USA increases almost steadily. Specifically, the number of severe storm events display a significant escalation, while this event

¹¹<https://www.climate.gov/news-features/blogs/beyond-data/2021-us-billion-dollar-weather-and-climate-disasters-historical>

type is highly linked to port operation disruptions causing great economic damages.

Next, the separate data frames containing daily weather data for each specific year are concatenated in a single data frame named DATA in order to facilitate the next cleaning steps. Following, the columns *MONTH_NAME*, *BEGIN_DAY*, *BEGIN_DATE_TIME* and *END_DATE_TIME* are renamed to *MONTH*, *DAY*, *BEGIN_TIME* and *END_TIME* respectively. Additionally, months are mapped to their numerical value so as to enable sorting, while by applying an additional filter, from the total set of States existing in the data set, only the those that have big commercial ports are maintained¹². Specifically, are maintained only the following States with the 10 biggest commercial ports, CALIFORNIA with the Port of Los Angeles, Port of Long Beach and Port of Oakland, GEORGIA with the Ports of Savannah and Brunswick, WASHINGTON with the Port of Seattle and Port of Tacoma, TEXAS with the Port of Houston, SOUTH CAROLINA with the Ports of Charleston and Georgetown, VIRGINIA with the Port of Virginia along with FLORIDA with the Port of Miami. Regarding the Port of New York and the Port New Jersey, we maintain only the State of NEW YORK (Port Authority of New York New Jersey). On these lines, is assumed that maintaining weather data for the whole state of NEW JERSEY will lower the model accuracy since NEW JERSEY is a very big state and the ports we are interested in are geographically very close.



Figure 24: Geographical proximity of Port of New York and New Jersey.

Further, the data frame DATA is checked for not having null values. Further, we create

¹²<https://www.icontainers.com/us/2017/05/16/top-10-us-ports/>

a new column calculating the duration of the weather event by imposing *datetime* format to the columns *BEGIN_TIME* and *END_TIME* and subtracting one out of the other. Besides, the data set DATA is split into 8 different data frames by filtering the name of the state of interest, namely CALIFORNIA_all, NEW_YORK_all, GEORGIA_all, WASHINGTON_all, VIRGINIA_all, TEXAS_all, SOUTH_CAROLINA_all, FLORIDA_all. In all 8 data frames the weather information is sorted firstly by year, next by month and lastly by day. Moreover, in each separate state data frame, a new column is inserted containing the year, month and day in a *date.time* format and is set as index. Given the previous step, only the *EVENT_TYPE* and *DURATION* column are maintained. At this point, it should be mentioned that, the initial data set contains registrations of 66 different types of weather events but for the purpose of this analysis and in the lines of (Athanasatos et al., 2014,) and (Cao and Lam, 2019,) we filter the 8 state data sets only for weather events that have significant impact on the port operation routine. In this framework, each data frame is filtered for weather events of High Wind, Heavy Rain, Winter Storm, Dense Fog, Thunderstorm Wind, Marine Thunderstorm Wind, Tornado, Strong Wind, Flood, Marine High Wind, Dust Storm, Extreme Cold/Wind Chill, High Surf, Flash Flood, Excessive Heat, Ice Storm, Coastal Flood, Tsunami, Marine Hail, Marine Strong Wind, Tropical Storm, Marine Dense Fog and Marine Tropical Storm types.

Additionally, the initial data set contained weather event registrations that often were multiple for one day. Meaning, for the same day the data had multiple registrations of events with different duration, while the sequence of days didn't have a constant step e.g. registrations for every consecutive day. The first problem is treated by grouping each state data set by event type and by duration. Hence, after this transformation the data set contains for a specific day the event types in a joined string (e.g. "Flash Flood,Flood") accompanied by the aggregated duration of the total number of events for that day. The second problem of the missing days is treated by expanding the data set of each port including the missing days, where we do the assumption that the missing day, was a day that was overall good and not severe weather events occurred. In this context, we fill the NaN values that were summoned after expanding the data set to its full per day length imposing an *EVENT_TYPE* of "No event" and a *DURATION* of "0 days 24:00:00". Furthermore, regarding *DURATION* we faced another challenge regarding the aggregation of duration of the registered *EVENT_TYPES*. In practice, we observed that the event types were registered in the system probably by different radars, where multiple radars recognized the same signal. Hence, often when adding the various event types for specific day, the duration sums up to more than 24 hours which is something not possible. The way we decided to encounter this problem since data for radar locations are not available in the raw data set to choose only a single one, is to filter the registrations in all 8 data frames and when a duration of a "bad weather" day incidents overpass 24 consecutive hours, then we assume that in all 24 hours of the day intense weather events were occurring, and we state that by replacing the previous duration with the 24 : 00 : 00 format. After all the aforementioned preprocessing

steps, the 8 data frames per state are grouped by season to prevent accuracy failures due to the inherent seasonality of weather data sets.

Next, the final preprocessing step is the transformation of the created data sets to a format that could be fed to an LSTM architecture. To do so, taking the example of the port of Florida, the four seasonal subsets *fl_autumn*, *fl_winter*, *fl_spring*, *fl_summer* are pre-processed. Firstly, the column of DURATION is transformed to integer seconds type, and next EVENT_TYPE column is transformed to string format filling the NAN values with the empty string. Secondly, the EVENT_TYPE column in all 4 season data sets is transformed using a LabelEncoder into numerical values. Thirdly, the seasonal data sets are scaled using the MinMaxScaler because event types have values between 0 and 185 mostly while duration has often 4 or 5 digit number, preventing LSTM to perform. Finally, an important step is to transform the four data frames into 4 arrays, from which the train and tests sets are created using a test size of 10%. The number of past values the LSTM will depend on to predict the future is imposed at five for a multivariate ($n_features = 2$) prediction. *TimeseriesGenerator* is used to handle variable length data sequences conducting the final transformation of training and tests sets to enter the LSTM. For the state of Florida, there are 4 couples of train and test set for each season, e.g., *generatorTrain_fl_AUTUMN* used to train the model and *generatorTest_fl_AUTUMN* used as validation set to test the model accuracy in unseen data. The data analysis and forecasting that will be elaborated in the following section, are applied on each clean 8 state data sets and for each of the four seasons. At this point it is crucial to state that given the lack of data specifically from each port region, we make the assumption that the data reference to the port location and that the port is in the center of gravity of the state. The next figure is a visualization of the ports¹³ under investigation throughout this thesis.

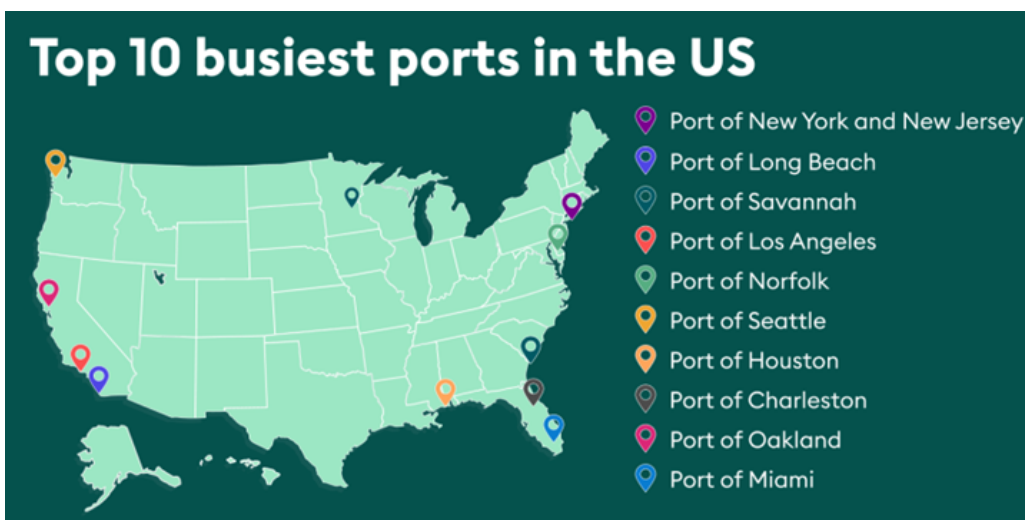


Figure 25: Geographical locations of the ports under observation

¹³<https://www.container-xchange.com/blog/busiest-ports-in-the-us/>

In the next chapter, we present the experiments regarding alternative LSTM architectures. Moreover, are presented the best model choices upon which weather forecasts are estimated aiming to explore possible future port operation disturbances.

5.3 Model Application

Investigating the best LSTM architectures are initially tested various alternatives presented in the table below. Based on this, we obtain the knowledge that the architecture of model A, comprising of two LSTM layers with 100 neurons each, stochastic gradient descent optimizer, ReLU activation function and mean squared error as loss metric, perform better than the other experimental models. In addition, model G applying the Adam optimizer and the H1.3. applying a LeakyRelu (alpha=0.2) activation function perform also relatively well. The table below displays the first attempt to determine the orientation towards the decision of the appropriate model architecture given the port data characteristics.

model	modelA	modelB	modelC	modelD	modelE	modelF	modelG	modelG.1	modelH	modelH.1	modelH.1.3	modelH.1.3.1
epochs	20	20	20	20	20	20	20	100	20	100	200	20
activation	ReLU	ReLU	ReLU	ReLU	tan	LeakyReLU	ReLU	ReLU	LeakyReLU	LeakyReLU	LeakyReLU	LeakyReLU, softmax
optimizer	SGD	SGD	SGD	SGD	SGD	SGD	Adam	Adam	Adam	Adam	Adam	Adam
time (sec)	315,8	617,5	412,9	623,46	621,29	377,56	353,4	1493,71	310,32	1457,86	5111,09	432,8
LSTM layers	2	3	2	2	2	2	2	2	2	2	2	2
neurons	100	100	128	100	100	100	100	100	100	100	100	100
loss	mse	mse	mse	mse	mse	mse	mse	mse	mse	mse	mse	CCE
Dense	1	1	1	1	1	1	1	1	1	1	1	1
Dropout	0	0	0	2 (0.2)	0	0	0	0	0	0	0	0

Figure 26: LSTM architecture investigation

In this direction, we expand the investigation by forming three models trained for various epochs as presented in the table below. The group of three models is applied in every state for all four seasons resulting in training a total of 96 models. The table below, displays indicatively the group of models applied for biggest commercial port of Texas for the autumn season.

Observing the model results we obtain the following insights. Firstly, different port regions have unique climatic characteristics which are identified by the LSTM models during the training steps. Thus, we consider meaningful to present the best case and the least best case encountered while training the three LSTM models for the various ports and seasons. Indeed we notice that for example the commercial port in the State of Georgia has a more stable climatic conditions, meaning less combinations of extreme weather events, throughout the years and this is translated into higher accuracy in the training sets presented in the figures below. Specifically the aforementioned port is the only case where the three LSTM models perform

	te_AUTUMN		
	model1	model2	model3
epochs	50	50	100
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	4.952	3.491	3.194
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0

Figure 27: Chosen LSTM model architectures for Texas in Autumn

relatively well in the training set for more than one season.

Alternatively, for the commercial ports of Florida, given the rich landscape of climatic conditions¹⁴ the applied models fail to achieve accuracy higher than 46,6% even on the training set for all seasons. A more analytical view of the Florida models performance is presented in the table below.

As general remarks, we note that consistently for summer season all three models achieve high accuracy in the training set avoiding overfitting for the majority of States (all except Florida ports). Further, the complexity of weather events in winter season in all States results to low performance of all LSTM models. Spring and Autumn present a mix of performance results as anticipated from the inherent characteristics of those seasons. In terms of model comparison in most cases model2 and model3 perform better than model1 and this is an expected outcome since both models use Adam instead of SGD optimizer which in bigger data sets performs better by using adaptive learning rates, incorporating momentum accelerating convergence and applying a bias correction mechanism. All models performance details are presented analytically in Appendix B.

¹⁴<https://floridaclimateinstitute.org/docs/climatebook/Ch20-Collins.pdf>

	geo_AUTUMN		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.526	1.903	3.619
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	55,2%	50,6%	52,1%
RMSE	15,7%	15,4%	15,1%

	geo_SPRING		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.984	1.261	6.869
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	59,1%	60,9%	60,7%
RMSE	18,8%	18,7%	18,0%

	geo_SUMMER		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.704	1.724	7.227
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	91,6%	87,4%	74,9%
RMSE	29,7%	28,6%	26,4%

	geo_WINTER		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.656	1.613	5.763
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	18,8%	20,9%	19,9%
RMSE	14,6%	14,3%	13,9%

Figure 28: Performance of LSTM models for Georgia ports for every season

	fl_AUTUMN		
	model1	model2	model3
epochs	50	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.377	4.060	5.916
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	27,2%	25,8%	27,9%
RMSE	22,0%	20,9%	21,2%

	fl_SPRING		
	model1	model2	model3
epochs	80	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	3.419	6.886	7.076
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	35,4%	38,9%	40,6%
RMSE	17,5%	17,1%	17,1%

	fl_SUMMER		
	model1	model2	model3
epochs	100	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	6.000	6.436	6.992
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	40,3%	46,6%	45,4%
RMSE	29,4%	28,8%	28,6%

	fl_WINTER		
	model1	model2	model3
epochs	50	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.455	6.283	5.975
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	27,6%	33,5%	32,7%
RMSE	16,9%	16,4%	16,6%

Figure 29: Performance of LSTM models for Florida for every season

5.4 Model evaluation

In the process of evaluating the predicting ability of trained models is firstly investigated the performance over the test set and secondly a forecast for the next five days beyond the test set. Among the 96 models, and given the challenging task of accurate weather prediction a threshold of 55% accuracy over the training set is applied. The results over the test set for those models are analytically presented in the Appendix B. Indicatively, in the two figures below is presented the accuracy percentage of model3 in winter California and in summer South Carolina. More analytically, the graph on the left displays the loss minimization over the training epochs and the respective results over the test set for each specific port/season. We note that since event types are different per region and per season y-axis has max value the maximum number of event one-hot-encoding while it is crucial to mention event types is a discrete variable. For example, in Figure 29 on the left, from 1995-2022 California has registered 175 different event type combinations. On the x-axis are displayed the first 20 days of the data set since after 20 days the LSTM weather prediction model loses it's predictive power because the occurrence of weather events in the far future becomes even more uncertain. Regarding the graph, for day 1 the model predicted the true value of test set, denoted by a red dot. Subsequently, for the second day the model predicted a different value from the true hence this difference is denoted by a gray dot (the true) falling not in the same position as the blue dot (model prediction). In any case that the true does no equal the predicted value, the proximity of the two on the graph does not signify a gradually better model since the event types are encoded as distinct incidents and all 175 represent weather events that jeopardize equally the resilience of the port. In this context, the visualization of predicted and true values on the test set in this specific case indicates that the model was able to predict correctly the 30% of the 20 days. The best accuracy for all ports and seasons over the test set is 50% and the worse is 10% with an average predictive ability of 27%.

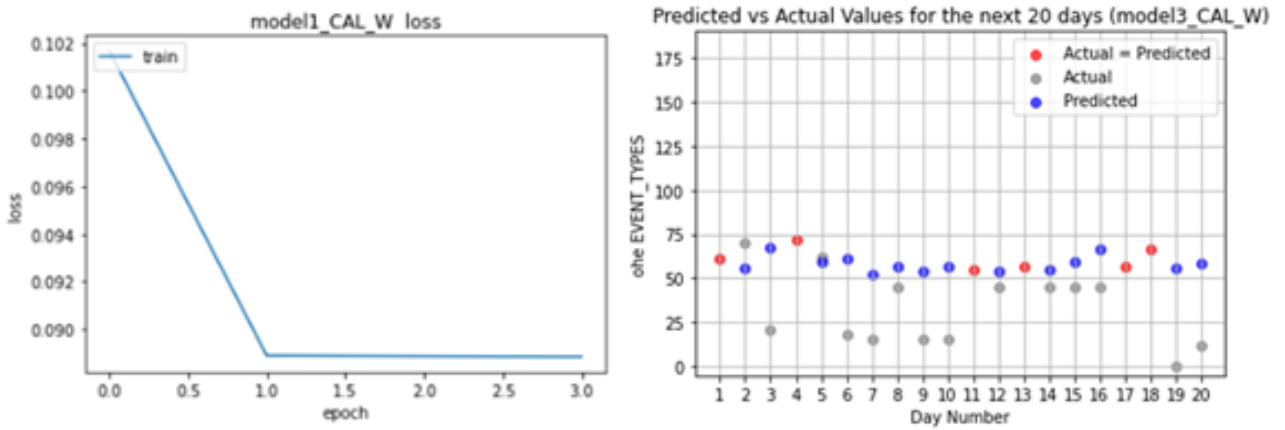


Figure 30: On the left, model loss stops decreasing after the first period, on the right are presented the predictions over the first 20 days on the test set of winter California port.

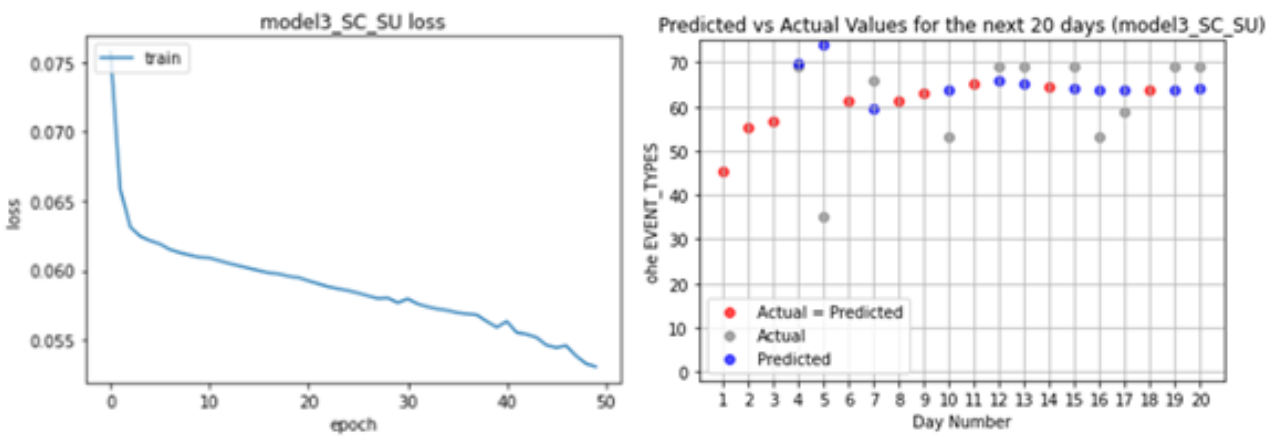


Figure 31: On the left, model loss decrease over the 50 training epochs, on the right are presented the predictions over the first 20 days on the test set of summer South Carolina port.

Finally, the forecast for the next 5 days beyond the test set are presented on the figure below. Interpreting this result, signifies that an extra assumption. More analytically, we assume that the prediction for the next 5 days¹⁵ for example in winter California ports is related to the previous winter months from the previous years starting from 1995. However, the magnitude of data allows us to claim that weather patterns related to climate change have been incorporated in the past winter days. Although the aforementioned affirmation is rational and true, we note that the prediction of the next 5 winter days in California, is related to the way LSTM is structured to identify weather patterns in the past winter seasons but is also related in fact to the specific year's registered events independent of the season. All model forecasts for the next five days are presented analytically in the Appendix B.



Figure 32: Indicative table of predictions beyond the test set

¹⁵The forecast for winter California ports refers to the days: 1-12-2023, 2-12-2023, 3-12-2023, 4-12-2023, 5-12-2023, and accordingly for the other ports/seasons

6 Conclusion

The ambition of the present Thesis is to investigate the link between climate change related extreme weather events to port resilience. The proposed methodology consists in building neural network models able to provide reliable weather predictions. Discussing both man-made and natural factors, the thesis introduces the idea of climate change, exploring into the literature to examine how extreme weather events affect port operations. The economic effects of weather-related port damages and disruptions are also examined along with potential adaptation strategies that might make ports less susceptible to the impacts of climate change. Furthermore, presenting the neural network applications used in weather prediction, we develop three LSTM models which are trained for eight different port regions and for four seasons each. Models with higher accuracy are used to create forecasts over the test and five days beyond the test set. Based on the model predictions, we can summarize some significant remarks in the following three points:

1. Models that seem to be robust in some ports are less robust in others. This is an anticipated result, since different states that accommodate big commercial ports, face different climatic conditions due to their geographic location. For example, the same three models that perform well in the two big ports of Georgia, achieve low performance in the case of Florida's commercial port. This fact establish the need for different model architectures built taking into account the specific climatic conditions of each port.
2. Models perform better on summer seasons for the majority of the ports in comparison to the other seasons, due to the facts that summer months tend to have less intense and less frequent weather incidents.
3. The best achieved accuracy from all 96 trained models is 50% over the test set, however this is an expected result since raw data include registered weather event types from different radars around each state. We consider of at most importance that in order to train a weather forecasting LSTM's able to achieve high accuracy is needed to feed them with historical event registrations taken from the same radar station close to the port.

Certainly, the present Thesis sets a well-established ground with respect to a neural network application for weather prediction in port regions. Nevertheless, regarding our future work, we endeavor to extend this application linking it to an elaborate analysis that prices the potential economic damage of the examined ports due to increase in frequency and intensity extreme weather events. This extension, could investigate a pivotal concern for port authorities, governments, insurance companies and ship owners that will be a significant addition to the ongoing research for academic purposes and real world business needs.

7 Appendix A

```
# Dependencies

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
from sklearn.preprocessing import MinMaxScaler
import keras
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras.models import Sequential
import pandas as pd
from keras.layers import Dense
from keras.layers import LSTM
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.losses import MeanSquaredError
from tensorflow.keras.metrics import RootMeanSquaredError
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import optimizers
from sklearn.metrics import r2_score, explained_variance_score
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.optimizers import SGD
import time
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
```

```

# Import raw data from directory

data22 = pd.read_csv("storm_data2022.csv")
data21 = pd.read_csv("storm_data2021.csv")
data20 = pd.read_csv("storm_data2020.csv")
data19 = pd.read_csv("storm_data2019.csv")
data18 = pd.read_csv("storm_data2018.csv")
data17 = pd.read_csv("storm_data2017.csv")
data16 = pd.read_csv("storm_data2016.csv")
data15 = pd.read_csv("storm_data2015.csv")
data14 = pd.read_csv("storm_data2014.csv")
data13 = pd.read_csv("storm_data2013.csv")
data12 = pd.read_csv("storm_data2012.csv")
data11 = pd.read_csv("storm_data2011.csv")
data10 = pd.read_csv("storm_data2010.csv")
data09 = pd.read_csv("storm_data2009.csv")
data08 = pd.read_csv("storm_data2008.csv")
data07 = pd.read_csv("storm_data2007.csv")
data06 = pd.read_csv("storm_data2006.csv")
data05 = pd.read_csv("storm_data2005.csv")
data04 = pd.read_csv("storm_data2004.csv")
data03 = pd.read_csv("storm_data2003.csv")
data02 = pd.read_csv("storm_data2002.csv")
data01 = pd.read_csv("storm_data2001.csv")
data00 = pd.read_csv("storm_data2000.csv")
data99 = pd.read_csv("storm_data1999.csv")
data98 = pd.read_csv("storm_data1998.csv")
data97 = pd.read_csv("storm_data1997.csv")
data96 = pd.read_csv("storm_data1996.csv")
data95 = pd.read_csv("storm_data1995.csv")

# Keep only columns of interest

data22 = data22[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]

```

```

data21 = data21[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data20 = data20[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data19=data19[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data18=data18[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data17=data17[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data16=data16[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data15=data15[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data14=data14[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data13=data13[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data12=data12[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data11=data11[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data10=data10[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data09=data09[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]

```

```
data08=data08[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data07=data07[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data06=data06[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data05=data05[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data04=data04[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data03=data03[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data02=data02[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data01=data01[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data00=data00[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data99=data99[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data98=data98[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data97=data97[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
data96=data96[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]
```



```

data95=data95[['YEAR', 'MONTH_NAME', 'BEGIN_DAY', 'BEGIN_DATE_TIME', '
    END_DATE_TIME',
    'STATE', 'EVENT_TYPE']]

#Merge all data frames in a single data frame

all_data = [data22, data21, data20, data19, data18, data17, data16, data15,
    data14, data13, data12,
    data11, data10, data09, data08, data07, data06, data05, data04,
    data03, data02, data01,
    data00, data99, data98, data97, data96, data95]

DATA = pd.concat(all_data)

# Replace column names

DATA.rename(columns={"MONTH_NAME": "MONTH"}, inplace=True)
DATA.rename(columns={"BEGIN_DAY": "DAY"}, inplace=True)
DATA.rename(columns={"BEGIN_DATE_TIME": "BEGIN_TIME"}, inplace=True)
DATA.rename(columns={"END_DATE_TIME": "END_TIME"}, inplace=True)

# Replace month names

m = {'January':1, 'February':2, 'March':3, 'April':4, 'May':5, 'June':6, 'July
    ':7, 'August':8, 'September':9, 'October':10, 'November':11, 'December':12}
DATA.MONTH = DATA.MONTH.map(m)

# Keep only extreme weather events that have strong impact on port
    resilience

DATA = DATA[DATA.EVENT_TYPE.isin(['High_Wind', 'Heavy_Rain', 'Winter_Storm', '
    Dense_Fog', 'Thunderstorm_Wind',
    'Marine_Thunderstorm_Wind', 'Tornado', 'Strong_Wind', 'Flood', 'Marine_High
    _Wind', 'Dust_Storm', 'Extreme_Cold/Wind_Chill', 'High_Surf', 'Flash
    _Flood',
    'Excessive_Heat', 'Ice_Storm', 'Coastal_Flood', 'Tsunami', 'Marine_Hail'
    , 'Marine_Strong_Wind', 'Tropical_Storm', 'Marine_Dense_Fog', '
    Marine_Tropical_Storm'
))]

```

```
# Replace event names
```

```
e = {'High_Wind': 'A',  
     'Heavy_Rain': 'B',  
     'Winter_Storm': 'C',  
     'Dense_Fog': 'D',  
     'Thunderstorm_Wind': 'E',  
     'Marine_Thunderstorm_Wind': 'F',  
     'Tornado': 'G',  
     'Strong_Wind': 'H',  
     'Flood': 'I',  
     'Marine_High_Wind': 'J',  
     'Dust_Storm': 'K',  
     'Extreme_Cold/Wind_Chill': 'L',  
     'High_Surf': 'M',  
     'Flash_Flood': 'N',  
     'Excessive_Heat': 'O',  
     'Ice_Storm': 'P',  
     'Coastal_Flood': 'Q',  
     'Tsunami': 'R',  
     'Marine_Hail': 'S',  
     'Marine_Strong_Wind': 'T',  
     'Tropical_Storm': 'U',  
     'Marine_Dense_Fog': 'V',  
     'Marine_Tropical_Storm': 'W'}
```

```
DATA.EVENT_TYPE= DATA.EVENT_TYPE.map(e)
```

```
DATA_1 = DATA
```

```
# Maintain only rows for the states with big commercial ports
```

```
STATES_ports = ['CALIFORNIA', 'NEW_YORK', 'GEORGIA', 'WASHINGTON', 'VIRGINIA',  
               'TEXAS', 'SOUTH_CAROLINA', 'FLORIDA']
```

```
DATA_2 = DATA_1[DATA_1.STATE.isin(['CALIFORNIA', 'NEW_YORK', 'GEORGIA', '  
    WASHINGTON', 'VIRGINIA', 'TEXAS', 'SOUTH_CAROLINA', 'FLORIDA'])]
```

```
#Check for NA and null values
```

```

DATA_2.isnull().sum()

# Set format for every column

DATA_2.EVENT_TYPE.astype(str)
DATA_2 = DATA_2.astype({'YEAR':'int32', 'MONTH':'int32', 'DAY':'int32', 'STATE
    ':'category', 'EVENT_TYPE':'category'})

# Calculate duration

DATA_2['BEGIN_TIME'] = pd.to_datetime(DATA_2['BEGIN_TIME'])
DATA_2['END_TIME'] = pd.to_datetime(DATA_2['END_TIME'])
DATA_2['DURATION'] = DATA_2['END_TIME'] - DATA_2['BEGIN_TIME']
DATA_2['BEGIN_TIME'] = DATA_2['BEGIN_TIME'].dt.time
DATA_2['END_TIME'] = DATA_2['END_TIME'].dt.time

DATA_3 = DATA_2

# Filter dataframe to keep only events that have DURATION greater than 30
    min

DATA_4 = DATA_3[DATA_3['DURATION'] >= '0_days_0:30:00']

# Data Preprocessing for CALIFORNIA and forecasting models

CALIFORNIA_all = DATA_4[DATA_4["STATE"]=="CALIFORNIA"]
CALIFORNIA_all.sort_values(by=['YEAR', 'MONTH', 'DAY'])
CALIFORNIA_all_s1 = CALIFORNIA_all

#Create new column date combining YEAR,MONTH,DAY

CALIFORNIA_all_s1['date'] = pd.to_datetime(CALIFORNIA_all_s1[['YEAR', 'MONTH',
    'DAY']])

# Set column date as index

CALIFORNIA_all_s1.set_index('date', inplace=True)
CALIFORNIA_all_s2 = CALIFORNIA_all_s1

```

```

# Keep only the two columns

CALIFORNIA_all_s3 = CALIFORNIA_all_s2[['EVENT_TYPE', 'DURATION']]

# Group data frame by day and sum the duration of the events

CALIFORNIA_event_type = CALIFORNIA_all_s3.groupby(['date'], as_index =True).
    agg({'EVENT_TYPE':''.join})
CALIFORNIA_duration = CALIFORNIA_all_s3.groupby(['date'], as_index =True).agg
    ({'DURATION':'sum'})
CALIFORNIA_all_s4 = pd.merge_asof(CALIFORNIA_event_type, CALIFORNIA_duration,
    on='date', direction='forward')

# Maintain unique events and put it in a list in the EVENT_TYPE column

CALIFORNIA_all_s4['EVENT_TYPE'] = CALIFORNIA_all_s4['EVENT_TYPE'].str.replace(
    ',','')
CALIFORNIA_all_s4['EVENT_TYPE'] = CALIFORNIA_all_s4['EVENT_TYPE'].apply(list
    ,1)
CALIFORNIA_all_s4['EVENT_TYPE'] = CALIFORNIA_all_s4.apply(lambda row: list(
    set(row['EVENT_TYPE'])), axis=1)
CALIFORNIA_all_s5 = CALIFORNIA_all_s4

# Sort data frame by date

CALIFORNIA_all_s5_sorted = CALIFORNIA_all_s5.sort_values(by='date')

# Fill missing values
cal = CALIFORNIA_all_s5_sorted
cal1 = cal.set_index('date')
cal_all_days = pd.date_range(cal1.index.min(), cal1.index.max(), freq='D')
cal2 = cal1.reindex(cal_all_days)
cal2.EVENT_TYPE = cal2.EVENT_TYPE.fillna('Z').apply(list)
cal2.DURATION = cal2.DURATION.fillna('0days_24:00:00')

# Filter data frame to keep only events that have daily aggregated DURATION
    of the events less than 24 hours

```

```

cal2_=cal2
cal2_['DURATION'] = cal2_['DURATION'].mask(cal2_['DURATION'] > '1_days_0:00:00', '1_days_0:00:00')

# Create new column "total_sec" and drop DURATION column

cal2_['total_sec'] = cal2_.DURATION / pd.Timedelta(seconds=1)
cal2_ = cal2_.drop(columns=['DURATION'])

# Split initial filled State data frame into 4 seasons

cal_split_seasons = cal2_

s = cal_split_seasons.reset_index()

cal_winter = s[s['index'].dt.month.isin([12,1,2])]
cal_spring = s[s['index'].dt.month.isin([3,4,5])]
cal_summer = s[s['index'].dt.month.isin([6,7,8])]
cal_automn = s[s['index'].dt.month.isin([9,10,11])]

# Set index column as index of the data frame and impose integer type in the
total_sec column

cal_automn.set_index('index', inplace=True)
cal_winter.set_index('index', inplace=True)
cal_spring.set_index('index', inplace=True)
cal_summer.set_index('index', inplace=True)

cal_automn.total_sec = cal_automn.total_sec.astype('int')
cal_winter.total_sec = cal_winter.total_sec.astype('int')
cal_spring.total_sec = cal_spring.total_sec.astype('int')
cal_summer.total_sec = cal_summer.total_sec.astype('int')

# Transform column EVENT_TYPE in strings

cal_automn.EVENT_TYPE = cal_automn.apply(lambda row : "".join(row['EVENT_TYPE',
]), axis=1)
cal_winter.EVENT_TYPE = cal_winter.apply(lambda row : "".join(row['EVENT_TYPE',
]), axis=1)

```

```

cal_spring.EVENT_TYPE = cal_spring.apply(lambda row : "".join(row['EVENT_TYPE',
    ]), axis=1)
cal_summer.EVENT_TYPE = cal_summer.apply(lambda row : "".join(row['EVENT_TYPE',
    ]), axis=1)

cal_automn.EVENT_TYPE.fillna(value='␣',inplace=True)
cal_winter.EVENT_TYPE.fillna(value='␣',inplace=True)
cal_spring.EVENT_TYPE.fillna(value='␣',inplace=True)
cal_summer.EVENT_TYPE.fillna(value='␣',inplace=True)

# Save all unique event 1st encoding EVENT_TYPES into a new data frame for
    every season

cal_e_t_comb_AUTUMN = cal_automn.EVENT_TYPE.unique()
cal_e_t_comb_WINTER = cal_winter.EVENT_TYPE.unique()
cal_e_t_comb_SPRING= cal_spring.EVENT_TYPE.unique()
cal_e_t_comb_SUMMER = cal_summer.EVENT_TYPE.unique()

# Create encodings only for the EVENT_TYPE column using a label encoder

encoder = LabelEncoder()

cal_automn['EVENT_TYPE'] = encoder.fit_transform(cal_automn[['EVENT_TYPE']])
cal_winter['EVENT_TYPE'] = encoder.fit_transform(cal_winter[['EVENT_TYPE']])
cal_spring['EVENT_TYPE'] = encoder.fit_transform(cal_spring[['EVENT_TYPE']])
cal_summer['EVENT_TYPE'] = encoder.fit_transform(cal_summer[['EVENT_TYPE']])

# Array of the ohe EVENT_TYPES

ohe_cal_e_t_comb_AUTUMN = cal_automn.EVENT_TYPE.unique()
# Array of the ohe EVENT_TYPES
ohe_cal_e_t_comb_AUTUMN = cal_automn.EVENT_TYPE.unique()
ohe_cal_e_t_comb_SPRING = cal_spring.EVENT_TYPE.unique()
ohe_cal_e_t_comb_SUMMER = cal_summer.EVENT_TYPE.unique()

# Scale data because seconds and event types have big numerical difference

scaler = MinMaxScaler()

```

```

cal_automn[cal_automn.columns] = scaler.fit_transform(cal_automn[cal_automn.
    columns])
cal_winter[cal_winter.columns] = scaler.fit_transform(cal_winter[cal_winter.
    columns])
cal_spring[cal_spring.columns] = scaler.fit_transform(cal_spring[cal_spring.
    columns])
cal_summer[cal_summer.columns] = scaler.fit_transform(cal_summer[cal_summer.
    columns])

# Transform the four dataframes to numpy array

cal_automnA = cal_automn.to_numpy()
cal_winterA = cal_winter.to_numpy()
cal_springA = cal_spring.to_numpy()
cal_summerA = cal_summer.to_numpy()

# Split to train and test set

train_cal_AUTUMN, test_cal_AUTUMN = train_test_split(cal_automnA, test_size
    =0.10, shuffle=False)
train_cal_WINTER, test_cal_WINTER = train_test_split(cal_winterA, test_size
    =0.10, shuffle=False)
train_cal_SPRING, test_cal_SPRING = train_test_split(cal_springA, test_size
    =0.10, shuffle=False)
train_cal_SUMMER, test_cal_SUMMER = train_test_split(cal_summerA, test_size
    =0.10, shuffle=False)

n_input = 5
n_features = 2

# Final transformation to LSTM compatible input format

generatorTrain_cal_AUTUMN = TimeseriesGenerator(train_cal_AUTUMN,
    train_cal_AUTUMN, length=n_input, batch_size=1)
generatorTest_cal_AUTUMN = TimeseriesGenerator(test_cal_AUTUMN,
    test_cal_AUTUMN, length=n_input, batch_size=1)

generatorTrain_cal_WINTER = TimeseriesGenerator(train_cal_WINTER,
    train_cal_WINTER, length=n_input, batch_size=1)

```

```

generatorTest_cal_WINTER = TimeseriesGenerator(test_cal_WINTER,
    test_cal_WINTER, length=n_input, batch_size=1)

generatorTrain_cal_SPRING = TimeseriesGenerator(train_cal_SPRING,
    train_cal_SPRING, length=n_input, batch_size=1)
generatorTest_cal_SPRING = TimeseriesGenerator(test_cal_SPRING,
    test_cal_SPRING, length=n_input, batch_size=1)

generatorTrain_cal_SUMMER = TimeseriesGenerator(train_cal_SUMMER,
    train_cal_SUMMER, length=n_input, batch_size=1)
generatorTest_cal_SUMMER = TimeseriesGenerator(test_cal_SUMMER,
    test_cal_SUMMER, length=n_input, batch_size=1)

# Model architectures

# model1 for AUTUMN

model1_CAL_A = Sequential()
model1_CAL_A.add(LSTM(100, activation='relu', input_shape=(n_input, n_features
    ), return_sequences=True))
model1_CAL_A.add(LSTM(100, activation='relu'))
model1_CAL_A.add(Dense(2))

model1_CAL_A.compile(loss='mse', optimizer = SGD(), metrics=['accuracy', '
    RootMeanSquaredError'])
callback1_CAL_A = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time1_CAL_A = time.time()
history1_CAL_A = model1_CAL_A.fit(generatorTrain_cal_AUTUMN, batch_size=100,
    epochs=200, shuffle=False, verbose=2) #batch_size=72
model1_CAL_A.save("model1_CAL_A.h5")
end_time1_CAL_A= time.time()

total_time1_CAL_A = end_time1_CAL_A - start_time1_CAL_A
print("Total training time:", total_time1_CAL_A, "seconds")

#Display loss

plt.plot(history1_CAL_A.history['loss'])

```



```

plt.title('model1_CAL_A_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper_left')
plt.show()

# model1 for WINTER

model1_CAL_W = Sequential()
model1_CAL_W .add(LSTM(100, activation='relu', input_shape=(n_input,
    n_features), return_sequences=True))
model1_CAL_W .add(LSTM(100, activation='relu'))
model1_CAL_W .add(Dense(2))

model1_CAL_W .compile(loss='mse', optimizer = SGD(), metrics=['accuracy', '
    RootMeanSquaredError'])
callback1_CAL_W = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time1_CAL_W = time.time()
history1_CAL_W = model1_CAL_W .fit(generatorTrain_cal_WINTER, batch_size=100,
    epochs=4, shuffle=False, verbose=2) #batch_size=72 epochs=200

model1_CAL_W.save("model1_CAL_W.h5")

end_time1_CAL_W = time.time()

total_time1_CAL_W = end_time1_CAL_W - start_time1_CAL_W
print("Total_training_time:", total_time1_CAL_W , "seconds")

#Display loss

plt.plot(history1_CAL_W .history['loss'])
plt.title('model1_CAL_W_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper_left')
plt.show()

# model1 for SPRING

```

```

model1_CAL_S = Sequential()
model1_CAL_S .add(LSTM(100, activation='relu', input_shape=(n_input,
    n_features), return_sequences=True))
model1_CAL_S .add(LSTM(100, activation='relu'))
model1_CAL_S .add(Dense(2))

model1_CAL_S .compile(loss='mse', optimizer = SGD(), metrics=['accuracy', '
    RootMeanSquaredError'])
callback1_CAL_S = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time1_CAL_S = time.time()
history1_CAL_S = model1_CAL_S .fit(generatorTrain_cal_SPRING, batch_size=100,
    epochs=4, shuffle=False, verbose=2) #batch_size=72

model1_CAL_S.save("model1_CAL_S.h5")
end_time1_CAL_S = time.time()

total_time1_CAL_S = end_time1_CAL_S - start_time1_CAL_S
print("Total training time:", total_time1_CAL_S , "seconds")

#Display loss

plt.plot(history1_CAL_S .history['loss'])
plt.title('model1_CAL_S loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# model1 for SUMMER

model1_CAL_SU = Sequential()
model1_CAL_SU .add(LSTM(100, activation='relu', input_shape=(n_input,
    n_features), return_sequences=True))
model1_CAL_SU .add(LSTM(100, activation='relu'))
model1_CAL_SU .add(Dense(2))

```

```

model1_CAL_SU .compile(loss='mse', optimizer = SGD(), metrics=['accuracy', '
    RootMeanSquaredError'])
callback1_CAL_SU = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time1_CAL_SU = time.time()
history1_CAL_SU = model1_CAL_SU .fit(generatorTrain_cal_SUMMER, batch_size
    =100, epochs=50, shuffle=False, verbose=2) #batch_size=72
model1_CAL_SU.save("model1_CAL_SU.h5")
end_time1_CAL_SU = time.time()

total_time1_CAL_SU = end_time1_CAL_SU - start_time1_CAL_SU
print("Total training time:", total_time1_CAL_SU , "seconds")

#Display loss

plt.plot(history1_CAL_SU .history['loss'])
plt.title('model1_CAL_SU loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Model 2 CAL AUTUMN

model2 = Sequential()
model2.add(LSTM(100, activation='relu', input_shape=(n_input, n_features),
    return_sequences=True))
model2.add(LSTM(100, activation='relu'))
model2.add(Dense(2))
model2.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
    RootMeanSquaredError'])
callback2 = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time2 = time.time()
history2 = model2.fit(generatorTrain_cal_AUTUMN, batch_size=100, epochs=150,
    shuffle=False, verbose=2) #batch_size=72
model2.save("model2_CAL_A.h5")
end_time2 = time.time()

```

```

total_time2 = end_time2 - start_time2
print("Total training time:", total_time2, "seconds")

#Display loss

plt.plot(history2.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# Model 2 CAL WINTER

model2_CAL_W = Sequential()
model2_CAL_W.add(LSTM(100, activation='relu', input_shape=(n_input, n_features
), return_sequences=True))
model2_CAL_W.add(LSTM(100, activation='relu'))
model2_CAL_W.add(Dense(2))
model2_CAL_W.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
RootMeanSquaredError'])
callback2_CAL_W = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time2_CAL_W = time.time()
history2_CAL_W = model2_CAL_W.fit(generatorTrain_cal_WINTER, batch_size=100,
epochs=150, shuffle=False, verbose=2) #batch_size=72
model2_CAL_W.save("model2_CAL_W.h5")
end_time2_CAL_W = time.time()

total_time2_CAL_W = end_time2_CAL_W - start_time2_CAL_W
print("Total training time:", total_time2_CAL_W, "seconds")

#Display loss

plt.plot(history2_CAL_W.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')

```

```

plt.show()

# Model 2 CAL SPRING

model2_CAL_S = Sequential()
model2_CAL_S.add(LSTM(100, activation='relu', input_shape=(n_input, n_features
    ), return_sequences=True))
model2_CAL_S.add(LSTM(100, activation='relu'))
model2_CAL_S.add(Dense(2))
model2_CAL_S.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
    RootMeanSquaredError'])
callback2_CAL_S = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time2_CAL_S = time.time()
history2_CAL_S = model2_CAL_S.fit(generatorTrain_cal_SPRING, batch_size=100,
    epochs=150, shuffle=False, verbose=2)
model2_CAL_S.save("model2_CAL_S.h5")
end_time2_CAL_S = time.time()

total_time2_CAL_S = end_time2_CAL_S - start_time2_CAL_S
print("Total training time:", total_time2_CAL_S, "seconds")

#Display loss

plt.plot(history2_CAL_S.history['loss'])
plt.title('model2_CAL_S_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper_left')
plt.show()

# Model 2 CAL SUMMER

model2_CAL_SU = Sequential()
model2_CAL_SU.add(LSTM(100, activation='relu', input_shape=(n_input,
    n_features), return_sequences=True))
model2_CAL_SU.add(LSTM(100, activation='relu'))
model2_CAL_SU.add(Dense(2))

```

```

model2_CAL_SU.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
    RootMeanSquaredError'])
callback2_CAL_SU = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time2_CAL_SU = time.time()
history2_CAL_SU = model2_CAL_SU.fit(generatorTrain_cal_SUMMER, batch_size=100,
    epochs=150, shuffle=False, verbose=2) #batch_size=72
model2_CAL_SU.save("model2_CAL_SU.h5")
end_time2_CAL_SU = time.time()

total_time2_CAL_SU = end_time2_CAL_SU - start_time2_CAL_SU
print("Total training time:", total_time2_CAL_SU, "seconds")

#Display loss

plt.plot(history2_CAL_SU.history['loss'])
plt.title('model2_CAL_SU_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper_left')
plt.show()

# model 3 CAL AUTUMN

model3 = Sequential()
model3.add(LSTM(100, activation=LeakyReLU(alpha=0.2), input_shape=(n_input,
    n_features), return_sequences=True))
model3.add(LSTM(100, activation=LeakyReLU(alpha=0.2)))
model3.add(Dense(2))
model3.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
    RootMeanSquaredError'])
callback3 = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time3 = time.time()
history3 = model3.fit(generatorTrain_cal_AUTUMN, batch_size=100, epochs=150,
    shuffle=False, verbose=2) #batch_size=72
model3.save("model3_CAL_A.h5")
end_time3 = time.time()

```

```

total_time3 = end_time3 - start_time3
print("Total training time:", total_time3, "seconds")

#Display loss

plt.plot(history3.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'train'], loc='upper left')
plt.show()

# MODEL 3 CAL WINTER

model3_CAL_W = Sequential()
model3_CAL_W.add(LSTM(100, activation=LeakyReLU(alpha=0.2), input_shape=(
    n_input, n_features), return_sequences=True))
model3_CAL_W.add(LSTM(100, activation=LeakyReLU(alpha=0.2)))
model3_CAL_W.add(Dense(2))
model3_CAL_W.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
    RootMeanSquaredError'])
callback3_CAL_W = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time_CAL_W = time.time()
history3_CAL_W = model3_CAL_W.fit(generatorTrain_cal_WINTER, batch_size=100,
    epochs=150, shuffle=False, verbose=2) #batch_size=72
model3_CAL_W.save("model3_CAL_W.h5")
end_time3_CAL_W = time.time()

total_time3_CAL_W = end_time3_CAL_W - start_time3_CAL_W
print("Total training time:", total_time3_CAL_W, "seconds")

#Display loss

plt.plot(history3_CAL_W.history['loss'])
plt.title('model3_CAL_W loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'train'], loc='upper left')

```

```

plt.show()

# MODEL 3 CAL SPRING

model3_CAL_S = Sequential()
model3_CAL_S.add(LSTM(100, activation=LeakyReLU(alpha=0.2), input_shape=(
    n_input, n_features), return_sequences=True))
model3_CAL_S.add(LSTM(100, activation=LeakyReLU(alpha=0.2)))
model3_CAL_S.add(Dense(2))
model3_CAL_S.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
    RootMeanSquaredError'])
callback3_CAL_S = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time3_CAL_S = time.time()
history3_CAL_S = model3_CAL_S.fit(generatorTrain_cal_SPRING, batch_size=100,
    epochs=150, shuffle=False, verbose=2) #batch_size=72
model3_CAL_S.save("model3_CAL_S.h5")
end_time3_CAL_S = time.time()

total_time3_CAL_S = end_time3_CAL_S - start_time3_CAL_S
print("Total training time:", total_time3_CAL_S, "seconds")

#Display loss

plt.plot(history3_CAL_S.history['loss'])
plt.title('model3_CAL_S_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'train'], loc='upper_left')
plt.show()

# MODEL 3 CAL SUMMER

model3_CAL_SU = Sequential()
model3_CAL_SU.add(LSTM(100, activation=LeakyReLU(alpha=0.2), input_shape=(
    n_input, n_features), return_sequences=True))
model3_CAL_SU.add(LSTM(100, activation=LeakyReLU(alpha=0.2)))
model3_CAL_SU.add(Dense(2))

```



```

model3_CAL_SU.compile(loss='mse', optimizer = 'Adam', metrics=['accuracy', '
    RootMeanSquaredError'])
callback3_CAL_SU = keras.callbacks.EarlyStopping(monitor='loss', patience=3)

start_time3_CAL_SU = time.time()
history3_CAL_SU = model3_CAL_SU.fit(generatorTrain_cal_SUMMER, batch_size=100,
    epochs=150, shuffle=False, verbose=2) #batch_size=72
model3_CAL_SU.save("model3_CAL_SU.h5")
end_time3_CAL_SU = time.time()

total_time3_CAL_SU = end_time3_CAL_SU - start_time3_CAL_SU
print("Total training time:", total_time3_CAL_SU, "seconds")

#Display loss

plt.plot(history3_CAL_SU.history['loss'])
plt.title('model3_CAL_SU_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'train'], loc='upper_left')
plt.show()

```

The same preprocessing steps are repeated with the respective reformulations for all other ports, in total for 96 models.

8 Appendix B

The model results according to different architectures are displayed in the tables below.

	cal_WINTER		
	model1	model2	model3
epochs	200	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	4.206	2.722	4.823
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	38,8%	55,2%	57,3%
RMSE	29,8%	19,2%	18,0%

	cal_SPRING		
	model1	model2	model3
epochs	200	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	4.205,63	2.721,67	5.346
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	21,7%	38,2%	40,1%
RMSE	28,7%	22,7%	22,2%

	cal_SUMMER		
	model1	model2	model3
epochs	50	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	4.205,63	4.708	4.527
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	68,9%	76,1%	72,0%
RMSE	24,6%	20,3%	19,7%

	cal_AUTUMN		
	model1	model2	model3
epochs	200	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	4.205,63	2.721,67	3.060
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	38,6%	43,4%	48,6%
RMSE	26,0%	20,3%	20,3%

Figure 33: Model results for California

	ny_AUTUMN		
	model1	model2	model3
epochs	50	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	601,1	3.270	2.368
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	28,5%	33,8%	33,2%
RMSE	17,5%	16,9%	17,0%

	ny_SPRING		
	model1	model2	model3
epochs	50	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	699	3.478	3.730
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	17,0%	23,3%	26,1%
RMSE	18,3%	16,6%	17,1%

	ny_SUMMER		
	model1	model2	model3
epochs	150	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.149	3.393	3.619
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	90,2%	80,9%	74,3%
RMSE	24,0%	21,7%	22,0%

	ny_WINTER		
	model1	model2	model3
epochs	50	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	778,55	2.848	3.034
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	13,9%	20,8%	19,9%
RMSE	18,3%	17,1%	17,7%

Figure 34: Model results for New York

	geo_AUTUMN		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.526	1.903	3.619
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	55,2%	50,6%	52,1%
RMSE	15,7%	15,4%	15,1%

	geo_SPRING		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.984	1.261	6.869
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	59,1%	60,9%	60,7%
RMSE	18,8%	18,7%	18,0%

	geo_SUMMER		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.704	1.724	7.227
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	91,6%	87,4%	74,9%
RMSE	29,7%	28,6%	26,4%

	geo_WINTER		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.656	1.613	5.763
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	18,8%	20,9%	19,9%
RMSE	14,6%	14,3%	13,9%

Figure 35: Model results for Georgia

	wa_AUTUMN		
	model1	model2	model3
epochs	150	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.719	2.528	9.087
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	55,6%	51,9%	47,2%
RMSE	15,0%	14,8%	14,5%

	wa_SPRING		
	model1	model2	model3
epochs	50	150	120
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	3.067	2.880	6.869
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	31,8%	32,2%	20,0%
RMSE	16,9%	16,4%	20,0%

	wa_SUMMER		
	model1	model2	model3
epochs	150	150	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	3.338	2.908	8.076
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	54,5%	52,7%	51,0%
RMSE	15,0%	14,8%	14,4%

	wa_WINTER		
	model1	model2	model3
epochs	50	150	10
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.446	2.925	4.856
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	15,8%	19,8%	16,3%
RMSE	25,3%	23,6%	25,4%

Figure 36: Model results for Washington

	vi_AUTUMN		
	model1	model2	model3
epochs	50	50	100
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	1.627	3.035	8.641
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	18,5%	28,4%	32,6%
RMSE	17,9%	16,9%	16,6%

	vi_SPRING		
	model1	model2	model3
epochs	150	50	100
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	5.593	3.581	4.173
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	31,7%	36,6%	38,7%
RMSE	21,3%	20,3%	19,3%

	vi_SUMMER		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	6.854	3.647	11.533
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	75,4%	75,2%	72,9%
RMSE	28,3%	27,9%	24,4%

	vi_WINTER		
	model1	model2	model3
epochs	50	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	1.711	2.772	4.173
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	15,5%	16,5%	15,9%
RMSE	19,7%	19,2%	19,2%

Figure 37: Model results for Virginia

	te_AUTUMN		
	model1	model2	model3
epochs	50	50	100
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	4.952	3.491	3.194
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	46,2%	51,0%	55,8%
RMSE	26,8%	25,3%	25,5%

	te_SPRING		
	model1	model2	model3
epochs	150	50	80
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	6.057	4.974	7.969
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	51,3%	56,7%	58,0%
RMSE	31,3%	27,8%	27,2%

	te_SUMMER		
	model1	model2	model3
epochs	150	50	150
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	8.918	9.102	11.022
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	86,8%	79,3%	77,7%
RMSE	30,0%	24,3%	22,3%

	te_WINTER		
	model1	model2	model3
epochs	50	50	80
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	3.831	3.155	6.104
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	22,4%	29,2%	28,9%
RMSE	26,5%	25,1%	24,4%

Figure 38: Model results for Texas

	sc_AUTUMN		
	model1	model2	model3
epochs	80	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	3.426	3.700	6.300
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	29,1%	32,0%	29,9%
RMSE	27,0%	24,6%	24,3%

	sc_SPRING		
	model1	model2	model3
epochs	80	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	6.357	5.586	5.460
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	21,8%	28,6%	27,8%
RMSE	28,2%	26,3%	26,4%

	sc_SUMMER		
	model1	model2	model3
epochs	100	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	6.000	6.212	3.715
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	66,8%	73,5%	70,3%
RMSE	24,5%	23,4%	23,0%

	sc_WINTER		
	model1	model2	model3
epochs	50	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	4.278	3.672	6.457
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	40,3%	44,3%	44,6%
RMSE	31,5%	27,7%	27,8%

Figure 39: Model results for South Carolina

	fi_AUTUMN		
	model1	model2	model3
epochs	50	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.377	4.060	5.916
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	27,2%	25,8%	27,9%
RMSE	22,0%	20,9%	21,2%

	fi_SPRING		
	model1	model2	model3
epochs	80	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	3.419	6.886	7.076
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	35,4%	38,9%	40,6%
RMSE	17,5%	17,1%	17,1%

	fi_SUMMER		
	model1	model2	model3
epochs	100	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	6.000	6.436	6.992
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	40,3%	46,6%	45,4%
RMSE	29,4%	28,8%	28,6%

	fi_WINTER		
	model1	model2	model3
epochs	50	50	50
activation	ReLU	ReLU	LeakyReLU
optimizer	SGD	Adam	Adam
time (sec)	2.455	6.283	5.975
LSTM layers	2	2	2
neurons	100	100	100
loss	mse	mse	mse
Dense	1	1	1
Dropout	0	0	0
accuracy	27,6%	33,5%	32,7%
RMSE	16,9%	16,4%	16,6%

Figure 40: Model results for South Florida

In the following section presents the code used to obtain the predictions of the models over the test set and the forecast for the next 5 days for each port and season. We chose to present among 96 models only the results of those that achieve model accuracy over the training set greater than 55%.

```
# Load the saved model
model3_CAL_W = load_model('model3_CAL_W.h5')

n_input = 5
n_features = 2

model3_CAL_W_predictions = []
model3_CAL_W_first_batch = train_cal_AUTUMN[-n_input:]
model3_CAL_W_current_batch = model3_CAL_W_first_batch.reshape((1, n_input,
    n_features))

for i in range(len(test_cal_WINTER)):
    model3_CAL_W_current_pred = model3_CAL_W.predict(
        model3_CAL_W_current_batch)[0] # the event type
    model3_CAL_W_predictions.append(model3_CAL_W_current_pred)
    model3_CAL_W_current_batch_remove_first = model3_CAL_W_current_batch[:,
        1:, :]
    model3_CAL_W_current_batch = np.append(
        model3_CAL_W_current_batch_remove_first, [[model3_CAL_W_current_pred]],
        axis=1)

# Reverse the scaling operation

model3_CAL_W_predictions_actual_scale = scaler.inverse_transform(
    model3_CAL_W_predictions)
model3_CAL_W_test_data_actual_scale = scaler.inverse_transform(test_cal_WINTER
    )

# Line plot to have a clearer view for the true predicted values

plt.plot(model3_CAL_W_predictions_actual_scale[:20,0], c='cyan')
plt.plot(model3_CAL_W_test_data_actual_scale[:20,0], color='gray')
plt.xlabel('Future_consecutive_days')
```

```

plt.ylabel('Actual_Values_vs_20_days_Prediction')
plt.title('Predictions_vs_Actual_Values_within_the_next_20_days_(model3_CAL_W)
        ')
# Set x-axis ticks to integers
plt.xticks(range(1, len(actual_values) + 1))
plt.grid()
plt.show()

# Arrays of predictions and actual values

predictions = model3_CAL_W_predictions_actual_scale[:20, 0]
actual_values = model3_CAL_W_test_data_actual_scale[:20, 0]

# Create an array of day numbers

day_numbers = range(1, len(actual_values) + 1)

# Scatter plot with different colors

for actual, pred, day_num in zip(actual_values, predictions, day_numbers):
    color = 'b'
    if actual == pred:
        plt.scatter(day_num, actual, c='r', alpha=0.7)
    elif day_num in [1,4,11,13,17,18]:
        plt.scatter(day_num, pred, c='r', alpha=0.7)
    else:
        plt.scatter(day_num, actual, c='gray', alpha=0.7)
        plt.scatter(day_num, pred, c=color, alpha=0.7)

plt.xlabel('Day_Number')
plt.ylabel('ohe_EVENT_TYPES')
plt.title('Predicted_vs_Actual_Values_for_the_next_20_days_(model3_CAL_W)')

# Set x-axis ticks to integers

plt.xticks(range(1, len(actual_values) + 1))

# Add legend manually for custom colors
plt.scatter([], [], c='r', label='Actual_=_Predicted', alpha=0.7)

```

```

plt.scatter([], [], c='gray', label='Actual', alpha=0.7)
plt.scatter([], [], c='b', label='Predicted', alpha=0.7)
plt.legend()
plt.grid()
plt.show()

FORECAST_model13_CAL_W_NEXT_5_DAYS = []
FORECAST_model13_CAL_W_NEXT_5_DAYS_first_batch = test_cal_WINTER[-n_input:] #
    take into account 5 previous values
FORECAST_model13_CAL_W_NEXT_5_DAYS_current_batch =
    FORECAST_model13_CAL_W_NEXT_5_DAYS_first_batch.reshape((1, n_input,
        n_features))

for i in range(len(test_cal_WINTER[-5:])):
    FORECAST_model13_CAL_W_NEXT_5_DAYS_current_pred = model13_CAL_W.predict(
        FORECAST_model13_CAL_W_NEXT_5_DAYS_current_batch)[0]
    FORECAST_model13_CAL_W_NEXT_5_DAYS.append(
        FORECAST_model13_CAL_W_NEXT_5_DAYS_current_pred)
    FORECAST_model13_CAL_W_NEXT_5_DAYS_current_batch_remove_first =
        FORECAST_model13_CAL_W_NEXT_5_DAYS_current_batch[:, 1:, :]
    FORECAST_model13_CAL_W_NEXT_5_DAYS_current_batch = np.append(
        FORECAST_model13_CAL_W_NEXT_5_DAYS_current_batch_remove_first, [[
            FORECAST_model13_CAL_W_NEXT_5_DAYS_current_pred]], axis=1)

FORECAST_model13_CAL_W_NEXT_5_DAYS

# Reverse the scaling operation

results = scaler.inverse_transform(FORECAST_model13_CAL_W_NEXT_5_DAYS)

# Try for values [0:4] to obtain all 5 forecasted values

FORECAST_EVENT_TYPE = int(results[:, :][4][0])
FORECAST_DURATION = results[:, :][4][1]

print("The model predicts the next 5 winter days in California will have
    incidents of type {} the will last for {}".format(FORECAST_EVENT_TYPE, str
        (timedelta(seconds=int(FORECAST_DURATION)))))

```

```
# Create dataframe with the first encoding and the ohe

EVENT_TYPE_transformations = pd.DataFrame(data= [cal_e_t_comb_WINTER,
        ohe_cal_e_t_comb_WINTER]).T
EVENT_TYPE_transformations.columns=['EVENT_TYPE_f1', 'EVENT_TYPE_ohe']

# Output of the model prediction

EVENT_TYPE_transformations[EVENT_TYPE_transformations.EVENT_TYPE_ohe ==
        FORECAST_EVENT_TYPE]
```

We repeat the last steps forecasting the next 20 days on the training set and the next 5 days beyond the training set in the future, for models $model1_{CAL_SU}$, $model2_{CAL_SU}$, $model3_{CAL_SU}$. Specifically this regards to all summer NEW YORK models, model1 for autumn Georgia, and all summer and spring Georgia models, model1 for autumn Washington, all summer Virginia models, model1 for winter Texas, model2 and model3 for spring Texas, all summer Texas models and all summer South Carolina models that present the best accuracy in the training set.

The following figures display on the left the monitoring of loss decrease over the training periods and on the right the predictions over first 20 days of the test set.

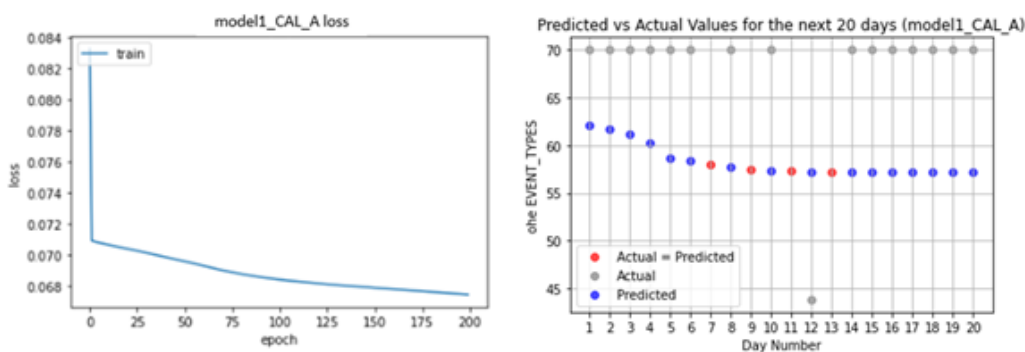


Figure 41: model1 accuracy over the test set for autumn California:20%

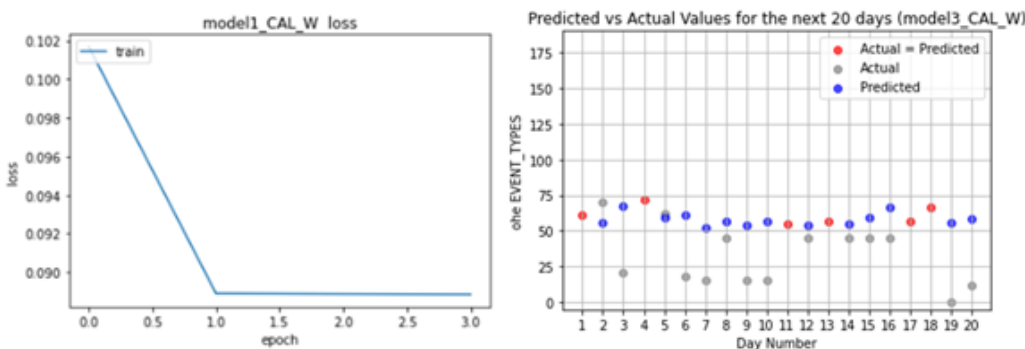


Figure 42: model3 accuracy over the test set for winter California: 30%

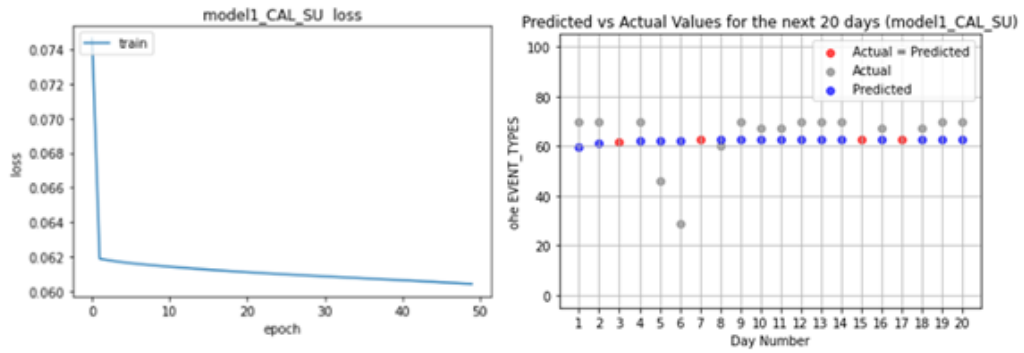


Figure 43: model1 accuracy over the test set for summer California: 20%

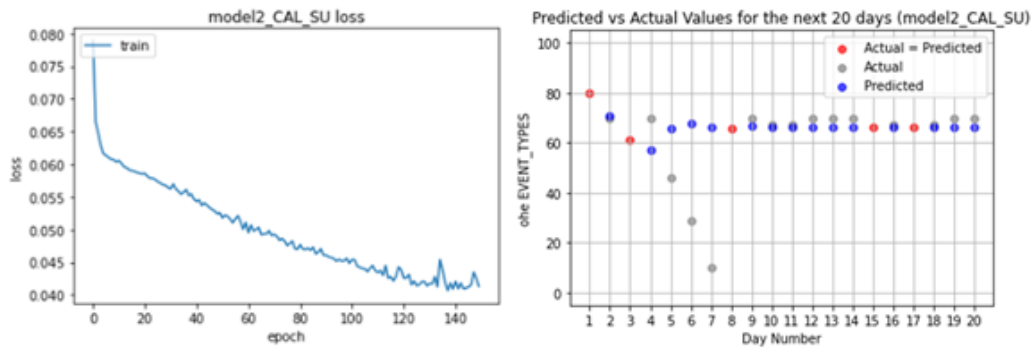


Figure 44: model2 accuracy over the test set for summer California: 25%

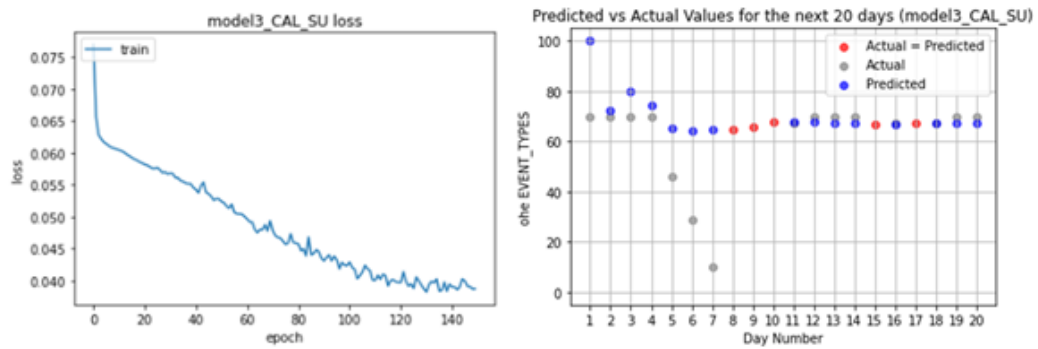


Figure 45: model3 accuracy over the test set for summer California: 25%

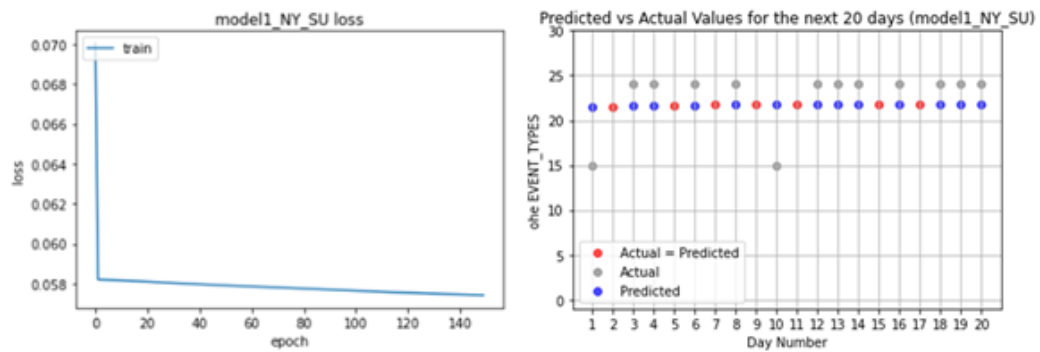


Figure 46: model1 accuracy over the test set for summer New York: 35%

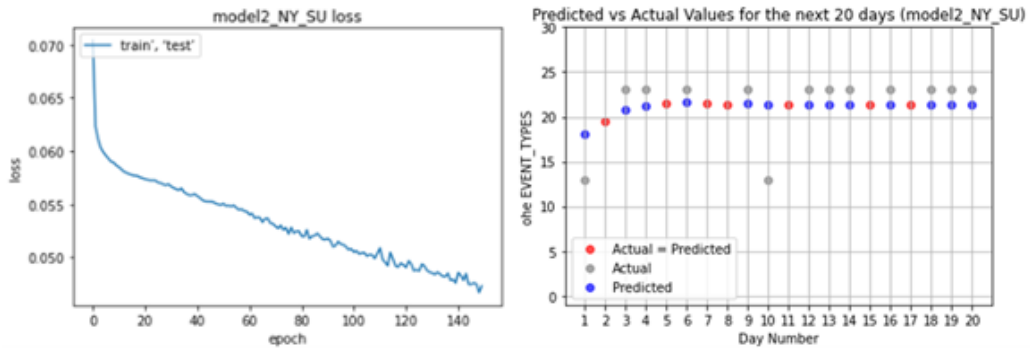


Figure 47: model1 accuracy over the test set for summer New York: 35%

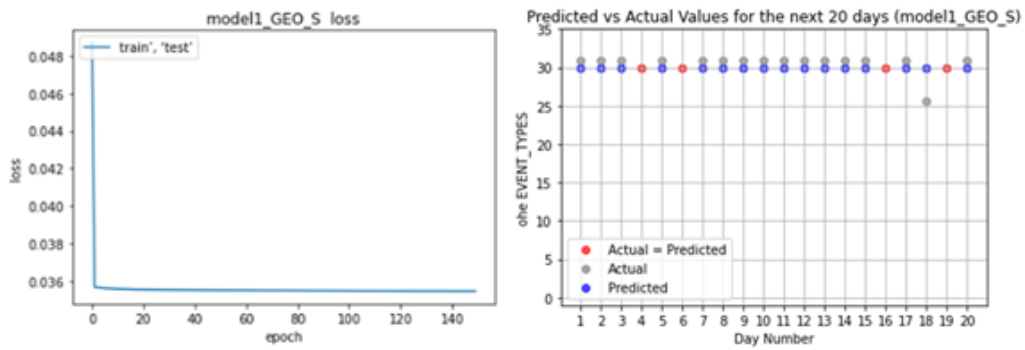


Figure 48: model1 accuracy over the test set for spring Georgia: 20%

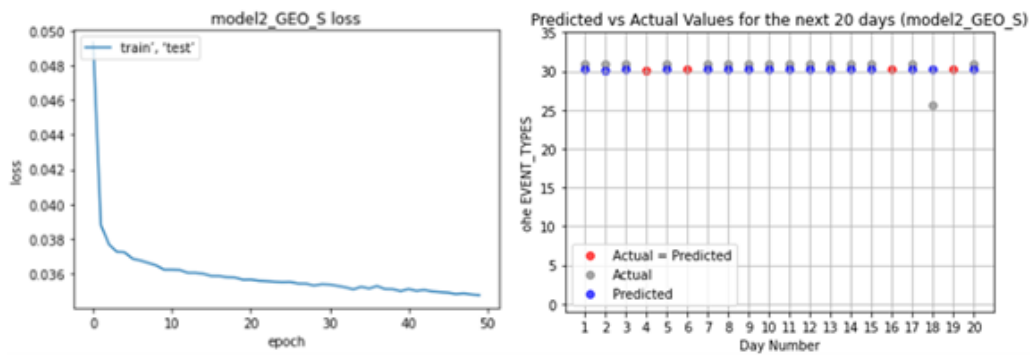


Figure 49: model1 accuracy over the test set for spring Georgia: 20%

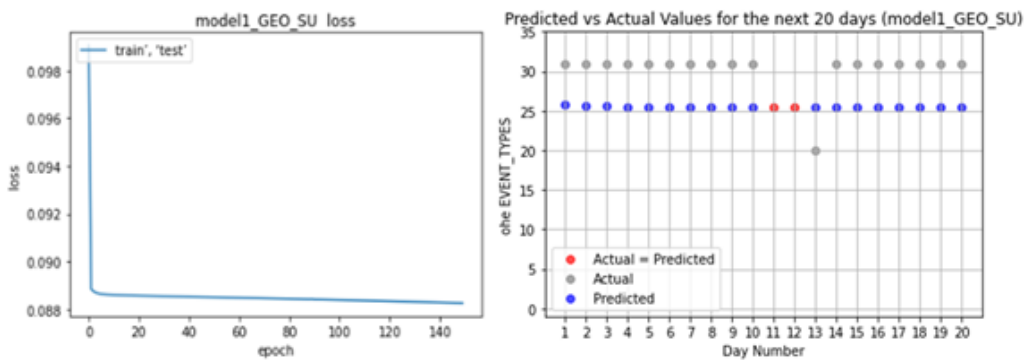


Figure 50: model1 accuracy over the test set for summer Georgia: 10%

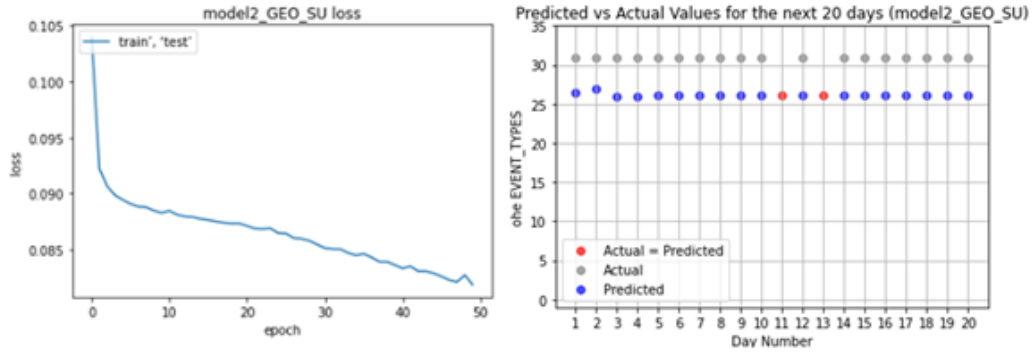


Figure 51: model2 accuracy over the test set for summer Georgia: 10%

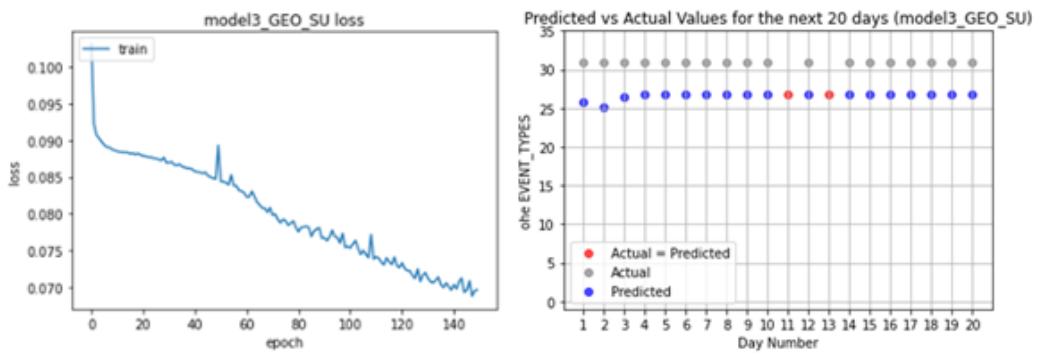


Figure 52: model3 accuracy over the test set for summer Georgia: 10%

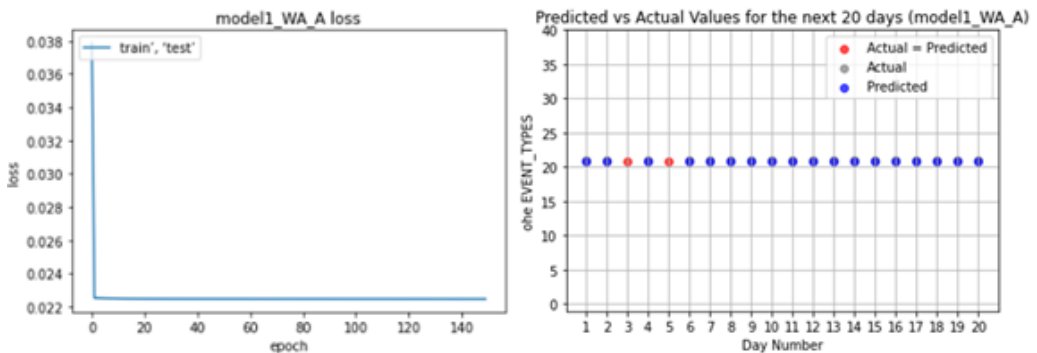


Figure 53: model3 accuracy over the test set for autumn Washington: 10%

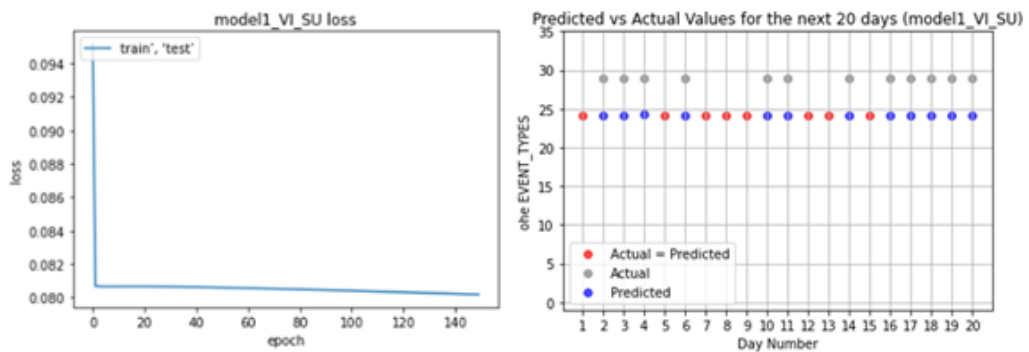


Figure 54: model1 accuracy over the test set for summer Virginia: 40%

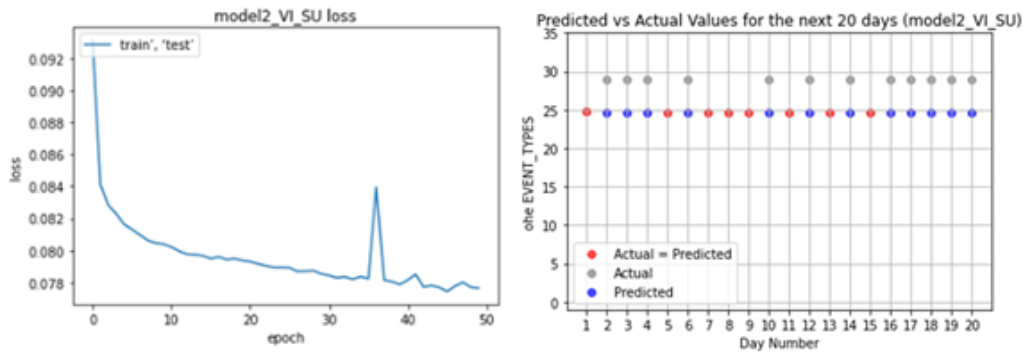


Figure 55: model2 accuracy over the test set for summer Virginia: 40%

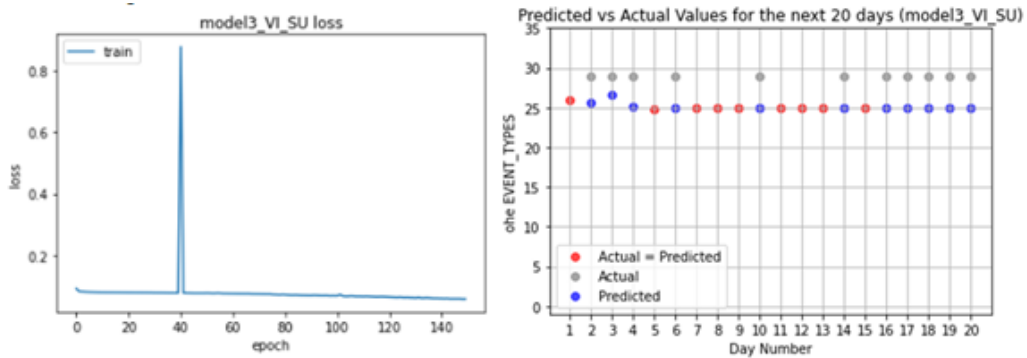


Figure 56: model3 accuracy over the test set for summer Virginia: 45%

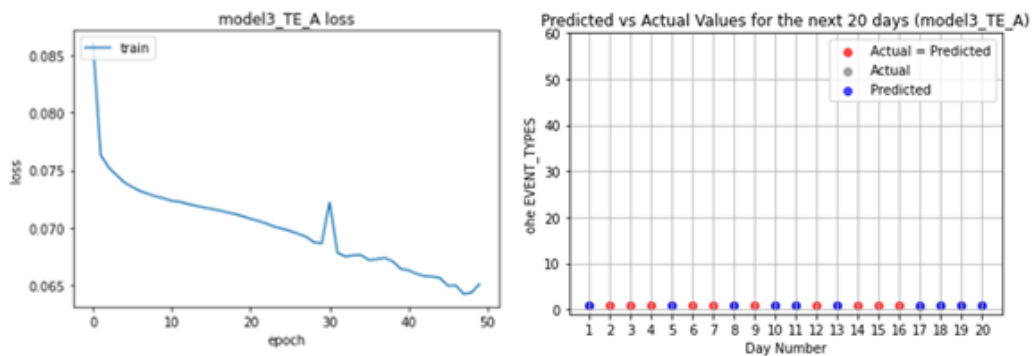


Figure 57: model3 accuracy over the test set for autumn Texas: 50%

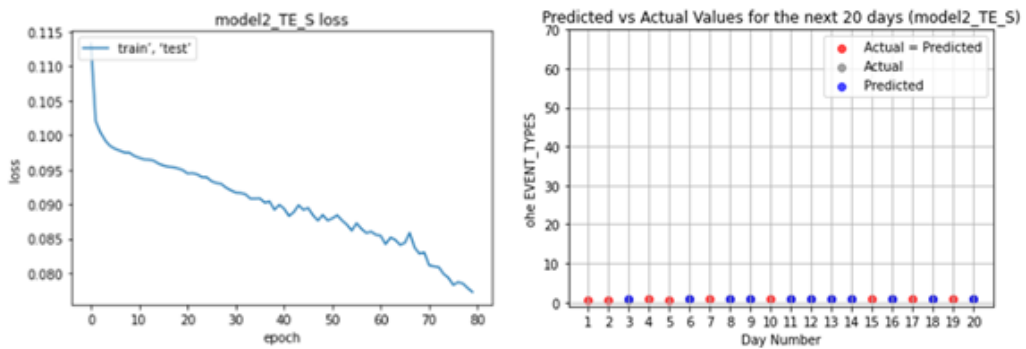


Figure 58: model2 accuracy over the test set for spring Texas: 45%

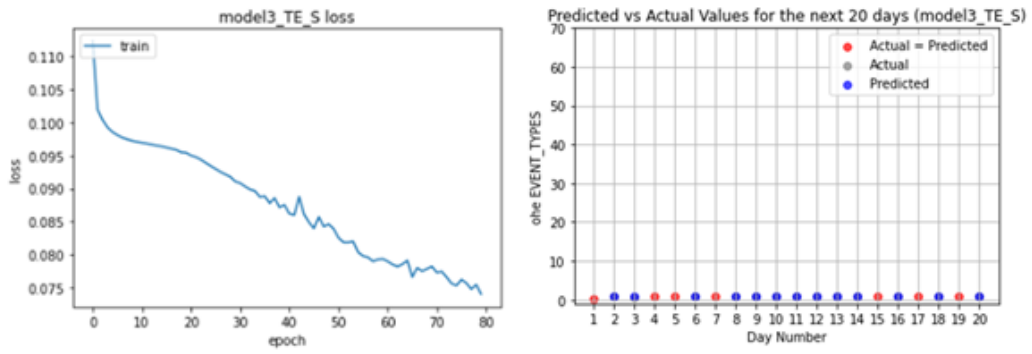


Figure 59: model3 accuracy over the test set for spring Texas: 35%

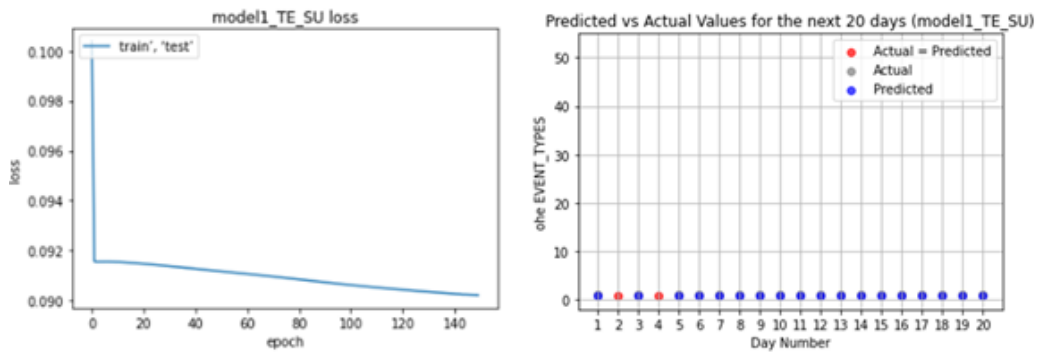


Figure 60: model1 accuracy over the test set for summer Texas: 10%

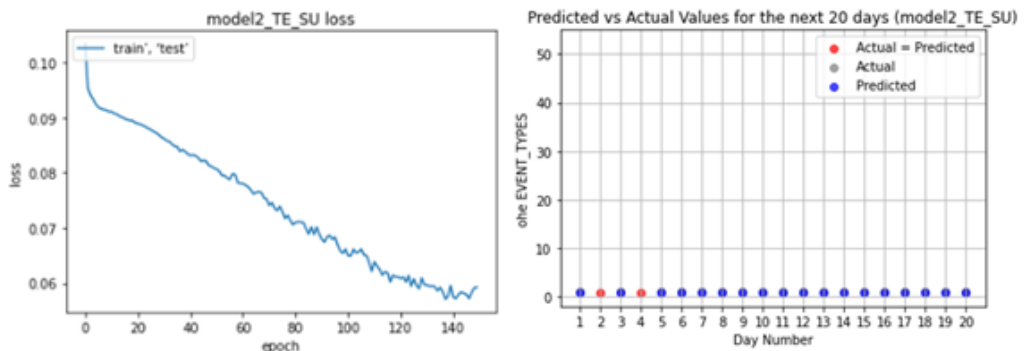


Figure 61: model2 accuracy over the test set for summer Texas: 10%

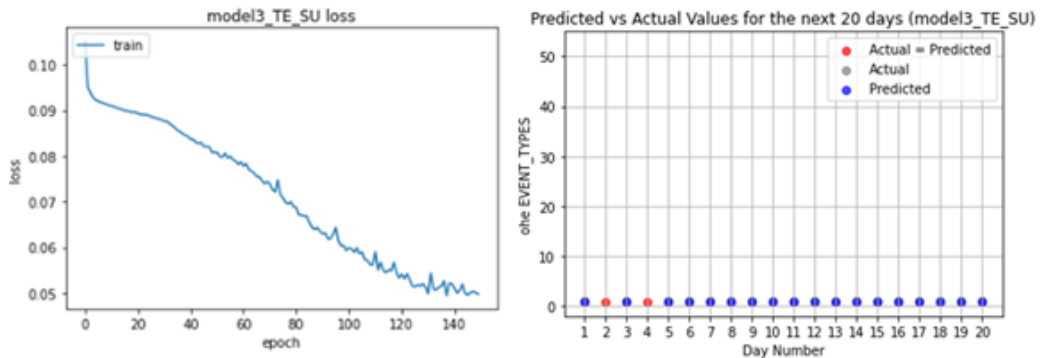


Figure 62: model3 accuracy over the test set for summer Texas: 10%

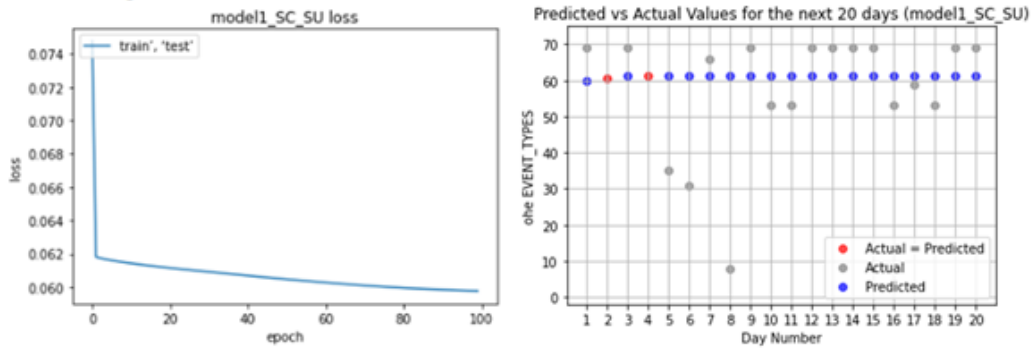


Figure 63: model1 accuracy over the test set for summer South Carolina: 35%

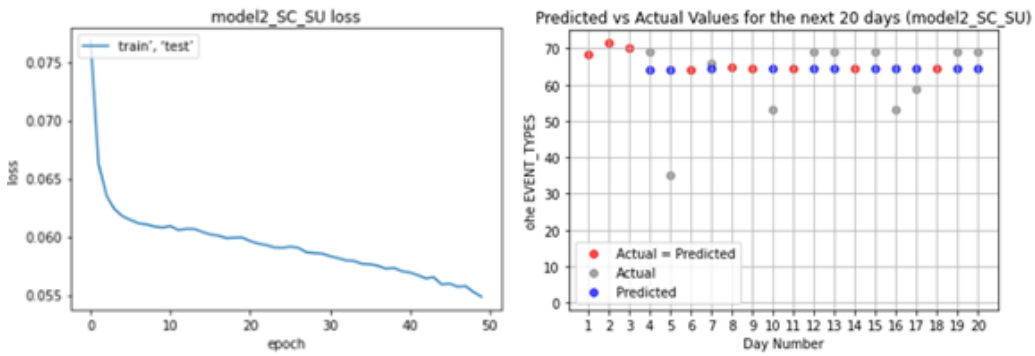


Figure 64: model2 accuracy over the test set for summer South Carolina: 45%

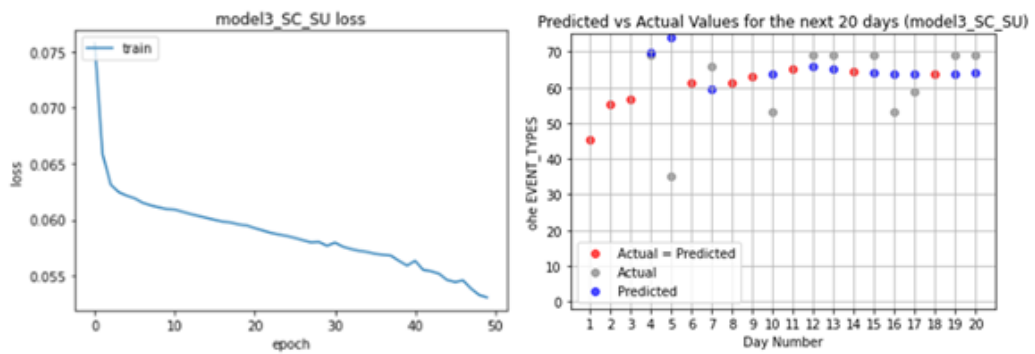


Figure 65: model3 accuracy over the test set for summer South Carolina: 35%

	High Wind	Heavy Rain	Winter Storm	Dense Fog	Thunderstorm Wind	Marine Thunderstorm Wind	Tornado	Strong Wind	Flood	Marine High Wind	Dust Storm	Extreme Cold/Chill	High Surf	Flash Flood	Excessive Heat	Ice Storm	Coastal Flood	Tsunami	Marine Heavy Wind	Tropical Storm	Marine Dense Fog	Marine Tropical Storm	Estimated Duration
models																							
CHL WINTER																							
1st	X	X	X		X			X	X	X			X	X						X			24:00:00
2nd	X				X			X	X	X			X	X						X			24:00:00
3rd	X				X			X	X	X			X	X						X			23:40:15
4th	X				X			X	X	X			X	X						X			22:02:34
5th	X				X			X	X	X			X	X						X			7:24:00
models																							
geo_AUTUMN																							
1st		X			X			X	X	X			X	X						X			21:41:54
2nd		X			X			X	X	X			X	X						X			21:40:49
3rd		X			X			X	X	X			X	X						X			21:41:38
4th		X			X			X	X	X			X	X						X			21:51:06
5th		X			X			X	X	X			X	X						X			21:50:56
models																							
geo_AUTUMN																							
1st	X	X			X			X	X	X			X	X						X			23:23:28
2nd	X	X			X			X	X	X			X	X						X			23:21:58
3rd	X	X			X			X	X	X			X	X						X			23:21:51
4th	X	X			X			X	X	X			X	X						X			23:22:23
5th	X	X			X			X	X	X			X	X						X			23:26:01
models																							
be_AUTUMN																							
1st	X				X			X	X	X			X	X						X			< 30 min
2nd	X				X			X	X	X			X	X						X			< 30 min
3rd	X				X			X	X	X			X	X						X			< 30 min
4th	X				X			X	X	X			X	X						X			< 30 min
5th	X				X			X	X	X			X	X						X			< 30 min
models																							
be_SPRING																							
1st	X				X			X	X	X			X	X						X			< 30 min
2nd	X				X			X	X	X			X	X						X			< 30 min
3rd	X				X			X	X	X			X	X						X			< 30 min
4th	X				X			X	X	X			X	X						X			< 30 min
5th	X				X			X	X	X			X	X						X			< 30 min
models																							
be_SPRING																							
1st	X				X			X	X	X			X	X						X			< 30 min
2nd	X				X			X	X	X			X	X						X			< 30 min
3rd	X				X			X	X	X			X	X						X			< 30 min
4th	X				X			X	X	X			X	X						X			< 30 min
5th	X				X			X	X	X			X	X						X			< 30 min

Figure 66: Forecasts for the next 5 days per season and port for 2023

	High Wind	Heavy Rain	Winter Storm	Dense Fog	Thunderstorm	Thunderstorm Wind	Marine Thunderstorm	Tornado	Strong Wind	Flood	Marine High Wind	Dust Storm	Extreme Cold/Wind Chill	High Surf	Flash Flood	Excessive Heat	Ice Storm	Coastal Flood	Tsunami	Marine Hail	Marine Strong Wind	Tropical Storm	Dense Fog	Marine Tropical Storm	Estimated Duration
cal_SUMMER																									
model1	1st	X													X	X									0:07:18
	2nd	X													X	X									21:09:30
	3rd	X													X	X									21:09:55
	4th	X													X	X									21:28:23
	5th	X													X	X									21:09:20
model2																									
	1st	X													X	X									19:58:14
	2nd	X													X	X									23:44:21
	3rd	X													X	X									0:36:23
	4th	X													X	X									15:28:14
	5th	X													X	X									22:31:49
model3																									
	1st	X													X	X									19:58:14
	2nd	X													X	X									22:35:08
	3rd	X													X	X									20:23:46
	4th	X													X	X									13:41:15
	5th	X													X	X									9:58:59
ny_SUMMER																									
model1	1st	X													X										20:09:52
	2nd	X													X										20:07:39
	3rd	X													X										20:22:17
	4th	X													X										20:33:41
	5th	X													X										20:37:21
model2																									
	1st																								19:10:56
	2nd																								19:08:14
	3rd																								19:08:15
	4th																								19:16:30
	5th																								19:16:04
model3																									
	1st																								17:48:28
	2nd	X													X										21:53:17
	3rd														X										19:44:33
	4th														X										19:51:02
	5th														X										20:03:06

Figure 67: Forecasts for the next 5 days per season and port for 2023

	High Wind	Heavy Rain	Winter Storm	Dense Fog	Thunderstorm Wind	Marine Thunderstorm Wind	Tornado	Strong Wind	Flood	Marine High Wind	Dust Storm	Extreme Cold/Chill	High Surf	Flash Flood	Excessive Heat	Ice Storm	Coastal Flood	Tsunami	Marine Strong Wind	Tropical Storm	Marine Dense Fog	Marine Tropical Storm	Estimated Duration	
models geo_SUMMER	1st	X																						21:17:55
	2nd					X			X												X			17:58:40
	3rd																				X			21:45:22
	4th																					X		20:16:42
	5th																						X	20:03:31
models sec_SUMMER	1st																							21:56:58
	2nd																							21:56:58
	3rd																							21:56:58
	4th																							21:56:58
	5th																							21:56:58
models sec_SUMMER	1st		X																					20:51:08
	2nd						X						X											21:20:39
	3rd																							23:09:28
	4th																							0:59:52
	5th																	X						23:18:35
models sec_SUMMER	1st						X																	23:30:02
	2nd		X																					18:31:21
	3rd																							19:25:40
	4th																							21:28:51
	5th																							23:51:30
models te_SUMMER	1st	X																						< 30 min
	2nd	X																						< 30 min
	3rd	X																						< 30 min
	4th	X																						< 30 min
	5th	X																						< 30 min

Figure 68: Forecasts for the next 5 days per season and port for 2023

	High Wind	Heavy Rain	Winter Storm	Dense Fog	Thunderstorm Wind	Marine Thunderstorm Wind	Tornado	Strong Wind	Flood	Marine High Wind	Dust Storm	Extreme Cold/Chill	High Surf	Flash Flood	Excessive Heat	Ice Storm	Coastal Flood	Tsunami	Marine Hail	Marine Strong Wind	Tropical Storm	Dense Fog	Marine Tropical Storm	Estimated Duration
1st	x																							< 30 min
2nd	x																							< 30 min
3rd	x																							< 30 min
4th	x																							< 30 min
5th	x																							< 30 min

Figure 69: Forecasts for the next 5 days per season and port ports for 2023

References

- Agrawal, S., Barrington, L., Bromberg, C., Burge, J., Gazen, C., and Hickey, J. (2019). Machine learning for precipitation nowcasting from radar images. *arXiv preprint arXiv:1912.12132*.
- Alderson, D. L., Brown, G. G., and Carlyle, W. M. (2015). Operational models of infrastructure resilience. *Risk Analysis*, 35(4):562–586.
- Anderson, J. and Bausch, C. (2006). Climate change and natural disasters: Scientific evidence of a possible relation between recent natural disasters and climate change. *Policy Department Economic and Scientific Policy*, 2.
- Athanasatos, S., Michaelides, S., and Papadakis, M. (2014). Identification of weather trends for use as a component of risk management for port operations. *Natural hazards*, 72(1):41–61.
- Awad, M. and Khanna, R. (2015). *Deep Neural Networks*, pages 127–147. Apress, Berkeley, CA.
- Bauer, P., Thorpe, A., and Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55.
- Brink, H. d., Können, G., Opsteegh, J., Oldenborgh, G., and Burgers, G. (2005). Estimating return periods of extreme events from ecmwf seasonal forecast ensembles. *International Journal of Climatology*, 25:1345–1354.
- Cao, X. and Lam, J. S. L. (2019). Simulation-based severe weather-induced container terminal economic loss estimation. *Maritime Policy & Management*, 46(1):92–116.
- Coles, S., Bawa, J., Trenner, L., and Dorazio, P. (2001). *An introduction to statistical modeling of extreme values*, volume 208. Springer.
- Esteban, M., Mikami, T., Shibayama, T., Takagi, H., Jonkman, S. N., and van Ledden, M. (2014). Climate change adaptation in tokyo bay: The case for a storm surge barrier. *Coastal Engineering Proceedings*, (34):35–35.
- Gharehgozli, A. H., Mileski, J., Adams, A., and von Zharen, W. (2017). Evaluating a “wicked problem”: A conceptual framework on seaport resiliency in the event of weather disruptions. *Technological forecasting and social change*, 121:65–75.
- Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.
- Hofmann, E. (2013). *Supply chain management: Strategy, planning and operation*, s. chopra, p. meindl.
- Höppe, P. and Grimm, T. (2008). Rising natural catastrophe losses—what is the role of climate change? In *Economics and Management of Climate Change*, pages 13–22. Springer.
- Izaguirre, C., Losada, I., Camus, P., Vigh, J., and Stenek, V. (2021). Climate change risk to global port operations. *Nature Climate Change*, 11(1):14–20.
- Khurana, R., Mugabe, D., and Etienne, X. L. (2022). Climate change, natural disasters, and institutional integrity. *World Development*, 157:105931.

- Lenton, T., Footitt, A., and Dlugolecki, A. (2009). *Major tipping points in the earth's climate system and consequences for the insurance sector*. World Wide Fund for Nature.
- León-Mateos, F., Sartal, A., López-Manuel, L., and Quintas, M. A. (2021). Adapting our sea ports to the challenges of climate change: Development and validation of a port resilience index. *Marine Policy*, 130:104573.
- Lorenç, A. C. (1986). Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112(474):1177–1194.
- Ng, A. K., Zhang, H., Afenyo, M., Becker, A., Cahoon, S., Chen, S.-l., Esteban, M., Ferrari, C., Lau, Y.-y., Lee, P. T.-W., et al. (2018). Port decision maker perceptions on the effectiveness of climate adaptation actions. *Coastal Management*, 46(3):148–175.
- Panahi, R., Ng, A. K., and Pang, J. (2020). Climate change adaptation in the port industry: A complex of lingering research gaps and uncertainties. *Transport Policy*, 95:10–29.
- Pielke, R. A. (2005). Attribution of disaster losses. *Science*, 310(5754):1615c–1616c.
- Pielke Jr, R. A., Agrawala, S., Bouwer, L. M., Burton, I., Changnon, S., Glantz, M. H., Hooke, W. H., Klein, R. J., Kunkel, K., Mileti, D., et al. (2005). Clarifying the attribution of recent disaster losses: a response to epstein and mccarthy. *Bulletin of the American Meteorological Society*, 86(10):1481–1483.
- Ren, X., Li, X., Ren, K., Song, J., Xu, Z., Deng, K., and Wang, X. (2021). Deep learning-based weather prediction: a survey. *Big Data Research*, 23:100178.
- Rose, A. and Wei, D. (2013). Estimating the economic consequences of a port shutdown: the special role of resilience. *Economic Systems Research*, 25(2):212–232.
- Seneviratne, S. I., Nicholls, N., Easterling, D., Goodess, C. M., Kanae, S., Kossin, J., Luo, Y., Marengo, J., Innes, K. M., Rahimi, M., Reichstein, M., Sorteberg, A., Vera, C., Zhang, X., Rusticucci, M., Semenov, V., Alexander, L. V., Allen, S., Benito, G., Cavazos, T., Clague, J., Conway, D., Della-Marta, P. M., Gerber, M., Gong, S., Goswami, B. N., Hemer, M., Huggel, C., den Hurk, B. V., Kharin, V. V., Kitoh, A., Tank, A. M. G. K., Li, G., Mason, S., Guire, W. M., Oldenborgh, G. J. V., Orłowsky, B., Smith, S., Thiaw, W., Velegakis, A., Yiou, P., Zhang, T., Zhou, T., and Zwiers, F. W. (2012). Changes in climate extremes and their impacts on the natural physical environment. pages 109–230.
- Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28.
- Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W.-k., and Woo, W.-c. (2017). Deep learning for precipitation nowcasting: A benchmark and a new model. *Advances in neural information processing systems*, 30.
- Smythe, T. C. (2013). *Assessing the impacts of Hurricane Sandy on the Port of New York and New Jersey's Maritime responders and response infrastructure*. Natural Hazards Center.
- Sønderby, C. K., Espeholt, L., Heek, J., Dehghani, M., Oliver, A., Salimans, T., Agrawal, S., Hickey, J., and Kalchbrenner, N. (2020). Metnet: A neural weather model for precipitation

- forecasting. *arXiv preprint arXiv:2003.12140*.
- Svensson, C. and Jones, D. A. (2002). Dependence between extreme sea surge, river flow and precipitation in eastern britain. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, 22(10):1149–1168.
- Van Aalst, M. K. (2006). The impacts of climate change on the risk of natural disasters. *Disasters*, 30(1):5–18.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Wang, Y., Gao, Z., Long, M., Wang, J., and Philip, S. Y. (2018). Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In *International Conference on Machine Learning*, pages 5123–5132. PMLR.
- Wang, Y., Long, M., Wang, J., Gao, Z., and Yu, P. S. (2017). Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. *Advances in neural information processing systems*, 30.
- Wendler-Bosco, V. and Nicholson, C. (2020). Port disruption impact on the maritime supply chain: a literature review. *Sustainable and Resilient Infrastructure*, 5(6):378–394.
- Yang, Y.-C. and Ge, Y.-E. (2020). Adaptation strategies for port infrastructure and facilities under climate change at the kaohsiung port. *Transport Policy*, 97:232–244.
- Zhang, G. P. and Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European journal of operational research*, 160(2):501–514.