Elias Sandner, Bsc

Refining Ocean Buoy Forecasting with Physics-Informed Regularization through Ocean Models

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Guetl

Institute of Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Kappe

Co-Supervisors

Dr. Mahdi Abdelguerfi

M.Sc. Austin Schmidt

Canizaro Livingston Gulf States Center for Environmental Informatics

Graz, September 2023

Elias Sandner, Bsc

Verfeinerung der Ozeanbojen-Prognose mit physikbasierter Regularisierung durch Ozeanmodelle

Masterarbeit

zur Erlangung des akademischen Grades eines

Diplom-Ingenieur

Masterstudium: Software Engineering and Management

eingereicht an der

Technische Universität Graz

Betreuer

Assoc.Prof. Dipl.-Ing. Dr.techn. Christian Guetl

Institute of Interactive Systems and Data Science
Vorstand: Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Kappe

Mitbetreuer

Dr. Mahdi Abdelguerfi

M.Sc. Austin Schmidt

Canizaro Livingston Gulf States Center for Environmental Informatics

Graz, September 2023

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

Abstract

Centered on marine weather forecasting, this thesis proposes a specialized toolkit for generating datasets and executing forecasting trials, additionally encompassing a study on physics-informed regularization in forecasting.

Deep learning has become the dominating approach in many forecasting domains. Within the realm of weather prediction, it confronts traditional numerical methods, but has not emerged as a full alternative. While the majority of weather forecasting emphasizes land, marine weather prediction is the central theme of this thesis. Advancements in this field are crucial for systems that protect mariners from extreme weather events and also for systems offering timely warnings for potential meteorological disasters along the coastlines. Moreover, enhancements in marine weather prediction can be used to improve ship routing, leading to cost savings and reduced emissions.

Although marine weather forecasting draws from various data sources, this work centers on forecasting ocean buoy observations, specifically those provided by the National Data Buoy Center (NDBC). Prediction of these observations can be achieved through numerical weather prediction, a theory-driven approach, or by leveraging a data-driven method, notably deep learning. Also, combining methods from both approaches is possible. Therefore, the dominant technique used is physics-informed neural networks. While this approach necessitates in-depth knowledge of meteorological correlations and tailored solutions for each weather parameter, this thesis introduces an alternative.

Weather models, such as the fifth generation of reanalysis for the global climate and weather (ERA5) provided by the European Centre for Medium-Range Weather Forecasts (ECMWF), are rooted in theory-driven calculations. This thesis aims to determine whether leveraging available physics-informed data can enhance the forecasting of real-world observations.

Consequently, two dataset representations are introduced to combine the spatial-temporal data from both NDBC and ERA5 sources. In addition, this work investigates four deep learning models, each tailored with a loss function that controls the impact of physics-informed data in the training process. This method of integrating physics into the dataset, rather than

directly into the neural network, is termed physics-informed regularization (PIR).

To thoroughly examine both dataset representations and PIR, a toolkit for dataset creation and forecasting experiments was developed. While tailored for this project, the toolkit is crafted to be adaptable and expandable for subsequent related research endeavors. Utilizing the toolkit, a study was carried out to evaluate the behavior of various PIR levels through four deep learning models: LSTM, GRU, CNN, and TCN.

The findings reveal that PIR has the potential to boost accuracy. Though not universally valid, it illuminates the effectiveness of the adopted strategy and lays the groundwork for prospective research.

Kurzfassung

In dieser Arbeit, die sich auf die Vorhersage des Seewetters konzentriert, wird ein spezielles Toolkit für die Erstellung von Datensätzen und die Durchführung von Vorhersage-Experimenten vorgeschlagen. Außerdem wird eine Studie über die Anwendung von physikalisch informierter Regularisierung bei der Vorhersage von Messwerten präsentiert.

Deep Learning hat sich in vielen Bereichen der Vorhersage zum dominierenden Ansatz entwickelt. Im Bereich der Wettervorhersage konkurriert es mit den traditionellen numerischen Methoden, hat sich aber noch nicht als vollständige Alternative erwiesen. Während sich die meisten Wettervorhersagen auf das Festland konzentrieren, ist die Wettervorhersage auf See das zentrale Thema dieser Arbeit. Fortschritte in diesem Bereich sind entscheidend für Systeme, die Schiffsmannschaften vor extremen Wetterereignissen schützen und auch für Systeme, die rechtzeitig vor möglichen Naturkatastrophen an den Küsten warnen. Darüber hinaus können Verbesserungen bei der Vorhersage des Seewetters genutzt werden, um die Routenplanung von Schiffen zu verbessern, was zu Kosteneinsparungen und geringeren Emissionen führt.

Obwohl die Vorhersage des Seewetters aus verschiedenen Datenquellen gespeist wird, konzentriert sich diese Arbeit auf die Vorhersage von Messwerten gesammelt durch Bojen. Konkret werden Messwerte die vom National Data Buoy Center (NDBC) zur Verfügung gestellt werden, berücksichtigt. Die Vorhersage dieser Beobachtungen kann durch numerische Wettervorhersage, einen theoriegesteuerten Ansatz, oder durch die Nutzung einer datengesteuerten Methode, insbesondere Deep Learning, erreicht werden. Auch die Kombination von Methoden aus beiden Ansätzen ist möglich. Dafür sind physikalisch informierte neuronale Netze der dominierende Ansatz. Während dieser ein tiefgreifendes Wissen über meteorologische Zusammenhänge und maßgeschneiderte Lösungen für jeden Wetterparameter erfordert, stellt diese Arbeit eine Alternative vor.

Wettermodelle, wie die fünfte Generation der Reanalyse für das globale Klima und Wetter (ERA5) des Europäischen Zentrums für mittelfristige Wettervorhersage (ECMWF), basieren auf theoriegestützten Berechnungen. In dieser Arbeit wird untersucht, ob die Nutzung verfügbarer physikalisch

informierter Daten die Vorhersage tatsächlichen Messwerten verbessern kann.

Dafür werden zwei Datensatzdarstellungen eingeführt, um die räumlich-zeitlichen Daten aus den NDBC- und ERA5-Quellen zu kombinieren. Darüber hinaus werden in dieser Arbeit vier Deep-Learning-Modelle untersucht, die jeweils mit einer Verlustfunktion ausgestattet sind, die den Einfluss der physikalischen Daten auf den Trainingsprozess kontrolliert. Diese Methode der Integration von physikalischen Daten in den Datensatz und nicht direkt in das neuronale Netzwerk wird als physikalisch informierte Regularisierung (PIR) bezeichnet.

Um sowohl die Repräsentationen der Datensätze als auch PIR gründlich zu untersuchen, wurde ein Toolkit für die Erstellung von Datensätzen und Vorhersageexperimente entwickelt. Das Toolkit ist zwar auf dieses Projekt zugeschnitten, aber es ist so konzipiert, dass es für spätere verwandte Forschungsvorhaben angepasst und erweitert werden kann. Mit Hilfe des Toolkits wurde eine Studie durchgeführt, um das Verhalten verschiedener PIR-Stufen mit Hilfe von vier Deep Learning-Modellen zu bewerten: LSTM, GRU, CNN und TCN.

Die Ergebnisse zeigen, dass PIR das Potenzial hat, die Vorhersagegenauigkeit zu erhöhen. Obwohl die Ergebnisse nicht allgemeingültig sind, verdeutlichen sie die Effektivität der gewählten Strategie und bilden die Grundlage für künftige Forschungen.

Acknowledgment

First and foremost, I would like to express my profound gratitude to Prof. Christian Gütl for his unwavering guidance and mentorship throughout the course of this thesis. His swift and comprehensive feedback was instrumental in shaping this work.

I am deeply thankful to the Gulf States Center of Environment Informatics (GulfSCEI), led by Dr. Mahdi Abdelguerfi, for hosting me in New Orleans for five enriching months. A special mention goes to PhD student Austin Schmidt; his patience and willingness to explain intricate details and offer valuable advice have been of immense help.

Last but by no means least, my heartfelt thanks go to my family. In particular, I owe a debt of gratitude to my mother. It is her relentless support and belief that transformed a child, who once faced challenges at the beginning of his educational journey, into someone who could complete a master's thesis at TU-Graz. Your faith in me has been my guiding light.

Contents

1 Introduction **1**
- 1.1 Aim and Objective . . . 1
- 1.2 Methodology and Contribution . . . 2
- 1.3 Structure . . . 3

2 Background and Related Work **5**
- 2.1 Numerical Weather Prediction . . . 5
  - 2.1.1 History of Numerical Weather Prediction . . . 6
  - 2.1.2 Current State and Trends in Numerical Weather Prediction . . . 7
- 2.2 Deep Learning . . . 10
  - 2.2.1 Recurrent Neural Network . . . 13
  - 2.2.2 Convolutional Neural Network . . . 20
- 2.3 Deep Learning Weather Prediction . . . 24
  - 2.3.1 Completely Data-Driven Hybrid Architectures . . . 25
  - 2.3.2 Theory-guided Coupling Architectures . . . 26
  - 2.3.3 NWP and DLWP: Advantages and Limitations . . . 27
- 2.4 Data Sources . . . 28
  - 2.4.1 National Data Buoy Center . . . 29
  - 2.4.2 Fifth ECMWF Reanalysis . . . 32
  - 2.4.3 Comparative Analysis of Dataset Features: NDBC and ERA5 . . . 34
- 2.5 Summary . . . 38

3 Requirements and Concept **40**
- 3.1 Motivation . . . 40
- 3.2 Problem Statement . . . 40
- 3.3 Requirements . . . 41
  - 3.3.1 Functional Requirements . . . 41
  - 3.3.2 Non-functional Requirements . . . 42
- 3.4 Conceptual Architecture . . . 43
- 3.5 Summary . . . 46

4 Design and Development **48**
- 4.1 Data Representation . . . 48
  - 4.1.1 Multi-Location Modelling . . . 48

4.1.2 Station-Specific Unified Modelling . . . 50
4.2 Neural Network Design . . . 51
4.2.1 Neural Network Architecture . . . 51
4.2.2 Custom Loss Function . . . 54
4.3 Toolkit Architecture . . . 55
4.4 Development Details . . . 59
4.4.1 Data Sourcing . . . 59
4.4.2 Dataset Builder . . . 63
4.4.3 Test Environment . . . 69
4.4.4 Report Creation . . . 73
4.5 Summary . . . 74

5 Empirical Analysis of Physics-Informed Regularization 76
5.1 Study Setup . . . 76
5.2 Results and Findings . . . 84
5.2.1 MLM-LSTM . . . 84
5.2.2 MLM-GRU . . . 87
5.2.3 MLM-CNN . . . 89
5.2.4 MLM-TCN . . . 91
5.2.5 SSUM-LSTM . . . 92
5.2.6 SSUM-GRU . . . 94
5.2.7 SSUM-CNN . . . 96
5.2.8 SSUM-TCN . . . 97
5.3 Discussion . . . 99
5.4 Summary . . . 102

6 Lessons Learned 104
6.1 Literature Research . . . 104
6.2 Development . . . 104
6.3 Study . . . 105
6.4 Personal . . . 105

7 Conclusion and Future Work 106
7.1 Conclusion . . . 106
7.2 Future Work . . . 108

Bibliography 110

List of Figures

2.1 Hierarchical classification of deep learning architectures; Inspired by (Goodfellow et al., 2016) . . . 10
2.2 Standard recurrent neural network: A repeating module . . . 14
2.3 Long Short Term Memory cell and its operation . . . 16
2.4 Gated Recurrent Unit cell and its operation . . . 19
2.5 LeNet-5 architecture as an example of a standard CNN (Cong & Zhou, 2023) . . . 22
2.6 Performance comparison between DLWP and NWP at different temporal and spatial scales. In the area covered by each approach, the darker the color, the better the performance (Ren et al., 2021). . . . 28
2.7 Insights from NDBC metadata . . . 31
a Number of available files by years . . . 31
b Distribution of station operator . . . 31
c Distribution of station types . . . 31

3.1 Conceptual architecture of the toolkit for dataset creation and test execution divided into 7 components. . . . 43

4.1 Concept comparison of MLM and SSUM . . . 49
4.2 Comparison of different data imputation techniques on NDBC data . . . 51
a Accuracy . . . 51
b Computation time . . . 51
4.3 Architecture of the integrated platform for dataset creation and forecast experiments . . . 56
4.4 Screenshot of station page (yellow: station owner, green: station type, blue: coordinates) . . . 60
4.5 Screenshot: mlm_build_dataset.ipynb GUI . . . 64
4.6 Example of NDBC data header . . . 65
4.7 Example of ERA5 data header . . . 66
4.8 Example of final MLM data header . . . 66
4.9 Screenshot: SSUM_Build_Dataset.ipynb GUI . . . 67
4.10 Example of SSUM data header before transforming to supervised problem . . . 68
4.11 Example SSUM final data header with $n_{\text{in}} = 3$ and $n_{\text{out}} = 2$ . . . 69

4.12 Screenshot: mlm_experiments.ipynb GUI 70
4.13 Screenshot: ssum_experiments.ipynb GUI 72

5.1 Map of considered area in Gulf of Mexico: Blue pins represent available stations, while red pins indicate the stations that have been chosen for the experiments. 77
5.2 Distribution of station owners in the considered area 77
5.3 Number of available NDBC data files in area of interest by year 78
5.4 File availability distribution (Files available in black) 79
5.5 Tree diagram of test scenarios 80
5.6 Unprocessed MAE Results of MLM experiment with LSTM architecture and Alpha=0 82
5.7 Unprocessed error rates of SSUM experiment with LSTM architecture and Alpha=0 83
5.8 Average MAE and MSE of LSTM architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted. 84
5.9 MLM-LSTM: Absolute improvement in relation to baseline experiment (Alpha=1) 85
a MAE 85
b MSE 85
5.10 MLM-LSTM: Average relative MAE and MSE improvement in percent across all features compared to baseline 86
5.11 Average MAE and MSE of GRU architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted. 87
5.12 MLM-GRU: Absolute improvement in relation to baseline experiment (Alpha=1) 87
a MAE 87
b MSE 87
5.13 MLM-GRU: Average relative MAE and MSE improvement in percent across all features compared to baseline 88
5.14 Average MAE and MSE of CNN architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted. 89
5.15 MLM-CNN: Absolute improvement in relation to baseline experiment (Alpha=1) 89
a MAE 89
b MSE 89

5.16 MLM-CNN: Average relative MAE and MSE improvement in percent across all features compared to baseline 90
5.17 Average MAE and MSE of TCN architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted. 91
5.18 MLM-TCN: Absolute improvement in relation to baseline experiment (Alpha=1) 91
a MAE 91
b MSE 91
5.19 MLM-TCN: Average relative MAE and MSE improvement in percent across all features compared to baseline 92
5.20 MAE and MSE of LSTM architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted. 92
5.21 SSUM-LSTM: Absolute improvement in relation to baseline experiment (Alpha=1) 93
a MAE 93
b MSE 93
5.22 SSUM-LSTM: Average relative MAE and MSE improvement in percent across all features compared to baseline 93
5.23 MAE and MSE of GRU architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted. 94
5.24 SSUM-GRU: Absolute improvement in relation to baseline experiment (Alpha=1) 94
a MAE 94
b MSE 94
5.25 SSUM-GRU: Average relative MAE and MSE improvement in percent across all features compared to baseline 95
5.26 MAE and MSE of CNN architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted. 96
5.27 SSUM-CNN: Absolute improvement in relation to baseline experiment (Alpha=1) 96
a MAE 96
b MSE 96
5.28 SSUM-CNN: Average relative MAE and MSE improvement in percent across all features compared to baseline 97
5.29 MAE and MSE of TCN architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted. 97

5.30 SSUM-TCN: Absolute improvement in relation to baseline experiment (Alpha=1) 98
a MAE 98
b MSE 98
5.31 SSUM-TCN: Average relative MAE and MSE improvement in percent across all features compared to baseline 98

List of Tables

2.1 DL Architectures comparison 14
2.2 Relevant studies on TSF using LSTM 17
2.3 Relevant studies on TSF using GRU 20
2.4 Relevant studies on TSF using CNN 22
2.5 Relevant studies on TSF using TCN 24
2.6 Data header of: Standard Meteorological Data (NDBC, n.d.-b) 30
2.7 Description of dataset: ERA5 hourly data on single levels (C3S CDS, 2023) 34
2.8 Excluded NDBC features (NDBC, n.d.-b) 35
2.9 Relevant NDBC measurements and their ERA5 counterparts (C3S CDS, 2023; NDBC, n.d.-b) 37

4.1 Layer-wise comparison between LSTM and GRU architectures 52
4.2 CNN Architecture 53
4.3 TCN Architecture 53
4.4 Libraries and their usages 57
4.5 mlm_build_dataset.ipynb parameters 64
4.6 SSUM_Build_Dataset.ipynb parameters 67
4.7 mlm_experiment.ipynb parameters 70
4.8 SSUM_Experiment.ipynb parameters 72
4.9 Overview of evaluation files created by report_generator.ipynb 73

5.1 Selected parameters for dataset builder 81
5.2 Selected parameters for test environment 81
5.3 Comparison and suggestion of investigated settings 102

1 Introduction

According to Greek mythology, the gods sent the weather directly from their seat of power, Mount Olympus, to the Earth. Already the Greek polymath Aristotle opposed this view and around 340 B.C., in his work about "Meteorologica", he described atmospheric phenomena (Frisinger, 1972). Consequently, humans have always been driven to understand and forecast the weather. Even though it is common understanding today that weather is a physical phenomenon governed by natural laws, the precise prediction of weather remains a challenge.

This challenge is primarily addressed by meteorology, a science that focuses on the physical and chemical processes in the atmosphere. However, due to the complexity of this science, without advances in computer science, it would not be possible to calculate future weather before it occurs (Lynch, 2007). As in many other areas of application, artificial intelligence is attracting increasing interest in the field of weather forecasting. However, based on the current state of technology, it is not capable of completely replacing numerical weather prediction. (Schultz et al., 2021)

Marine weather forecasting is a subfield that has been researched in New Orleans since the US Army Signal Corps made the first attempt at such a forecast in the US in 1859 (May et al., 2014). Fast forward to today, the Canizaro Livingston Gulf States Center for Environmental Informatics (GulfSCEI) at the University of New Orleans continues contributing to this field. In collaboration with GulfSCEI, this thesis delves into synergizing the potential of artificial intelligence with established numerical weather prediction methods to enhance the prediction of ocean buoy measurements.

1.1 Aim and Objective

The primary goal of this work is to advance the field of marine weather forecasting. Progress in this area can have implications across various fields. The emphasis lies in safeguarding mariners from extreme weather events and also protecting coastal residents by providing timely warnings for potential meteorological calamities. Additionally, enhanced accuracy in ocean weather

forecasting can streamline shipping routes, yielding both economic benefits and decreased emissions.

Marine weather forecasts utilize data gathered from diverse sources such as satellites, aircraft, ships, land-based observation stations, and ocean buoys. (Heaslip, 2023) The primary focus of this thesis is the prediction of data stemming from ocean buoys. This can either be based on theory-driven methods that take into account the underlying physics or on machine learning techniques. When it comes to integrating both approaches, physics-informed neural networks play a crucial role. Given that this approach requires in-depth domain knowledge in meteorology and individual solutions for each measurement, an alternative path is explored.

European Centre for Medium-Range Weather Forecasts (ECMWF) offers various datasets of meteorologic data, which are generated using physics-based numerical weather prediction methods. This thesis seeks to determine whether integrating this physics-informed data can enhance the accuracy of deep learning-based ocean buoy observation predictions.

In common physics-informed neural networks, physical laws are integrated directly in the training process. However, the approach explored in this research embeds this knowledge into the dataset itself. As a result, appropriate dataset representations are developed along with a specialized loss function for the neural networks. This function introduces a parameter that governs the impact of physics-informed features on the adjustment of weights and biases during the training process. Since the physics information is not implemented in the neural network but is incorporated through the dataset, this approach is termed physics-informed regularization (PIR). Multiple degrees of PIR are assessed using four distinct deep learning models. While the investigation of PIR is exemplified on ocean buoy prediction, this approach can also be interesting for any physics based problem solved with machine learning where additional physics based data is available.

1.2 Methodology and Contribution

To investigate physics-informed regularization in detail, first a toolkit for dataset creation and test execution was developed. This toolkit covers the whole pipeline from data sourcing to report creation. By specifying certain ocean buoys and years of interest, the integrated dataset builder generates individual datasets. To appropriately represent the available data, two approaches are adopted. Multi-Location Modelling (MLM) portrays an entire area, not just a single location within one instance while Station-Specific Unified Modelling (SSUM) joins data from various stations, yielding a more

comprehensive dataset. The generated datasets, following either approach, can further be used in the test environment. This environment includes four distinct models, customized for investigating PIR. Second, the developed toolkit is employed in an extensive experimental study. Four neural network architectures, namely LSTM, GRU, CNN and TCN, all implemented with a custom loss function for regulating the influence of physics-informed data, were considered. Taking into account varying degrees of physics-informed influence, in total 88 forecasting experiments were executed.

The toolkit is developed not specifically for the executed study but for general usage. Based on certain parameters, individual datasets can be created. Moreover, the test environment can be expanded to accommodate any machine learning method, not just those focused on PIR, offering flexibility in test design. Therefore, this project contributes by providing a solid base for further empirical studies in the domain of ocean buoy forecasting. Moreover, the presented study contributes by shedding light on the influence of data representation, highlighting the potential of PIR in this domain, and suggesting avenues for promising future research.

1.3 Structure

In the initial segment of this thesis, Chapter 2, "Background and Related Work," lays the groundwork by exploring Numerical Weather Prediction, its history, and current trends. The discussion then shifts to Deep Learning, diving into various architectures, notably Recurrent and Convolutional Neural Networks. The chapter also introduces the concept of Deep Learning Weather Prediction, highlighting the synergy and contrasts between NWP and DLWP, and concludes by examining the primary data sources pivotal to this research.

Moving on, Chapter 3 sets the stage by defining the motivation behind the research and articulating a clear problem statement. The delineation of both functional and non-functional requirements is presented, leading to a comprehensive overview of the conceptual architecture of the proposed solution.

Chapter 4 delves deeply into the intricacies of data representation methods and the design intricacies of the neural networks employed. The broader toolkit architecture is fleshed out, covering aspects from data sourcing and dataset building to the testing environment and report creation processes.

The empirical core of the research is encapsulated in Chapter 5. Here, the study setup is outlined, followed by a meticulous presentation of results and findings, segmented by different models and architectures. This chapter

offers a deep dive into the implications of these results within the wider context of the domain.

In Chapter 6, a reflective lens is cast on the research process, chronicling the insights, experiences, and lessons learned. This chapter provides an introspective look into the challenges encountered and the knowledge derived from them.

The final chapter, Chapter 7, draws the thesis to a close by synthesizing the entire research journey. Major takeaways are summarized, and the discussion looks ahead, suggesting potential paths for future exploration in the field.

2 Background and Related Work

This chapter provides an in-depth overview of the foundational concepts underlying this thesis, starting with the evolution of numerical weather prediction and a brief on deep learning techniques. The application of deep learning in weather prediction is also addressed, alongside a analysis of notable data sources. This chapter serves to contextualize the research contributions presented in later chapters.

2.1 Numerical Weather Prediction

Numerical weather prediction (NWP) constitutes a weather forecasting methodology that leverages a set of equations governing fluid flow. By converting these equations into computationally executable code, NWP incorporates numerical methods and parameterizations of various physical processes. The subsequent simulations encompass an array of initial and boundary conditions and are conducted over a specific geographic area. (Steffen, n.d.)

The models use current weather observations along with historical data to calculate likely future weather patterns. NWP is used by meteorologists and weather forecasters around the world to provide accurate weather forecasts for short and long-term periods. It is also used in fields such as aviation (Mazzarella et al., 2022), agriculture (CALANCA et al., 2011), and energy production (Nor et al., 2014), as well as for emergency management and disaster response planning (Sene, 2008). NWP is based on the very first attempts of operative weather prediction. Since than this approach was further improved but the core idea of using physical laws remained the same. According to Lynch (2007), the effective timeframe of deterministic forecasting extends by approximately one day every ten years. This chapter provides a concise overview of the historical development and present state of numerical weather prediction. Its purpose is to enhance comprehension regarding the continual improvement and enduring relevance of this approach. Furthermore, it aims to shed light on the potential benefits of extracting the physics-informed component of NWP and combining it with a data-driven approach.

2.1.1 History of Numerical Weather Prediction

Richardson (1922) conducted the first numerical approach to predict the weather in the 1920s. He was inspired by Bjerknes (1904) two-step plan for rational forecasting involving a quantitative analysis of the state of the atmosphere through charts valid at an initial time, followed by a systematic graphical processing method to deduce how this state would evolve over time. Although Bjerknes did not develop a numerical approach for his algorithm, Richardson proposed a method to represent the physical principles that regulate the atmosphere's behavior as a system of mathematical equations, and to use his finite difference method to solve this system (Lynch, 2007). To demonstrate his model, Richardson aimed to showcase its functionality using an example. It took him six weeks to calculate the change in weather for two locations within the next six hours. The resulting error was so significant that the outcome was virtually unusable. Nonetheless, this marked the first attempt at a numerical weather forecast. Richardson (1922) concluded by presenting a vision in which an organization of 64,000 "computers" - at the time, this term still referred to people - could be used to calculate global weather so quickly that weather forecasting would indeed become possible. Only with the invention of the computer as a machine did Richardson's vision become practical.

In 1950, Jule Charney, an American meteorologist, led a team that achieved the first successful numerical weather prediction using a digital computer, namely the ENIAC (Electronic Numerical Integrator and Computer). This team utilized a simplified version of atmospheric dynamics that reduced the computation time and memory required for the computations (Cox, 2002).

Although the computation time to generate the first weather forecasts using ENIAC was comparable to the length of the forecast horizon itself, the authors estimate that it would only need a renewed systematic effort and routinization of operations to calculate a 24-hour prediction within 12 hours (Charney et al., 1950).

This breakthrough paved the way for further advancements in computer-based weather forecasting technology. The use of operational weather forecasts, which refers to routine predictions for practical purposes, started in 1954 when a team in Stockholm led by Carl-Gustav Rossby produced a forecast based on barotropic equations (Harper et al., 2007). On July 1, 1954, the U.S. Weather Bureau, U.S. Air Force, and U.S. Navy came together to establish the Joint Numerical Weather Prediction Unit (JNWPU), which was tasked with the responsibility of applying advanced computer technology to the production of weather forecasts for operational use (Morone & Carmeyia Gillis, 2007). In 1959, the Japan Meteorological Agency (JMA) carried out their first operational NWP based forecast using a Northern

Hemisphere Balance Barotropic model (Nitta & Saito, 2004). 10 years later, the Australian Bureau of Meteorology conducted its first real-time forecasts (Leslie & Dietachmayer, 1992).

During the late 1960s, the Geophysical Fluid Dynamics Laboratory of the National Oceanic and Atmospheric Administration (NOAA) created an original climate model of general circulation that integrated both atmospheric and oceanic processes, enabling researchers to enhance their understanding of the interplay between these two systems (NOAA, 2021).

2.1.2 Current State and Trends in Numerical Weather Prediction

As a fluid medium, the atmosphere is subject to the principles of fluid dynamics and thermodynamics. In light of this, numerical weather prediction endeavors to gather information on the atmospheric conditions at a specific point in time and utilize these principles to forecast future atmospheric states. Modern weather prediction is heavily reliant on extensive numerical simulation systems, which are commonly administered by national weather agencies worldwide. These systems follow a complex process chain that is comprised of interdependent steps, each closely interacting with one another. (Schultz et al., 2021)

The present chapter presents a succinct exposition of the state-of-the-art NWP process, offering a broad perspective on the topic. The concluding paragraph emphasizes the currently achieved performance of operative weather prediction. However, for a more comprehensive treatment of the subject with a deeper analysis of the finer details, readers are encouraged to refer to the papers that are cited within.

Data Collection

The collection of a wide range of meteorological observations from various sources worldwide is essential for obtaining the initial state of the earth system, which encompasses the atmosphere, soil, and ocean. Such observations comprise data from weather and radiosonde stations, aircraft and ship measurements, and various types of remote sensing observations from sources such as radar and satellites. Despite the vast quantities of both direct and indirect measurements gathered each day, the resulting observations are often inadequate to fully characterize the complex state of the atmosphere. (Schultz et al., 2021)

Data Assimilation

The central task of the data assimilation (DA) is to create a complete picture of the current state of the atmosphere using the heterogeneous and incomplete measurements. The resulting output is typically a gridded system at kilometer-scale resolution, encompassing the entire Earth which represents an optimized estimation of the actual weather conditions. Modern data assimilation systems predominantly employ three- or four-dimensional variational techniques (3D-Var and 4D-Var) and ensemble methods (often utilizing the Kalman filter) (Bannister, 2017). Kanamitsu (1989) discribes the National Meteorological Center's (NMC) Global Assimilation and Forecast System. A detailed description of data assimilation in ocean models can be found in Anderson et al. (1996).

Numerical Simulation and the Use of Partial Differential Equations

By utilizing the NWP model and inputting the appropriate initial conditions, it is possible to simulate the atmospheric processes. The changes in the atmosphere related to momentum, mass, and enthalpy (amount of heat energy that is absorbed or released by the atmosphere) can be represented by a set of linked partial differential equations, known as the primitive equations (Bjerknes, 1904). These are initialized by observed data and their recent rates of change. The prediction of the atmosphere's future state is based on derivatives that are determined by the model. These rates of change are then applied to the current atmospheric state to determine the state at a future time. This process is repeated in a series of time steps until the desired forecast time is reached. The size of the time step used in the model is related to the distance between points on the computational grid and is chosen to maintain numerical stability (Pielke, 2002, p. 285–287). As mentioned, the calculation behind this simulation is based on the primitive equations. Exact solutions to these equations cannot be derived using analytical methods, with only a few idealized cases being the exception (Pielke, 2002; Strikwerda, 2004, p.65). To be more precise, the Navier-Stokes-Equations, which form a subset of the primitive equations, form one of the seven Millennium Prize problems (Constantin, 2006). The existence of a smooth and unique solution for the general three-dimensional case is currently unknown, as there is no proof or counterexample to support either conclusion (Fefferman, 2000). Thus, numerical methods are utilized to obtain solutions that are only approximations. Over the last decade, finite-difference or finite-volume discretizations on platonic solids projected on the sphere, such as icosahedral (Tomita & Satoh, 2004) or cubed-sphere grids (Ullrich & Jablonowski, 2012), have been developed as alternative numerical methods to global spectral transform models. This is because classical global spectral transform models

are not well-suited for performing approximations on a kilometer-scale within an acceptable time frame, which requires highly parallelizable algorithms for the dynamic core (Satoh et al., 2014). Fortunately, advancements in the development of discretization approaches have allowed for the grid-scale dynamics to follow energy, entropy, and mass conservation laws, while at the same time minimizing the need for numerical filters that suppress artificial numerical models. Supporting these claims, Ringler et al. (2010) and Wood et al. (2014) provide evidence of these advancements, and Gassmann (2018) provides further information on minimizing the need for numerical filters. For an in-depth examination of modern dynamical core architectures, Ullrich and Jablonowski (2012) provide a comprehensive overview.

Post Processing

To generate more accurate forecast products for end-users at a finer scale, a post-processing step is integrated into the numerical weather prediction (NWP) workflow. Common post-processing techniques include converting the vertical axis from sigma-coordinates to pressure levels or geometric height, applying bias corrections, and utilizing statistical methods to remove systematic biases from the NWP output and incorporate local scale adjustments (Schultz et al., 2021). These techniques enhance the accuracy and usefulness of the NWP output for end-users.

State-of-the-Art Performance in Operative NWP

In recent decades, significant progress has been made in improving the predictive capabilities of NWP models for atmospheric states. Present-day global NWP models can forecast synoptic-scale weather patterns for multiple days with remarkable accuracy. As an example, the European Centre for Medium-Range Weather Forecasts Integrated Forecast System yields deterministic forecasts with an 80% anomaly correlation coefficient for the 500 hPa geopotential height for around 7 days, while the root-mean-square error for 2 m temperature predictions for 72-hour forecasts is close to 2 K (Haiden et al., 2018).

Moreover, it has been reported by J.-H. Chen et al. (2019) that high-impact weather phenomena, for instance, the tracks of hurricanes, can be forecast with an accuracy of 150 km for up to four days in the future. It should also be pointed out that models differ not only in their accuracy, but also in their maximum prediction time. The Met Office's Unified Model predicts the weather six days ahead (Chan & Kepert, 2010), while the European Centre for Medium-Range Weather Forecasts model can forecast up to 10 days ahead (Holton, 2004). The Global Forecast System model run by the

Environmental Modeling Center is able to forecast weather up to 16 days in advance (Brown, 2008).

2.2 Deep Learning

At present, major operational weather centers rely on NWP and continue improving it (Bauer et al., 2015). Nonetheless, deep learning offers an promising data-driven alternative that is generating substantial interest. While the application of deep learning in this specific domain is discussed in section 2.3, this chapter provides an general overview. The term deep learning is categorized and generally described. Key architectures are highlighted, followed by a detailed explanation of their underlying concepts.

As visualized in Figure 2.1, deep learning, encompasses a subset of machine learning that resides within the broader domain of Artificial Intelligence. In the context of the wide spectrum of deep learning architectures, the Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) have been identified as highly promising models. (Goodfellow et al., 2016) Consequently, this chapter begins by providing an overview of the overall concepts, subsequently transitioning into a deeper analysis of the subject matter and concluding with a detailed examination of the considered deep learning architectures.

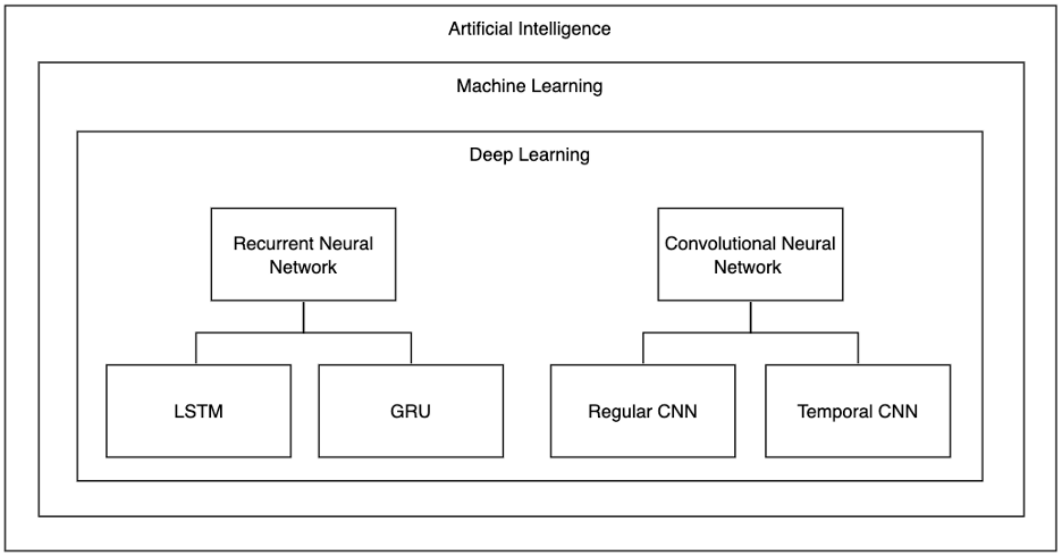


Figure 2.1: Hierarchical classification of deep learning architectures; Inspired by (Goodfellow et al., 2016)

Artificial Intelligence

Artificial Intelligence, as a broad field, encompasses all methodologies aimed at achieving machine intelligence comparable to that of the human brain, while computer scientists define it specifically as the study of "intelligent agents". Those refer to devices that possess the ability to perceive their surroundings and execute actions aimed at maximizing the probability of accomplishing their objectives. Intuitively, the term "artificial intelligence" is employed when a machine possesses the capacity to carry out tasks that humans typically associate with human cognition, such as learning and problem-solving. (Shinde & Shah, 2018)

Machine Learning

Machine learning, classified as a subfield of artificial intelligence, includes all computer programs that learn new knowledge without relying on explicit implementations. Formally, this can be described as:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E. (Mitchell, 1997)

In a multitude of computing applications, machine learning serves as a valuable tool when confronted with the complexity or impracticability of designing and coding explicit algorithms that deliver desirable performance. (Shinde & Shah, 2018)

In those cases, algorithms need to be constructed that can learn from and make predictions on data. These algorithms surpass the constraints of fixed program instructions by employing data-centric predictions or decisions, accomplished through the construction of a model based on available data. (Ongsulee, 2017)

Depending on the usage of labeled data, machine learning approaches are divided into supervised (about 70%) and unsupervised (10 to 20%) algorithms. Semi-supervised learning and reinforcement learning are two additional technologies that are occasionally utilized in certain contexts. Supervised learning algorithms are trained through the utilization of labeled examples, where inputs with known desired outputs are provided. By applying approaches such as classification, regression, prediction, and gradient boosting, supervised learning leverages patterns to anticipate the label values of unlabeled data points. Supervised learning is frequently employed in applications where historical data is utilized to forecast probable future events. Commonly employed supervised learning algorithms include

support vector machines, linear regression, naive Bayes, and artificial neural networks (supervised in the majority of use cases). When labeling is not a viable option or comes with significant expenses, unsupervised learning is the preferred approach since it can handle datasets composed solely of unlabeled data. Unsupervised learning techniques offer the capability to detect segments exhibiting comparable attributes, suggest similar items, and identify anomalies within data. Notable methodologies for these purposes encompass nearest-neighbor mapping, k-means clustering, self-organizing maps, and singular value decomposition. (Ongsulee, 2017)

Deep Learning

Deep learning, which has yielded remarkable advancements in various domains, is predominantly driven by the utilization of artificial neural networks (ANN). The design of ANNs takes inspiration from the work of the neurophysiologists David H. Hubel and Torsten Wiesel in 1959 (Hubel & Wiesel, 1959). Through their groundbreaking investigations of the primary visual cortex, they made a pivotal breakthrough by identifying two distinct cell types: simple cells and complex cells. Artificial neural networks can be conceptualized as sequential models composed of cells, drawing inspiration from the biological observations mentioned. (Ongsulee, 2017)

According to our best knowledge, Aizenberg et al. (2000) firstly used the term "deep learning" in the context of ANNs. Deep learning is a part of machine learning methods that are rooted in learning different representations of data. In the realm of artificial neural networks and machine learning algorithms, the presence of multiple (more than one) hidden layers distinguishes a network as being deep. In general, these neural networks employ a series of interconnected layers consisting of nonlinear processing units to extract and transform features. Each subsequent layer takes the output of the preceding layer as input. Deep networks are designed to learn multiple levels of features or representations from the data, forming a hierarchical representation where higher-level features are derived from lower-level features. (Ongsulee, 2017)

Particularly in the past decade, deep learning approaches have exhibited superior performance compared to traditional models across diverse domains of machine learning. Notable exemplifications include computer vision (Hinton & Salakhutdinov, 2006), speech recognition (Snyder et al., 2018), and natural language processing (Bengio, 2012), as well as scientific disciplines such as physics (Baldi et al., 2014; Bhimji et al., 2018), chemistry (Schütt et al., 2017) and bioinformatics (Z. Chen et al., 2020).

In this project, the focus lies in predicting ocean buoy measurements.

Consequently, an extensive exploration will be undertaken into utilization of deep neural networks in the realm of time series forecasting (TSF), another problem domain where those networks became the dominating approach in recent time. (Schmidhuber, 2015)

Deep neural networks offer a notable advantage over classical statistical models by enabling the modeling of complex non-linear feature interactions, which exceeds the capabilities of linear relationship modeling typically found in traditional approaches. (Reyes & Ventura, 2019)

Additionally, DNNs possess the advantageous capability to adjust directly to the data without making any prior assumptions. This attribute confers significant benefits, especially when confronted with sparse information regarding the time series. (Lara-Benıétez, Carranza-Garcıéa, Garcıéa-Gutiérrez, et al., 2020)

Besides that, the continuous expansion of available data contributed to the advancement of deep learning architectures, leading to notable improvements in forecast accuracy. (Gamboa, 2017)

Lara-Benıétez et al. (2021) conducted a study that commenced with a literature survey to identify the most promising deep learning (DL) architectures for time series forecasting (TSF), which are detailed in Table 2.1. Subsequently, the authors extensively compared these architectures by training a substantial number of models (over $38,000$) on 12 distinct forecasting problems (excluding weather-related scenarios). The results of this experiment highlight the accurate predictive performance of all architectures, with the exception of Multilayer Perceptron (MLP). Moreover, the study additionally demonstrated that the Elman Recurrent Neural Network (ERNN) exhibits inferior performance compared to LSTM and GRU models overall. Echo State Networks (ESN) excel in terms of time efficiency compared to other recurrent neural networks, constituting their primary advantage. Nonetheless, when considering performance, LSTM and GRU models surpass ESNs as well. Consequently, the focus shifts solely to recurrent and convolutional neural networks, which are described in detail in subsequent chapters.

2.2.1 Recurrent Neural Network

As discussed in the general introduction of deep learning in subsubsection 2.2, neural networks draw inspiration from biological observations of the brain. Human cognition maintains continuity, leveraging prior understanding to grasp subsequent information without resetting the thinking process. Human thoughts have persistence, a property basic neural networks do not have, which represents a significant limitation. Recurrent neural net-

Type	Architecture	Advantage	Disadvantage
ANN	Multi-layer perceptron (MLP)	Widely used, good generalization capability	Sensitive to noisy data, may overfit
RNN	Elman Recurrent Neural Networks (ERNN)	Simple architecture, captures sequential dependencies	Limited long-term memory, prone to vanishing gradient
RNN	Long short-term memory (LSTM)	Effective in capturing long-term dependencies	Computationally expensive, complex architecture
RNN	Echo State Network (ESN)	Fast training, efficient in time series forecasting	Performance may vary based on reservoir initialization
RNN	Gated recurrent unit (GRU)	Balances performance and complexity	May struggle with very long-term dependencies
CNN	Basic convolutional neural network (CNN)	Effective in capturing spatial patterns	May overlook temporal dependencies
CNN	Temporal convolutional network (TCN)	Captures long-range dependencies, parallel processing	Sensitive to input sequence length

Table 2.1: DL Architectures comparison

works offer a solution to this matter by incorporating loops that enable the enduring retention of information. (Olah, 2015)

By connecting each time step to its previous counterparts, recurrent neural networks adeptly capture the temporal dependency exhibited by the data. The network processes observations sequentially, assimilating knowledge about preceding observations and their relevance to forecasting. This enables the network to learn patterns not only between input and output but also internal patterns among the observations within the sequence. (Lara-Benítez et al., 2021)

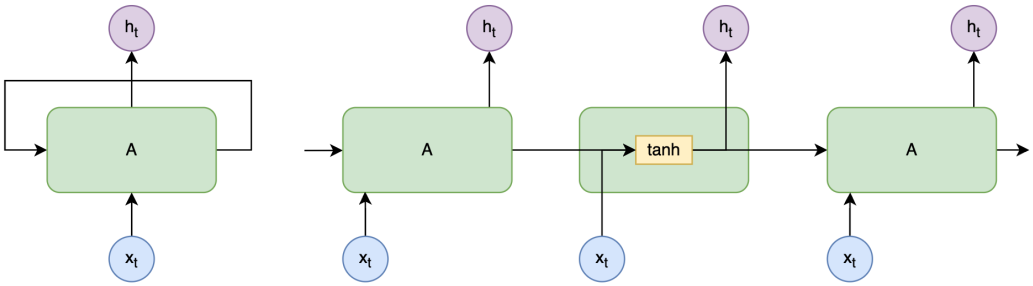


Figure 2.2: Standard recurrent neural network: A repeating module

The left side of Figure 2.2 illustrates a segment of the neural network, labeled A, which processes an input x_t and produces an output value h_t. By

incorporating a loop mechanism, information can be passed from one step of the network to the subsequent step. The right side of Figure 2.2 demonstrates that a recurrent neural network can be conceptualized as multiple replicas of the identical network, with each copy conveying a message to its succeeding counterpart. The chain-like structure of recurrent neural networks unveils their close association with sequences and lists, making them the natural choice of architecture for processing such data. (Olah, 2015)

Nevertheless, traditional RNNs can be challenging to train as they suffer from the problem of gradient vanishing and exploding, particularly when attempting to model long-term dependencies (Bengio et al., 1994; Hochreiter, 1991). In order to overcome these limitations, advanced RNN architectures, such as Long Short-Term Memory (LSTM) (Gers & Schmidhuber, 2000; Hochreiter & Schmidhuber, 1997) and Gated Recurrent Units (GRU) (Chung et al., 2014) have been proposed. These architectures incorporate gating mechanisms to selectively retain or forget information from previous time steps, making them more effective at modeling long-term dependencies in time series data.

Long Short Term Memory

In an attempt to mitigate the issues of vanishing and exploding gradients, Hochreiter and Schmidhuber (1997) introduced the Long Short-Term Memory (LSTM) model. By leveraging LSTM's unique architecture, temporal dependencies can be modeled over extended periods without disregarding short-term patterns. This capacity empowers LSTMs to effectively capture and represent complex sequential information.

Similar to conventional RNNs, LSTM networks also exhibit a sequential structure. However, their fundamental building block differs from a standard RNN as modern LSTMs comprises four distinct internal components instead of one. In its original formulation by Hochreiter and Schmidhuber (1997), the LSTM architecture comprised a memory cell, input gate, and output gate. However, an enhanced version of LSTM was introduced by Gers et al. (2000), which introduced a forget gate that enables an LSTM cell to learn to reset itself at appropriate times.

In Figure 2.3, the repeating module, consisting of four interconnected layers, is visually represented. The subsequent textual description provides a detailed account of the interactions and relationships among these components.

Central to the functioning of LSTMs is the cell state, represented by a horizontal line across the top of the diagram. The crucial aspect lies in the precise control exerted by the three gates, enabling the selective addition or

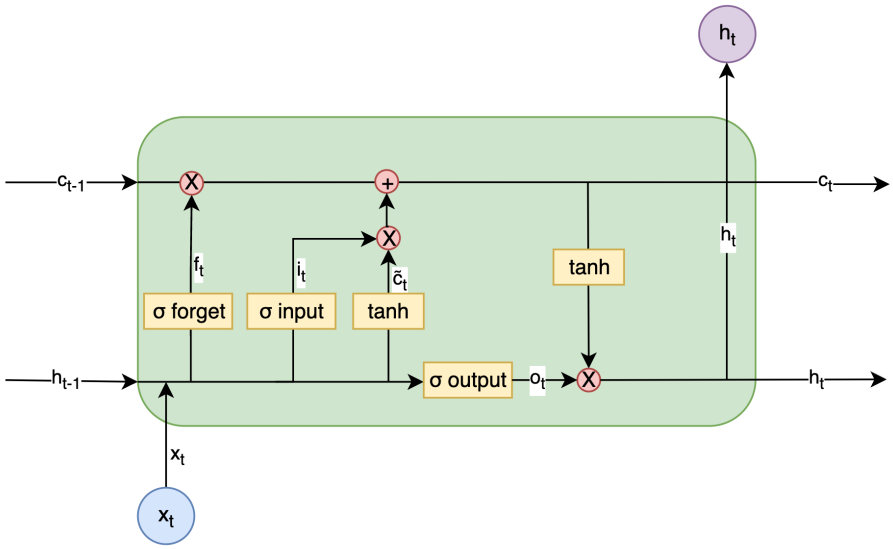


Figure 2.3: Long Short Term Memory cell and its operation

removal of information from the cell state. These gates are constructed using a sigmoid neural network layer, generating a weighting parameter within the range of 0 to 1, which is then combined with a pointwise multiplication operation.

To initiate the LSTM procedure, the initial focus lies in identifying the information that should be omitted from the cell state. This pivotal decision is entrusted to the forget gate layer. Through the examination of the previous hidden state h_{t-1} and the current input x_t, the forget gate layer produces a distinct weighting parameter f_t for each component of the cell state c_{t-1}. In the next step, the information that is added to the cell state is determined. Therefore, a sigmoid layer called the input gate layer selects the values to be updated. Next, a *tanh* layer creates a vector of new candidate values, $\tilde{c}_t$, that could be added to the cell state. Then, the outputs of these network layers are combined by multiplication.

The subsequent phase revolves around the decision-making process concerning the inclusion of fresh information within the cell state. This procedure entails two primary elements. Firstly, the input gate layer, characterized by a sigmoid layer, assesses and determines the specific values necessitating updates. Secondly, a *tanh* layer produces a vector of potential new candidate values, designated as $\tilde{c}_t$, that have the potential to supplement the state. In the subsequent step, these two elements are seamlessly combined to generate an updated state. The progression now proceeds to update the previous cell state, c_{t-1}, to the current cell state, c_t. This update involves two essential operations. Firstly, the old state is multiplied by the forget gate

value, f_t, enabling the elimination of the previously identified irrelevant information. Subsequently, the product of the input gate value, i_t, and the vector of new candidate values, $\tilde{c}_t$, is added to the multiplication outcome. This addition encompasses the scaled incorporation of the new candidate values based on the determined update magnitude for each state value.

Finally, the decision regarding the output selection becomes crucial. The generated output is a refined representation derived from the cell state. Initially, a sigmoid layer referred to as the output gate layer is employed on h_{t-1} to determine the specific components of the cell state that will contribute to the output. Furthermore, the cell state c_t is subjected to a hyperbolic tangent (*tanh*) activation function, which restricts the values to a range between -1 and 1. Subsequently, the transformed cell state is multiplied by the output of the sigmoid gate, effectively filtering and selectively outputting only the chosen components. (Lara-Benıétez et al., 2021; Olah, 2015; Zargar, 2021)

Reference	**Year**	**Technique**	**Outperforms**	**Domain**
Ma et al., 2015a	2015	LSTM	MLP, NARX, SVM	Traffic speed
Y. Tian and Pan, 2015	2015	LSTM	MLP, Autoencoders	Traffic flow
Fischer and Krauss, 2018	2018	LSTM	Logistic regression, MLP, RF	S&P 500 index
Bouktif et al., 2018	2018	LSTM	Linear regression, kNN, RF, MLP	Electric load
Sagheer and Kotb, 2019	2019	LSTM	ARIMA, ERNN, GRU	Petroleum production
Pan et al., 2019	2019	LSTM + Attention	LSTM, MLP	Solar generation
Smyl, 2020	2020	Hybrid ETS-LSTM	Competition winner	M4 competition
Bandara et al., 2020	2020	Clustering + LSTM	LSTM, ARIMA, ETS	CIF2016 and NN5 competitions

Table 2.2: Relevant studies on TSF using LSTM

The latest developments in deep learning for time series forecasting (TSF) are extensively reviewed by Lara-Benıétez et al. (2021). A detailed account of the outcomes associated with Long Short-Term Memory is provided in Table 2.2, while the findings of three randomly selected studies will be further discussed.

In the study conducted by Ma et al. (2015b), the application of LSTM was explored within the context of short-term traffic prediction. The findings demonstrated that LSTM effectively addressed the challenge of back-propagated error decay through the utilization of memory blocks, thereby exhibiting remarkable capabilities for predicting time series data with extensive temporal dependencies. Moreover, LSTM models demonstrated the inherent ability to automatically determine optimal time lags. A comparative analysis involving parametric and non-parametric algorithms revealed that LSTM models achieved the highest level of prediction performance in terms of accuracy and stability. In addition, LSTM networks were employed by Fischer and Krauss (2018) to forecast out-of-sample movements for the individual stocks comprising the S&P 500. The study revealed that LSTM networks surpassed memory-free classification methods, traditional neural networks, and logistic regression in terms of predictive performance. Furthermore, also Sagheer and Kotb (2019) undertook a comprehensive analysis of the performance of LSTM networks in the context of time series forecasting. The evaluation was carried out through the implementation of two case studies involving production data from real oilfields. In order to ensure a rigorous assessment, the proposed deep LSTM model was compared against multiple standard methods, encompassing statistical and soft computing techniques. The empirical results, obtained through the utilization of various measurement criteria, clearly indicated that the LSTM model outperformed other standard approaches.

Gated Recurrent Unit

Gated Recurrent Units (GRU) were proposed by Cho et al. (2014) as an alternative approach for addressing the challenges posed by the exploding gradient and vanishing gradient phenomena. GRU units can be viewed as a simplification of LSTM units, characterized by two key simplifications:

1. Merging of the cell state and hidden state
2. Combination of the input and forget gates into one unified gate known as the update gate.

By virtue of these modifications, the number of trainable parameters in GRU networks is reduced, resulting in improved computational efficiency without sacrificing performance, as demonstrated in Ravanelli et al. (2018).

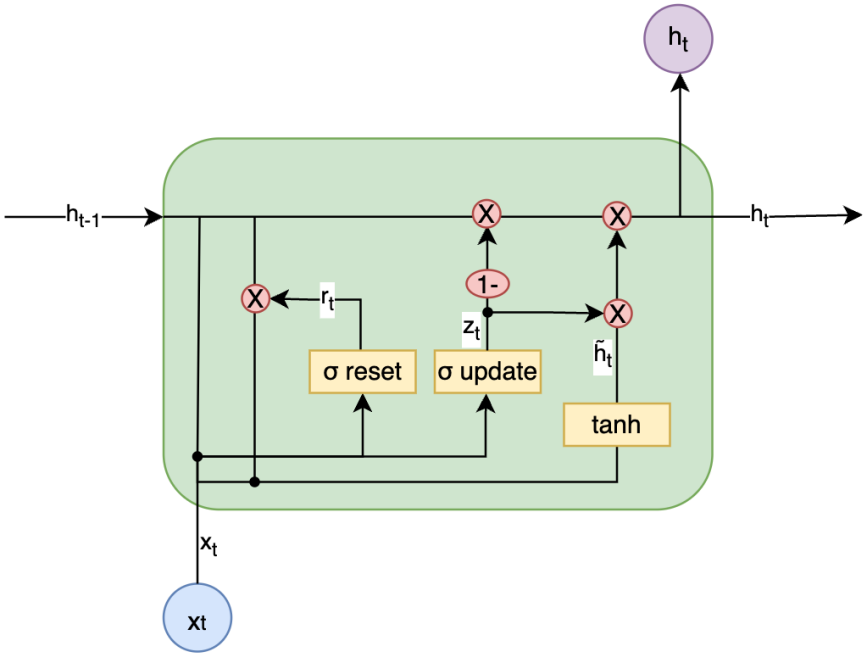


Figure 2.4: Gated Recurrent Unit cell and its operation

As demonstrated in Figure 2.4, the GRU cell incorporates an update gate and a reset gate. The update gate is responsible for determining the proportion of past information that should be carried forward to future steps. On the other hand, the reset gate governs the extent to which past information should be disregarded or forgotten. (Lara-Benıétez et al., 2021; Zargar, 2021)

As part of their research on electricity price forecasting, Ugurlu et al. (2018) employed multi-layer GRU models as an effective technique in their analysis. Multiple algorithms have been trained on the data of the Turkish day-ahead marked. Remarkably, their three-layered GRU model emerged as the top performer, surpassing all alternative neural network structures and cutting-edge statistical techniques in a statistically significant manner. Table 2.3 provides an overview of additional studies that demonstrate the strong performance of GRU in the domain of Time Series Forecasting (TSF). (Lara-Benıétez et al., 2021)

Reference	Year	Technique	Outperforms	Domain
Chung et al., 2014	2014	GRU	ERNN	Music and speech signal modeling
Kuan et al., 2017	2017	GRU	LSTM, GRU	Electricity load
Wang et al., 2018	2018	GRU	LSTM, SVM, ARIMA	Photovoltaic power
Ugurlu et al., 2018	2018	GRU	MLP, CNN, LSTM	Electricity price

Table 2.3: Relevant studies on TSF using GRU

2.2.2 Convolutional Neural Network

During the 1980s, Kunihiko Fukushima introduced the convolution process to the field of convolutional neural networks (CNN) with his creation called the neocognitron (Fukushima, 1988). This innovative concept drew inspiration from Hubel and Wiesel (1959). Furthermore, Yann Lecun significantly contributed to the advancement of CNNs by developing LeNet-5, a seven-level convolutional network. LeNet-5 employed backpropagation and adaptive weights for parameter optimization (Lecun et al., 1998; LeCun et al., 1989). Present-day major CNN architectures are derived from those principles (Ajit et al., 2020).

CNNs have recently gained a lot of interest and have shown impressive performance in domains such as computer vision (Bhatt et al., 2021), natural language processing (Jozefowicz et al., 2016) and autonomous vehicles (Dreossi et al., 2017; Farag & Saleh, 2018). The effectiveness of convolutional neural networks (CNNs) extends to all tasks involving data with high local correlation, as they excel at capturing the presence of the same pattern across diverse regions. Moreover, CNNs demonstrate effectiveness in handling high-dimensional data, leveraging their shared-weights architecture and translation invariance characteristics. (C. Tian et al., 2018)

The utilization of CNNs in the analysis of time series data is driven by the prospect of acquiring filters that encode repeated patterns within the series, treating sequence data as a one-dimensional image to extract features through the convolutional operation. These filters enable the CNNs to make forecasts regarding future values. Furthermore, CNNs possess the capability to autonomously learn and extract features from the raw data, eliminating the need for prior knowledge or feature engineering. This attribute enhances their efficacy in effectively handling noisy time series by iteratively discarding noise at each layer. The resulting hierarchical structure facilitates the extraction of meaningful features exclusively. (Borovykh et al.,

2017)

Basic CNN Architecture

The typical architecture of convolutional neural networks (CNNs) commonly involves a sequential arrangement of convolutional and pooling layers, which is subsequently followed by fully connected layers. The convolutional layers play a pivotal role and employ two essential techniques, namely local connectivity and weight sharing. Local connectivity refers to the specific arrangement where each convolutional node establishes connections with a small subset of inputs. This subset, referred to as the receptive field, represents a localized region within the input data. In contrast to standard feedforward neural networks, where all inputs are connected to every node, local connectivity allows CNNs to focus on extracting features from specific regions of the input. Furthermore, convolutions serve as a substitute for the weighted sums in standard neural networks. To be specific, in each convolutional layer, the input undergoes convolution with a weight matrix of predetermined dimensions, referred to as a filter or sliding window. This process results in the computation of a feature map by sliding the weight matrix over the input and calculating the scalar product between the input and weight matrix. Within one convolutional layer, the neurons exhibit weight sharing, meaning that they possess identical weights which allows them to collectively identify the same pattern, albeit in distinct regions of the input. The utilization of weight sharing, and local connectivity leads to a decrease in the overall number of weights that are required to be learned and stored. Consequently, this reduction facilitates faster training. The implementation of the convolutional and weight sharing mechanisms is accomplished by utilizing a filter with a specific kernel size. The number of nodes that share the same weights is determined by the chosen kernel size. Following a convolutional layer, a subsequent max-pooling layer calculates the maximum value among a group of neighboring neurons selected from the preceding convolutional layer. The incorporation of those two layers guarantees that the output from the max-pooling layer remains unaffected by shifts in the input data, which is an advantageous characteristic for handling real-world data. In order to compute the final result, a fully-connected layer is utilized, which consolidates the local features and converts them into global features. (Koprinska et al., 2018)

To facilitate a better understanding of this architecture, Figure 2.5 visually represents the LeNet-5 architecture mentioned at the beginning of this section. It comprises an input layer, two sets of convolution and pooling layers, and concludes with two fully connected layers and an output layer.

The utilization of convolutional neural network (CNN) models in the

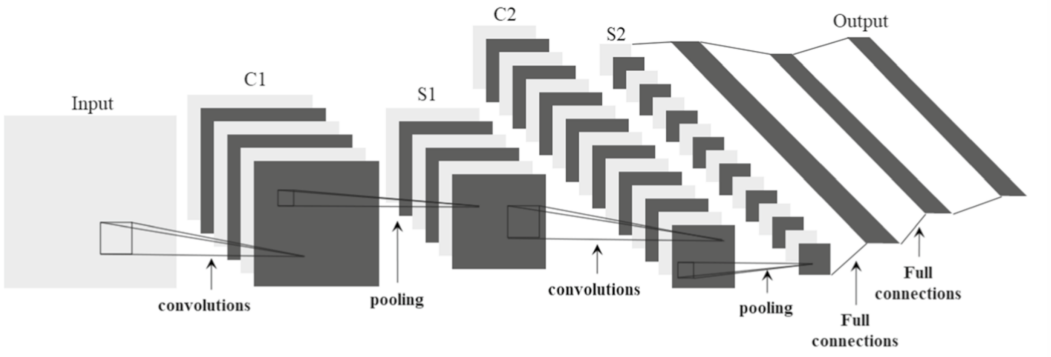


Figure 2.5: LeNet-5 architecture as an example of a standard CNN (Cong & Zhou, 2023)

realm of time series forecasting (TSF) literature has been relatively limited compared to the predominant focus on recurrent neural networks (RNNs). Nonetheless, a number of studies have put forth the adoption of CNNs either as standalone feature extractors or in tandem with recurrent blocks to facilitate accurate predictions. As indicated by Lara-Benıétez et al. (2021), the studies of utmost significance regarding these propositions are documented in Table 2.4.

Reference	Year	Technique	Outperforms	Domain
Tsantekidis et al., 2017	2017	CNN	SVM, MLP	Stock price
Kuo and Huang, 2018	2018	CNN	LSTM, MLP	Energy load
Koprinska et al., 2018	2018	CNN	LSTM	Solar power and electricity
C. Tian et al., 2018	2018	Hybrid CNN-LSTM	RNN and LSTM individually	Electricity
Liu et al., 2018	2018	Ensemble CNN-LSTM	ARIMA, ERNN, RBF	Wind speed
Cai et al., 2019	2019	CNN	RNN, ARIMAX	Building load
Shen et al., 2020	2019	Hybrid CNN-LSTM	LSTM, CNN	Financial data

Table 2.4: Relevant studies on TSF using CNN

Temporal Convolutional Network

Despite the exhibited potential of CNNs, recurrent architectures are commonly perceived as the default initial approach for tasks involving sequence

modeling. The work by Bai et al. (2018) challenges this assertion and presents an examination of Temporal Convolutional Networks (TCN), a distinct architecture family within CNNs, positing it as a more advantageous substitute. Their study reveals that TCNs demonstrate superior performance when compared to LSTM and GRU architectures, even in tasks that are widely accepted as standard benchmarks for evaluating RNNs. Given these promising findings, there is strong motivation to incorporate TCNs as an additional architectural variant in this study.

The TCN architecture draws insights from contemporary convolutional architectures tailored for sequential data, yet it sets itself apart from all existing models. Its design is rooted in fundamental principles, aiming to seamlessly integrate simplicity, autoregressive prediction, and the capacity for extensive memory retention.

The first defining characteristic of TCNs is their capacity to accept input sequences of any length and generate output sequences of equal length, exhibiting a behavior similar to that of Recurrent Neural Networks (RNNs). In order to accomplish this, the TCN utilizes a 1D fully-convolutional network (FCN) architecture as described in (Long et al., 2015). There the size of each hidden layer is identical to that of the input layer. Moreover, the inclusion of zero padding, with a designated length, ensures consistency in layer lengths across successive stages. The second critical characteristic entails the preservation of temporal causality, thereby avoiding the transmission of future information to past instances. This objective is accomplished by employing causal convolutions, where the convolution operation for each output at time `t` only considers elements from time `t` and earlier in the preceding layer, maintaining a strict temporal dependency. To establish the capability for the networks to glean insights from a significantly distant past when making predictions, a combination of very deeply stacked networks and dilated convolutions is employed. Incorporating dilated convolutions, as introduced by Oord et al. (2016), facilitates an extension of the network's receptive field. This mechanism allows the convolved neurons to cover a larger region of the input, eliminating the need for pooling operations. As a result, the network maintains its resolution, ensuring the retention of high-level details and avoiding any potential loss of resolution. In addition, TCNs incorporates residual connections to enable the expansion of network depth, thus enhancing its ability to handle a substantial history size more effectively. By employing residual connections, TCN establishes direct paths that facilitate the flow of information through the network, enabling efficient utilization of the network's capacity and ensuring its capability to effectively capture long-term dependencies.

Temporal Convolutional Networks (TCNs) have demonstrated several

advantages over conventional Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). These advantages include reduced memory requirements during training, attributed to the utilization of shared convolutional filters. Unlike RNNs that process input sequences sequentially, TCNs employ parallel convolutions, enabling efficient processing of long input sequences. Additionally, TCNs exhibit a more stable training scheme, effectively mitigating the occurrence of vanishing gradient problems commonly observed in RNNs. (Bai et al., 2018)

Table 2.5 provides a compilation of studies that highlight the successful application of TCNs in the context of time series forecasting. This compilation of empirical evidence further underscores the growing recognition and acceptance of TCNs as a favorable approach for analyzing and predicting temporal data patterns.

Reference	**Year**	**Technique**	**Outperforms**	**Domain**
Sun et al., 2022	2019	TCN	LSTM	Financial data
Y. Chen et al., 2020	2019	Encoder-decoder TCN	RNN	Retail sales
Wan et al., 2019	2019	TCN	LSTM, ConvLSTM	Meteorology
Lara-Benıétez, Carranza-Garcıéa, Luna-Romera, et al., 2020	2020	TCN	LSTM	Energy demand

Table 2.5: Relevant studies on TSF using TCN

2.3 Deep Learning Weather Prediction

Having comprehended the preeminence of Numerical Weather Prediction (NWP) as an advanced technology for weather forecasting, as well as the immense promise of deep learning, this section focuses on elucidating the present-day utilization of deep learning techniques within the domain of weather prediction.

Deep Learning Weather Prediction (DLWP) is a data-driven approach that uses weather observations as inputs for deep neural network (DNN) models to predict future weather conditions. Based on the characteristics of meteorological data, various neural network architectures are appropriate for the DLWP approach. (Schultz et al., 2021)

Meteorological data is commonly available in multi-dimensional real-type formats, where a single weather parameter can be represented by a four-dimensional array (latitude, longitude, level, and time). Autoencoders are typically used to decrease data dimensionality and are well-suited to handling real-type data (Baldi, 2012; Kamyshanska & Memisevic, 2015). As a result, autoencoders and their variations offer a distinctive advantage when processing meteorological data collections.

As discussed in subsection 2.2.2, convolutional neural network is a type of deep neural network that operates on the basis of feature representation and is specifically designed for image processing. As such, this architecture is widely utilized for identifying spatial relationships in satellite images (X. X. Zhu et al., 2017).

Meteorological data, such as buoy measurements, often comprise extensive amounts of long-term sequential data spanning multiple years or decades with temporal relationships between data elements. In accordance with the comprehensive discussion presented in subsubsection 2.2, various deep learning architectures have been identified as suitable for this task. Nevertheless, the most promising results can be achieved by either hybrid architectures composed of the basic DL models to capture more complex temporal and spatial features or coupling architectures of DL and NWP models as detailed in the following two subsections.

2.3.1 Completely Data-Driven Hybrid Architectures

Shi et al. (2015) introduce ConvLSTM networks for precipitation nowcasting, which incorporate a convolutional structure within the LSTM cell. In contrast to fully connected LSTMs, the encoding component of a ConvLSTM captures the spatio-temporal relationships of meteorological data and results in enhanced forecast accuracy. Nevertheless, ConvLSTM is not able to model spatially varying relationships.

In Shi et al. (2017), a trajectory gated recurrent unit (TrajGRU) model was introduced to enable the active learning of location-variant structures in natural motion and transformation. TrajGRU generates a local neighborhood set for each spatial-temporal stamp by taking into account both the present input and prior states.

Google Research recently introduced DLWP models for high-resolution precipitation nowcasting to forecast precipitation rates. In Agrawal et al. (2019), an image-to-image translation approach is employed using a ubiquitous U-NetCNN. An improved neural weather model (NWM) called MetNet is subsequently presented in Sønderby et al. (2020), which utilizes axial

self-attention mechanisms. The convolutional LSTM block from Shi et al. (2015) is employed to handle the downsampled time slices in the temporal direction. MetNet represents the first DLWP model that surpasses NWP in terms of accuracy and prediction quality at a particular spatiotemporal scale.

2.3.2 Theory-guided Coupling Architectures

Despite their effectiveness, fully data-driven DNN models are not capable of causal discovery from observational data (Runge et al., 2015) and model the entire complex weather system without incorporating scientific prior knowledge. Physics-Informed Neural Networks (PINNs) combine the data-driven and the theory-driven approach. They use supervised neural networks to learn the model but also incorporate physics equations to encourage consistency and interpretability with the known physics of the system.

For example, this was firstly done by Frnda et al. (2019) when proposing an end-to-end DL model for weather forecasting that integrates historical data and prior knowledge from NWP using an effective information fusion mechanism based on LSTM Autoencoder (Srivastava et al., 2015). The approach enables the prediction of multiple meteorological variables and incorporates a novel negative log-likelihood error loss function that allows for both single-value forecasting and uncertainty quantification.

In an attempt to account for the spatio-temporal dependencies among meteorological attributes, Grover et al. (2015) have proposed a hybrid architecture that includes a set of individual bottom-up predictors for each attribute trained on historical data, physical constraints ensuring spatial smoothness of the output, and a deep belief network comprising stacked Restricted Boltzmann Machines modeling joint statistical relations. The benefit of this approach is that it enforces long-term spatial dependencies, which allows for optimization of the predictive model to align with large-scale phenomena.

Incorporating physical laws can also be realized by using customized loss functions, also known as physics-based regularization. The loss function is usually a mean-square error or root-mean-square error loss that compares the predictions of the model to the ground truth during training to optimize the model. Physics-based terms are added to the loss function and act as regularization. These terms can be adjusted by hyper-parameters to control their relative importance compared to the standard MSE loss. Those physics-informed loss functions can be utilized to avoid over-fitting and tackle ill-posed problems. (Kashinath et al., 2021)

Raissi, Perdikaris, and Karniadakis (2019) have introduced PINNs to solve nonlinear partial differential equations (PDE) in fluid dynamics, quantum mechanics, reaction-diffusion systems, and nonlinear wave dynamics. Furthermore, Raissi, Perdikaris, and Karniadakis (2019) point out that PINNs can be trained using small data sets but still result in a robust model. This is an important key property when it comes to use cases where data acquisition may be prohibitive but also when, like in the use case of this project, the available data is noisy, sparse, and incomplete.

Moreover, Y. Zhu et al. (2019) is working on surrogate modeling of transient PDEs in turbulent flows, Beucler et al. (2019) use physics-based regularization to penalize the violation of conservation laws and Daw et al. (2017) are using customized loss functions for modeling lake temperatures.

Those interested in further approaches that integrate both physics- and data-driven methods can find a comprehensive overview in the Case Study by Kashinath et al. (2021).

2.3.3 NWP and DLWP: Advantages and Limitations

DLWP models outpace NWP models in terms of speed at the same resolution (Reichstein et al., 2019), as evidenced by MetNet's ability to provide subsecond latency (Sønderby et al., 2020). By contrast, NWP models require much longer processing times, taking anywhere from ten minutes to several hours for the same tasks.

Additionally, the resolution of data-driven DLWP models does not impact their performance, whereas NWP models require eight times more computation to double the resolution (Bauer et al., 2015). This explains why high-performance computing and parallel computing technology are essential for NWP models (Fu et al., 2017; Ren et al., 2019; Xue et al., 2014), while DLWP models can rely on standard GPUs, TPUs (Jouppi et al., 2017), or specialized AI hardware (Pei et al., 2019) for most neural network training.

Ren et al. (2021) point out that DLWP results in higher forecast accuracy and timeliness compared to NWP for short-term and small-scale domains. Due to large number of tunable hyper parameters, training a DNN for long-range or global forecasting would entail significant levels of complexity similar or even greater than that of NWP. The performance of NWP and DLWP dependent on spatial and temporal scales is illustrated in Figure 2.6 by (Ren et al., 2021).

NWP models offer the potential for extrapolation beyond observed conditions as they predict future atmospheric behavior based on the current state and physics principles (Frnda et al., 2019; Reichstein et al., 2019). In contrast,

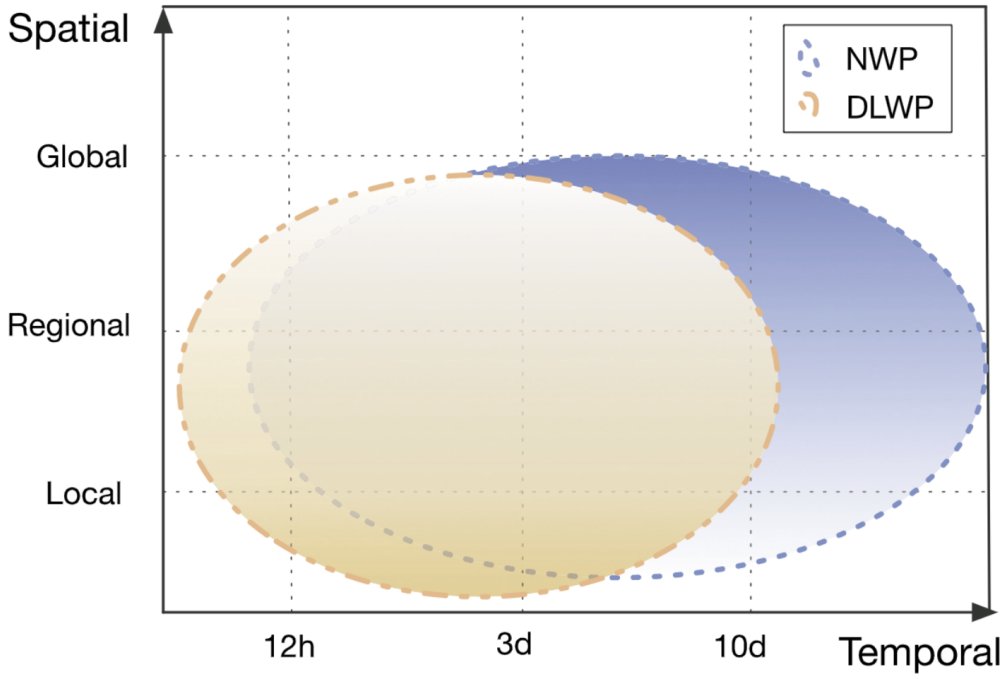


Figure 2.6: Performance comparison between DLWP and NWP at different temporal and spatial scales. In the area covered by each approach, the darker the color, the better the performance (Ren et al., 2021).

interpretability has been identified as a possible weakness of DL, which is caused by the presence of hidden parameters (Montavon et al., 2018).

2.4 Data Sources

The aim of this project is to improve accuracy of ocean buoy measurement forecasts through physical-informed regularization. The primary data source utilized in this study is the Standard Meteorological Data provided by the National Data Buoy Center (NDBC), a division of the National Oceanic and Atmospheric Administration (NOAA). The inclusion of physically informed data is expected to enhance the accuracy of the forecasts. In pursuit of this, the latest reanalysis dataset from the European Centre for Medium-Range Weather Forecasts (ECMWF) is employed. This dataset combines observational data and numerical weather prediction (NWP) models, creating a coherent and extensive physical-informed dataset.

2.4.1 National Data Buoy Center

For scientific purposes, the National Data Buoy Center (NDBC) operates a server that acts as a repository for comprehensive meteorological and hydrological datasets. This server encompasses a collection of historical records and written materials generated by the National Weather Service (NWS) and received from other authoritative entities. In total, 1311 stations are deployed; out of those 243 are owned and maintained by NDBC. (NDBC, n.d.-c)

The data of other stations is provided by their partners, such as universities and other governmental agencies. While the data collected by NDBC are the most accurate to date, as they have been subject to manual quality control by the Mission Control Center (MCC) data analysts, NDBC does not perform quality control on this partner data. (Hall & Jensen, 2021)

Historical Standard Meteorological Data

NDBC provides a repository[1] of all provided datasets on their website. Within the framework of this project's experiments, the utilization of standard meteorological historical data (STDMET) is employed. Post-processing analysis has been performed on these files. Furthermore, the format of those files is similar to the real-time data files except for how missing data is represented. While those are represented by variable numbers of 9's (like 999.0, 99.0) in the historical data, missed values are marked by "MM" in the real-time data files. This allows an easy adoption of scripts and models to be used for real-time data. The STDMET dataset is composed of multiple .txt files, with each file representing the data pertaining to a specific station and year. Each file can be accessed by the five-digit station identifier, assigned by the World Meteorological Organization (WMO). For instance, the data from station 41117 for the year 2020 is denoted by the filename `41117h2020.txt.gz` and can be retrieved when appended to the STDMET base URL[2]. Since 1970, the NDBC has undertaken extensive data collection initiatives. However, it is imperative to recognize that the availability of data for all years is not consistent across all buoys within their network. In each available file, the first two rows can be considered headers, whereas the first line represents the column header, and the second line represents the used (generally metric) unit. All lines that are not data lines begin with the character "#" and may be considered to be comments. For each record, the first five columns represent the timestamp of the measurement in coordinated universal time (UTC). (NDBC, n.d.-b)

[1] https://www.ndbc.noaa.gov/data/

[2] https://www.ndbc.noaa.gov/data/historical/stdmet/41117h2020.txt.gz

#YY	MM	DD	hh	mm	WDIR	WSPD	GST	WVHT	DPD	APD	MWD	PRES	ATMP	WTMP	DEWP	VIS	TIDE
#yr	mo	dy	hr	mn	degT	m/s	m/s	m	sec	sec	degT	hPa	degC	degC	degC	mi	ft
2009	7	1	0	0	130	7.0	99.0	99.00	99.00	99.00	999	1019.0	19.2	21.0	13.5	99.0	99.00
2009	7	1	1	0	131	6.7	99.0	99.00	99.00	99.00	999	1020.0	19.2	21.0	13.5	99.0	99.00

Table 2.6: Data header of: Standard Meteorological Data (NDBC, n.d.-b)

Table 2.6 provides an illustration of the header content found in the STDMET data files. For an in-depth exploration of each feature, a detailed discussion can be found in subsection 2.4.3.

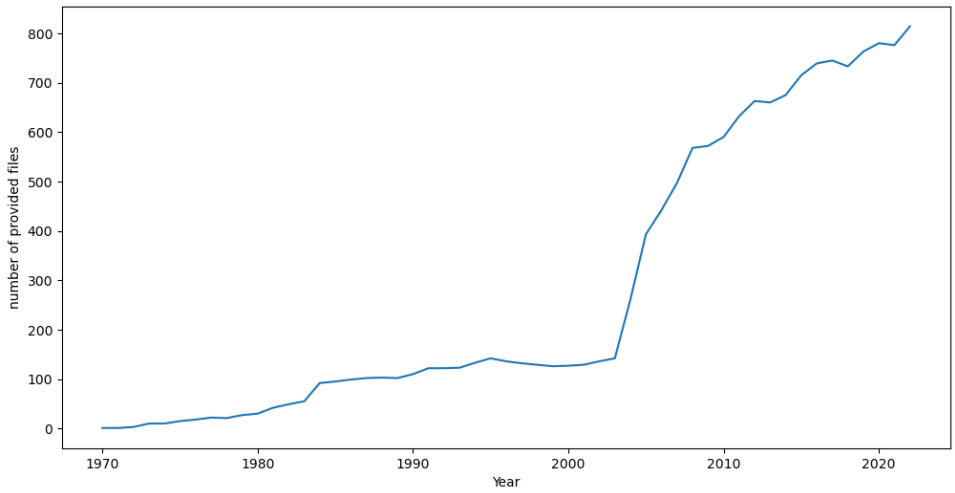
NDBC Metadata

Historical STDMET files do only provide observed measurements over time for the corresponding station and year but no metadata. However, NDBC is providing a metadata file[3] which also includes the station movement over time. Without a detailed analysis, this movement is expected to be negligible. Furthermore, this file does only provide the location of 433 stations. Alternatively, NDBC provides a "station page" for each station on their website which includes the most recent location as well as additional metadata.

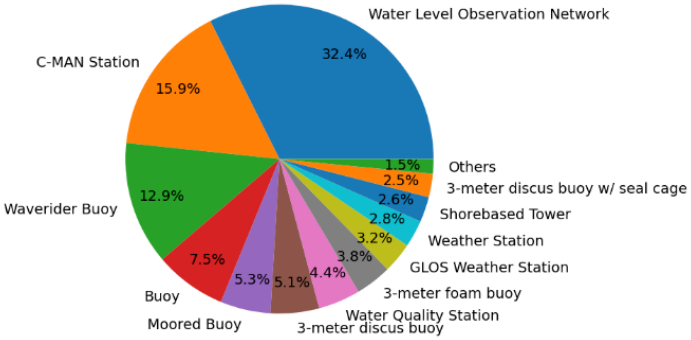
In-depth information regarding the gathering of metadata including file availability and station coordinates, operator and types can be found in subsubsection 4.4.1. An analysis of the generated metadata file shows that among the 1311 stations that were deployed, a total of 1247 stations contribute 14724 datafiles. The chart in Figure 2.7a demonstrates the progression of the number of provided files over time, clearly indicating an upward trend in the collection of data throughout the years. Furthermore, Figure 2.7b and Figure 2.7c offer graphical representations that depict the distribution of station ownership and station types, respectively.

Beyond the previously mentioned issue of stations not providing files for specific years, it is equally critical to highlight that the available files themselves may contain missing values. This is caused by the inherent complexities of NDBC data, stemming from real-world origin. Data gaps within NDBC files might result from numerous causes, such as data corruption, recording failures at the stations, or technical disruptions. Recognizing and addressing these discrepancies is crucial when handling datasets of this nature.

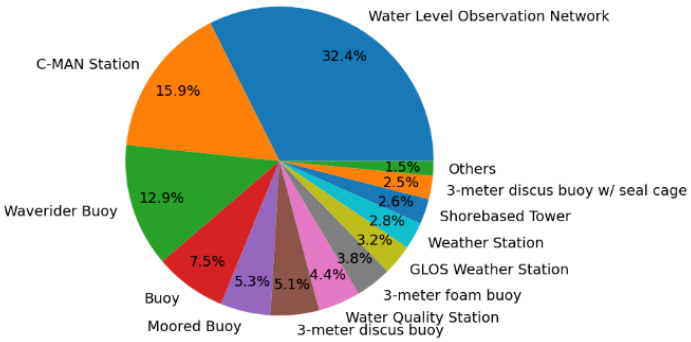
[3]https://www.ndbc.noaa.gov/metadata/stationmetadata.xml



a) Number of available files by years



b) Distribution of station operator



c) Distribution of station types

Figure 2.7: Insights from NDBC metadata

2.4.2 Fifth ECMWF Reanalysis

ERA5, generated by the European Centre for Medium-Range Weather Forecasts (ECMWF), represents the latest advancement in a series of global atmospheric reanalysis datasets. In this context, reanalysis entails the assimilation of observational data and model simulations, leveraging the principles of data assimilation rooted in physical laws, to generate a coherent and extensive dataset. (C3S CDS, 2023)

In the field of atmospheric sciences, reanalyses have gained significant traction, particularly in operational weather centers. They are used to assess advancements in modeling and assimilation techniques, evaluate the influence of changes in observing systems, and obtain up-to-date climatological data for evaluating forecast errors. ECMWF has made significant contributions in the field of reanalysis, with the ongoing evolution of ECMWF forecast models consistently benefiting the reanalysis process. Over time, successive atmospheric reanalyses have consistently demonstrated improvements, such as higher horizontal resolution and more advanced data assimilation schemes. ERA5 includes a land component, ocean surface wave and atmospheric ozone products. (Hersbach et al., 2020)

ECMWF offers a global ocean reanalysis known as ORAS5 (Ocean Reanalysis System 5). Despite the specific interest in weather data pertaining to the oceans, reliance continues on ERA5 due to ORAS5's limitation in providing only monthly temporal resolution. (C3S CDS, 2021)

ERA5 Configuration

ERA5 is built upon the Integrated Forecast System Cycle 41r2, which served as the operational medium-range forecasting system at ECMWF from 8 March to 21 November 2016. Notably, compared to the Cy31r2 framework that underlies ECMWFs previous reanalysis, ERA5 incorporates an additional decade of research and development efforts encompassing all its components. Moreover, significant improvements have been made to the data assimilation methodology, adopting the hybrid incremental Four-Dimensional Variational Data Assimilation system (4D-Var) developed by Bonavita et al. (2016). The set of four dimensions comprises three dimensions related to space and one dimension related to time. The goal of 4D-Var is to find the optimal set of initial conditions that minimizes the difference between the model's predicted state and the observed state over the assimilation window. It involves solving an optimization problem, where the cost function quantifies the mismatch between model and observations, along with additional constraints and error statistics. Those initial conditions are used to generate an ensemble of simulations through the High-Resolution Ensemble System

(HRES). The HRES assimilation system utilizes 12-hourly windows to incorporate observations from specific time ranges. Analysis windows, following the temporal evolution within each window, are saved hourly. Within each analysis window, the gathered information is propagated via a concise forecast that commences from the analysis fields 9 hours into the window. This forecast serves as the initial state for the subsequent assimilation. The HRES assimilation system combines incremental 4D-Var, based on the methodology presented by Courtier et al. (1994), for the atmosphere and ozone, with land data assimilation (LDAS). This configuration exemplifies a case of weak coupling, as discussed by Penny et al. (2017) This type of coupling entails the incorporation of land surface influence and other observations in subsequent analyses solely through the combined short forecast derived from the resulting sub-analyses. The incremental formulation of the HRES assimilation system employs stepwise minimization of a linearized quadratic 4D-Var cost function at reduced resolution in inner loops. The outer loops incorporate nonlinear updates at full resolution and involve the integration of the coupled model over the duration of the assimilation window. In ERA5 HRES, three inner loops are utilized, while its ensemble component employs two inner loops. In the HRES assimilation system, the ensemble component relies on ECMWF's Ensemble of Data Assimilation (EDA) system to estimate uncertainty in analysis and short-range forecasts. A detailed description of EDA can be found in Isaksen et al. (2010). The ocean wave analysis, carried out through optimal interpolation (OI), is integrated into the final model evolution of 4D-Var. To align with ERA5's hourly output, the OI procedure has transitioned from a 6-hourly frequency, as observed in ERA-Interim, to an hourly interval. Data pertaining to sea surface temperature (SST) and sea ice concentration (SIC) is obtained from external sources which offer comprehensive gridded datasets. The integration process involves a slight interpolation step, where the data is regridded onto the ERA5 model grid, taking into account the model's land-sea mask and cross-validation between the two variables. A comprehensive overview of the ERA5 configuration and a detailed examination of its various components can be found in Hersbach et al. (2020).

ERA5 hourly data on single levels

Within this project, the ERA5 dataset, specifically identified as "ERA5 hourly data on single levels from 1940 to present" (Hersbach et al., 2023), is utilized. It offers hourly estimates for a diverse set of atmospheric and ocean wave variables. This dataset consists of a regridded subset extracted from the full ERA5 dataset while preserving its native resolution. It is stored online on spinning disk, ensuring fast and convenient accessibility. (C3S CDS, 2023)

Table 2.7 presents a detailed overview of the important parameters related to this dataset. A frequently updated overview of all ERA5 datasets can be found in Berrisford et al. (2020).

Data Type	Gridded
Projection	Regular latitude-longitude grid
Horizontal Coverage	Global
Horizontal Resolution	0.25° x 0.25° (atmosphere), 0.5° x 0.5° (ocean waves)
Temporal Coverage	1940 to present
Temporal Resolution	Hourly
File Format	GRIB
Update Frequency	Daily

Table 2.7: Description of dataset: ERA5 hourly data on single levels (C3S CDS, 2023)

In this project, buoy station measurements and variables provided by ERA5 at corresponding locations are combined. Disregarding the precision of NDBC buoy coordinates, the resolution constraint of 0.25° allows for an exact location match only within a tolerance of 0.125°, which can be converted to approximately 13.875 km in terms of spatial difference.

2.4.3 Comparative Analysis of Dataset Features: NDBC and ERA5

NDBC standard meteorological dataset provides 14 measurements (NDBC, n.d.-b) while ERA5 provides 262 variables (C3S CDS, 2023). Five NDBC measurements lack a corresponding variable in ERA5, and therefore, they are excluded from further consideration in this study. To ensure completeness, Table 2.8 provides a description of these measurements. A detailed examination of the wind, wave, pressure, and temperature parameters, considered in this study, will be presented in this section. Moreover, Table 2.9 offers a succinct comparison between the features provided by NDBC and their ERA5 counterparts.

Wind

While wind is described with direction and speed in NDBC dataset, ERA5 provides u (east) and v (north) components. To unify the description of the wind, the ERA5 components are converted to angle and magnitude using Pythagoras' theorem and circular functions. A further variation exists in the elevation at which wind data is provided. ERA5 data encompasses altitudes

Feature	Description
GST	Peak 5 or 8 second gust speed (m/s) measured during the eight-minute or two-minute period. The 5 or 8 second period can be determined by payload.
DPD	Dominant wave period (seconds) is the period with the maximum wave energy.
VIS	Station visibility (nautical miles). Note that buoy stations are limited to reports from 0 to 1.6 nautical miles.
PTDY	Pressure Tendency is the direction (plus or minus) and the amount of pressure change (hPa) for a three-hour period ending at the time of observation. (not in Historical files)
TIDE	The water level in feet above or below Mean Lower Low Water (MLLW).

Table 2.8: Excluded NDBC features (NDBC, n.d.-b)

of 10 or 100 meters above the surface, whereas the sensor height at individual stations varies. A substantial number of stations do not disclose the sensor height in their metadata, but among those that do, it is predominantly below 10 meters, barring a few exceptional cases. Hence, the decision was made to utilize the ERA5 10-meter wind components. (C3S CDS, 2023; NDBC, n.d.-b, n.d.-d) Furthermore, it is important to note, that the ECMWF explicitly mentions, that the assimilated data can't be directly compared to actual measurements:

> "Care should be taken when comparing this parameter with observations, because wind observations vary on small space and time scales and are affected by the local terrain, vegetation and buildings that are represented only on average in the ECMWF Integrated Forecasting System (IFS)." (C3S CDS, 2023, see 10m-u-component of wind, 10m-v-component of wind)

Wave

Wave measurements reported by the NDBC are not acquired directly through sensors installed on the buoys. Instead, the buoys utilize accelerometers or inclinometers to measure the heave acceleration or vertical displacement of the buoy hull. In addition, the measurement of hull azimuth, pitch, and roll is essential for capturing directional waves, but these measurements are only reported for specific stations. On the buoy, the data undergoes transformation from the time domain to the frequency domain through the utilization of a Fast Fourier Transform (FFT) performed by the onboard

processor. Only the transformed data, but not the raw measurements is transmitted to the shore-side. The transformed data is then used to obtain features like significant wave height, average wave period and the direction from which the wave comes. (NDBC, n.d.-a)

Equivalent variables are available in ERA5, therefore no further adoption of the wave data is required. Additional insights and comprehensive information about NDBC's wave measuring system can be found in reference (Steele & Mettlach, 1993).

Pressure

Both datasets provide data of the atmospheric pressure at sea level, which refers to the air pressure exerted by the Earth's atmosphere at a specific location. NDBC reports pressure values in hectopascals (hPa), whereas ERA5 expresses them in pascals (Pa). To ensure consistency, a unit conversion is essential. (C3S CDS, 2023; NDBC, n.d.-b)

Temperature

NDBC furnishes measurements of air and sea surface temperature, along with the direct assessment of dewpoint temperature. ERA5 provides assimilated counterparts of these variables. The sea surface temperature measured by NDBC directly accounts for diurnal variations resulting from the daily cycle of the sun, whereas ERA5 provides a reference sea surface temperature (SST) that has diurnal fluctuations eliminated. Air temperature sensors and the dewpoint hygrometers are positioned at the same elevation, although this elevation differs across various stations. As for ERA5, values at a height of 2 meters are utilized. Since ERA5 presents temperature values in Kelvin (K), whereas NDBC reports data in degrees Celsius (°C), all values of ERA5 require a subtraction of 273.15 to ensure matching units. (C3S CDS, 2023; NDBC, n.d.-b)

NDBC DATA			ERA5 Equivalent		
Feature Name	Unit	Description	Feature Name	Unit	Description
WSPD	m/s	Wind speed averaged over an eight-minute period for buoys and a two-minute period for land stations.	10m v-component of wind	m/s	Northward component of the 10m wind. It is the horizontal speed of air moving towards the north, at a height of ten metres above the surface of the Earth.
WDIR	degree	Direction the wind is coming from in degrees clockwise from true North during the same period used for WSPD.	10m u-component of wind	m/s	Eastward component of the 10m wind. It is the horizontal speed of air moving towards the east, at a height of ten metres above the surface of the Earth.
WVHT	m	Significant wave height is calculated as the average of the highest one-third of all wave heights during the 20-minute sampling period.	Significant height of total swell	m	Significant height of total swell represents the average height of the highest third of surface ocean/sea waves associated with swell. It represents the vertical distance between the wave crest and the wave trough.
APD	s	Average wave period of all waves during the 20-minute period.	Mean wave period	s	Mean wave period is the average time it takes for two consecutive wave crests, on the surface of the ocean/sea, to pass through a fixed point.
MWD	degree	Mean direction from which the waves at the dominant period (DPD) are coming in degrees clockwise from true North.	Mean wave direction	degree	Mean wave direction is the mean direction ocean surface waves are comming from in degrees clockwise from true North.
PRES	hPa	Sea level pressure (hPa)	Mean sea level pressure	Pa	Mean sea level pressure is the pressure (force per unit area) of the atmosphere at the surface of the Earth, adjusted to the height of mean sea level.
ATMP	°C	Air temperature (Celsius). Sensor hight depends on station.	2m temperature	K	Temperature of air at 2m above the surface which is calculated by interpolating between the lowest model level and the Earth's surface, taking account of the atmospheric conditions.
WTMP	°C	Sea surface temperature (Celsius). For buoys the depth is referenced to the hull's waterline. For fixed platforms it varies with tide, but is referenced to, or near Mean Lower Low Water (MLLW).	Sea surface temperature	K	Sea surface temperature is the temperature of sea water near the surface. In ERA5, this parameter is a foundation SST, which means there are no variations due to the daily cycle of the sun (diurnal variations).
DEWP	°C	Dewpoint temperature taken at the same height as the air temperature measurement.	2m dewpoint temperature	K	2m dewpoint temperature is the temperature to which the air, at 2 meters above the surface of the Earth, would have to be cooled for saturation to occur. It is a measure of the humidity of the air.

Table 2.9: Relevant NDBC measurements and their ERA5 counterparts (C3S CDS, 2023; NDBC, n.d.-b)

2.5 Summary

In this chapter, the background and related work relevant to research on weather prediction using deep learning techniques are explored.

The chapter starts by examining Numerical Weather Prediction (NWP) and its historical development, highlighting the continuous advancements leading to the state-of-the-art methods used today.

NWP relies on approximating partial differential equations to predict weather patterns, emphasizing the non-trivial nature of this task and the significant computational resources required. The challenges faced in weather prediction and the ongoing trends in NWP were discussed.

The concept of deep learning was explored, providing a general classification of this domain. Various deep learning architectures for time series forecasting were explored, with a focus on two promising approaches: Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). The internal functions and intuitions behind these architectures were elucidated, highlighting their capabilities for handling spatial (CNN) or temporal (RNN) data.

Moving on to deep learning weather prediction, the general usage in contemporary weather analysis was examined. The concept of completely data-driven hybrid architectures, which combine different deep learning approaches for improved weather prediction, was explored. Additionally, theory-guided coupling architectures were discussed, showcasing the benefits of combining theory-driven and data-driven methods.

To compare NWP and Deep Learning Weather Prediction (DLWP), their advantages and limitations were assessed. Factors such as computational resources, accuracy, resolution, and the ability to forecast short or long term, as well as small-scale or global weather patterns, were considered. Especially, it was pointed out that DLWP excels in local short-term forecasting, while NWP retains its superiority in global long-term forecasting.

Subsequently, the data sources used in the research were examined. The National Data Buoy Center (NDBC) and its Historical Standard Meteorological Data were discussed, explaining how this valuable dataset was collected. Additionally, the Fifth ECMWF Reanalysis dataset was discussed, sharing a similar intuition with NWP in its creation process. The reasoning behind the consideration of ERA5 hourly data on single levels was also provided.

Finally, a comparative analysis of the dataset features obtained from both NDBC and ERA5 was conducted. Differences in individual features were emphasized, and the necessary adjustments to create a consistent dataset that incorporates the attributes of both sources were highlighted.

In conclusion, this chapter provided a comprehensive overview of the background and related work rof this research project on enhancing ocean buoy predictions using with physics-informed regularization through ocean models. The exploration encompassed the historical evolution and contemporary trends in Numerical Weather Prediction, a deep dive into the fundamentals of deep learning, and a comprehensive investigation into its application in weather prediction. Moreover, a comparison was drawn between the advantages and limitations of NWP and DLWP. Additionally, the data sources utilized in this project were discussed.

3 Requirements and Concept

This pivotal chapter serves as the bridge between the reviewed literature and the practical application of the research objectives. First, the motivation of the research is presented, followed by the considered approach for integrating domain-specific knowledge into the forecasting process. Subsequently, the specified requirements are listed and the resulting conceptual architecture is described by outlining abstract design decisions for each component.

3.1 Motivation

The objective of this thesis is to examine the influence of incorporating physics-guided regularization using modeled data in the realm of ocean buoy prediction. Contributing to improvement in this field is motivated by the fact that it influences a wide range of areas. First of all, it is relevant for predicting extreme weather events, such as hurricanes, tsunamis, and storm surges. Furthermore, it is relevant for energy planning, particularly as the proportion of renewable energy sources such as solar, wind, wave and tidal resources continues to grow (Widén et al., 2015). Additionally, improvements in weather prediction enable advancements in ship weather routing, which involves optimizing ship routes and speeds based on anticipated weather conditions (Zis et al., 2020). Given the economies of scale, even slight percentage adjustments can lead to significant changes in fuel consumption, operational costs, and emissions of pollutants. Should improvements in ocean buoy weather prediction facilitate advancements in weather routing, resulting in a modest 1% reduction in fuel consumption, the consequential worldwide decrease would amount to around 7 million metric tons of CO2 emissions per year (Crippa et al., 2022).

3.2 Problem Statement

As mentioned in subsection 2.1.2, numerical weather prediction remains the predominant method for practical weather forecasting. However, as discussed in subsection 2.3.3, deep learning has shown promising results, particularly in localized short-term prediction, despite being in its early

stages. Given the track record of deep learning displacing existing technological approaches in several domains, attention is focused on this promising technology. However, the advantages of the theory-driven numerical weather prediction will not be ignored.

As discussed in subsection 2.3.2, the approach of physical informed neural networks is to incorporate physical information as regularization in neural network training involves integrating complex physical equations. However, this method has drawbacks, including the need for specialized domain knowledge and the requirement for individualized solutions for each feature.

The idea behind this project is derived from the fact that data models like the European reanalysis are based on physical laws. This research project aims to leverage this theory-driven dataset to enhance ocean buoy observation forecasting. In other words, the research's objective is to explore a neural network that is indirectly influenced by physical principles. In order to investigate this idea, the subsequent requirements were established.

3.3 Requirements

In the requirements engineering phase, the distinctions outlined by Sommerville (2011) were utilized to categorize the system's requirements into functional and non-functional types. Functional requirements depict the services the system is expected to deliver, specifying its core tasks. Conversely, non-functional requirements define properties and limitations on these tasks, generally relating to the system as a whole rather than isolated features.

3.3.1 Functional Requirements

Data Representation

In order to investigate physics-regulation, a dataset consisting of real-world data and theory driven data is necessary. An appropriate data representation must be formulated, incorporating data from both source types. This representation needs to aptly capture the intricate relationships within oceanic weather. The created dataset must provide comprehensive, pertinent data records available in large numbers. Furthermore, features from both sources must be coincident in both chronological and geographical aspects.

Preprocessing

The system shall include proper preprocessing. Given the inherent nature of real-world observations, it must effectively address the issue of missing values. The dataset shall also be structured for time series forecasting which includes transforming it to a stationary supervised problem divided into test and train sets.

Tests

The system must be capable to run forecasting trials. The result of the tests must represent the performance of the particular network used in a way that allows comparison and conclusions to other tests.

Models

The machine learning models considered for testing shall be constructed in a manner that allows for the modulation of theory-driven impact. The system must be able to run several experiments testing the same model but with different levels of physics-informed regularization.

3.3.2 Non-functional Requirements

Reusability

Given the Gulf States Center of Environmental Informatics' strong interest in advancing research in this field, it is important to ensure the development of a reusable implementation. Consequently, the dataset generation process should possess the ability not only to construct a singular, particular dataset but also to exhibit the necessary adaptability for generating a diverse array of datasets by employing various combinations of NDBC buoys and timeframes. Moreover, the testing environment should have the capacity to conduct experiments beyond the scope of this project. Consequently, the provided implementation must be inherently extensible and easily adaptable to a wide range of application scenarios.

Neural Network Architectures

The system itself must not be limited to certain neural network architectures. However, it is crucial to identify the most promising architectures for the study of this thesis. It must be pointed out that this project does not primarily seek to identify the most optimal architecture or hyperparameters, but

rather aims to emphasize the impact of physics-informed regulation on each individual network.

Performance

Given the enormous available data in both data sources, the downloading process must be intelligently organized to minimize execution time. Furthermore, the system must be capable to handle large amount of data within a feasible execution time.

3.4 Conceptual Architecture

In this section, the conceptual architecture derived from the requirements is presented. Descriptions of the conceptual design decisions for the components depicted in Figure 3.1 are provided here, while further explanations can be found in chapter 4.

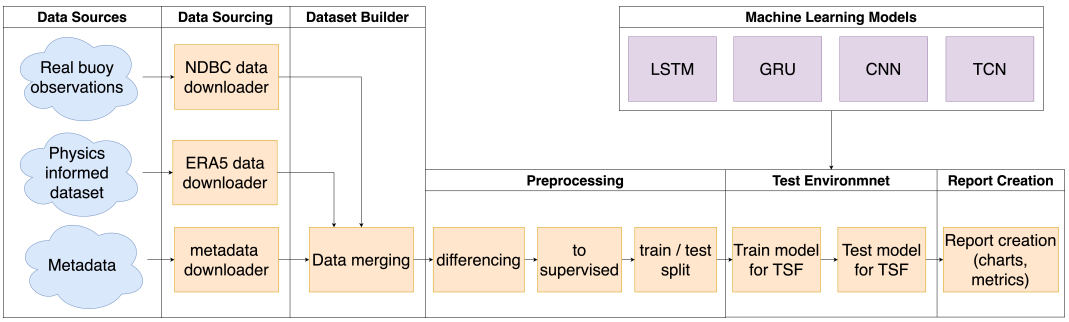


Figure 3.1: Conceptual architecture of the toolkit for dataset creation and test execution divided into 7 components.

Data Sources

The description of the architecture begins by specifying the considered data sources. The basis of this project rests upon real-buoy observations. Therefore, the historical standard meteorological data from NDBC is considered. A thorough description of this dataset can be found in subsubsection 2.4.1. To regulate the prediction of those observations, a physics informed dataset is needed. Therefore, the ERA5 hourly data on single levels from 1940 to present as described in subsubsection 2.4.2, is incorporated. For the sake of readability, these two specific datasets will henceforth be referred to just as NDBC and ERA5 data. As a third data source, NDBC provides a station page for each available station including relevant metadata.

Data Sourcing

For both datasets a procedure to download the relevant data must be implemented. These procedures prevent unnecessary downloads and ensure that once data is fetched, it can be directly read from the disk. Due to the absence of station location information in the NDBC dataset, it becomes necessary to acquire longitude and latitude data for ocean buoys through scraping the station pages. This is essential to establish geographical alignment. Within this step, it is very little effort to scrape additional metadata. This metadata can be incredibly valuable for station selection and can serve as a valuable foundation for analyses in related research endeavors.

Dataset Builder

The next component in the systems pipeline is the dataset builder. It is responsible for merging NDBC and ERA5 data while fulfilling the mentioned requirements. ERA5 exclusively furnishes data at full-hour timestamps. The sole viable approach to meet the requirement of chronological alignment is to selectively filter NDBC data and exclude all records obtained at timestamps other than full hours. In terms of geographical alignment, ERA5 values corresponding to the coordinates of NDBC buoys have to be employed. Oceans are complex interconnected systems. It is hypothesized, that representing a whole area instead of a single location in one data entry could help the neural network to detect patterns which leads to an increased forecast accuracy. However, adopting this approach poses challenges in terms of accommodating a substantial volume of records and complicates the management of missing values. Therefore, the decision to develop two data representations and compare them was made. The first one should represent several NDBC stations and their ERA5 counterparts within one instance. Since removing instances within this approach could lead to relevant data gaps, the data imputation technique suiting the given data best must be identified. The second approach represents only data gathered from one station within one record. Merging data records from several stations within one dataset allows removing all instances containing missing values but still meeting the requirement of providing a large number of data records. Both data representation techniques are detailed in section 4.1.

Preprocessing

Preprocessing plays a pivotal role in preparing the raw data for effective time series forecasting. The unprocessed time series exhibits non-stationarity, necessitating the differencing preprocessing step, where consecutive observations are subtracted to eliminate trend and seasonality. Furthermore, it

is essential to frame the data in a manner that allows machine learning models to learn from past observations to predict future values. Therefore, the subsequent preprocessing step transforms the data into a supervised problem, where input data is paired with corresponding target outputs. Subsequently, the prepared dataset is split into training and testing subsets. This division ensures that the machine learning model is trained on one set of data and its performance is subsequently evaluated on a separate set, which it hasn't previously encountered.

Machine Learning Models

As highlighted in the literature review, recurrent neural networks (RNNs) (Subsection 2.2.1) and convolutional neural networks (CNNs) (Subsection 2.2.2) emerge as the most promising architectures for time series forecasting. Due to the issue of vanishing and exploding gradients as described, no basic RNN architecture will be employed. Instead the focus will be on the two enhanced versions LSTM and GRU. In the context of CNNs, the consideration extends beyond the fundamental version to encompass TCN, a CNN architecture specifically designed for sequential data.

Within a neural network's loss function, the measure of the distance between the predicted value and the actual ground truth is evaluated. Through the process of training, the network iteratively adjusts its internal parameters, including weights and biases, with the objective of minimizing this divergence. By minimizing the loss, the neural network gains an understanding of the underlying patterns within the data and enhances its ability to make accurate predictions on new, unseen inputs. In order to maintain control over the impact of the two considered data sourced, a specialized loss function is devised. This function integrates a parameter that governs the effect of features from each data source on the adjustment of weights and biases.

In conclusion, four distinct neural networks are constructed, each of which is equipped with a custom loss function.

Similar to the case of physical informed neural networks, the training process is impacted by the presence of physical laws, thereby exerting an influence on the overall training dynamics. However, a key characteristic that defines PINNs is the incorporation of partial differential equations (PDEs) within the neural network architecture, as previously emphasized in scientific literature. (Cuomo et al., 2022; Raissi, Perdikaris, & Karniadakis, 2019) Hence, while the described neural network itself may not be inherently physics-informed, the data employed to govern the weight adjustment process exhibits physics-informed characteristics. Therefore, this approach

is termed as physics-informed regularization through modeled data.

Test Environment

The test environment serves as the operational core of this system. Building upon the preprocessed data crafted by the dataset builder, it collaborates with the chosen neural network model component to ensure a thorough training and testing process. The outcome of this component encompasses the predicted values as well as the factual observed values from the test set. It is capable to run experiments for exploring the change of different levels of physical informed regulation. However, it also can be used to test any neural network potentially added as a machine learning module and works with any dataset created by the dataset builder.

Report Creation

To simplify the evaluation of the experiments results, the raw output of the test environment serves as input for the report creation component. At this point, a comprehensive report is created, presenting an array of charts showcasing the predictions of features and metrics pertaining to the error rate.

3.5 Summary

This chapter forms an intersection of the literature review and the practical part of the thesis. It begins by underlining the motivation that steers this research – a pursuit to comprehend the effects of merging physics-guided regularization with modeled data for ocean buoy forecasting.

The importance of this research area lies in its widespread implications, ranging from predicting extreme weather conditions to its role in energy planning and its potential in optimizing ship routing.

In delving into the problem statement, while numerical weather predictions currently dominate the landscape, the emergence of deep learning offers fresh perspectives, especially for short-term local predictions. This research leans towards the potential of deep learning while still recognizing the benefits of conventional numerical weather forecasting. By integrating the physical principles embedded in datasets such as the European reanalysis with neural network capabilities, this approach does not directly infuse the network with physical laws. Instead, it steers the neural network using physics informed data.

The requirements of the system have been delineated and categorized into functional and non-functional types. On the functional front, the need for a rich dataset incorporating real-world and theory-driven data, ensuring proper preprocessing and a comprehensive test environment for modifiable machine learning models are emphasized. Non-functional requirements touch upon facets like reusability, extensibility, and system performance.

The conceptual architecture, built upon those requirements, lays down a structured roadmap for the system implementation. Beginning with the data sources that emphasize NDBC real-buoy observations, associated metadata, and the ERA5 data model, the specification moves to the data sourcing segment. It is described that data from both sources must be fetched, ensuring that redundancy is avoided and geographical alignment is maintained. The architecture further encompasses the dataset builder, which combines NDBC and ERA5 data, followed by the preprocessing segment which primes the data for machine learning application. The machine learning models component is pivotal, emphasizing recurrent and convolutional neural networks, while underscoring the importance of custom loss functions. This is the point at which physics-informed regularization through modeled data becomes operational. The test environment employs the preprocessed data and the chosen neural model to produce experimental results. Lastly, the report creation component processes the outcomes, delivering visualizations and metrics about error rates.

In essence, this chapter offers a comprehensive overview of the approach, system requirements, and conceptual design that underpin the research's quest to refine ocean buoy forecasting through an fusion of physics-informed data and deep learning techniques.

4 Design and Development

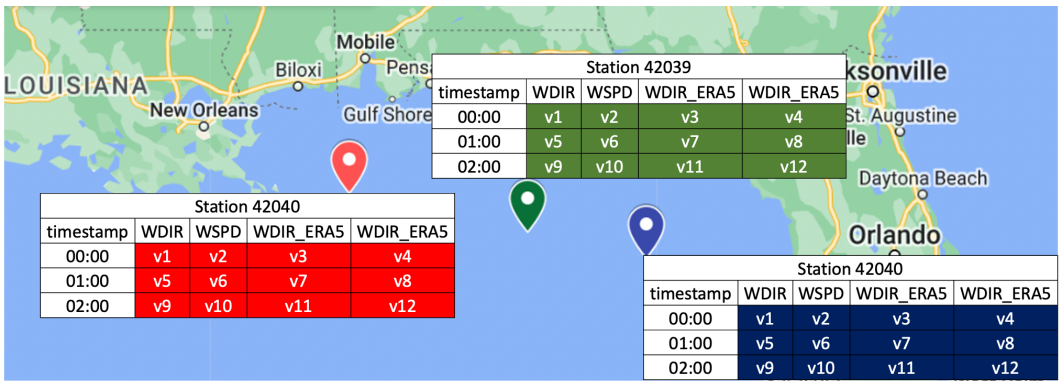
This chapter extends from the conceptual architecture, aiming to provide a comprehensive understanding of the final design decisions and the conclusive implementation. Initially, it introduces the definitive design decisions concerning data representations. Within this context, a thorough examination of diverse data imputation methods is also incorporated. Subsequently, the specific neural network models employed in the study of this project are outlined, encompassing a detailed exposition of the formulated loss function. Expanding on this groundwork, this chapter proceeds to describe the final architecture of the implemented toolkit. Consequently, it concludes with an exhaustive description of the implementation in the final section.

4.1 Data Representation

To capture the spatial-temporal data from both data sources within a unified dataset, two distinct methodologies are examined. Both are expected to be conducive to the application of physics-informed regulation within time series forecasting. Figure 4.1 illustrates the two data representation methods described in subsequent subsections.

4.1.1 Multi-Location Modelling

The first approach is called Multi-Location Modelling (MLM) since the focus lies on observing a specific area instead of a single point. In this approach, each instance encapsulates multiple features spanning different locations at a particular timestamp. The oceans are complex and elaborate systems with numerous interconnections. Therefore, it is hypothesized that this approach will empower the neural network to recognize intricate relationships between buoys relying on ocean currents or any other multi-location interconnection. The MLM approach ensures the availability of values for every timestamp and every feature in the created dataset. To address the problem of missed values in NDBC data, features undergo an initial filtering based on a threshold that specifies the permissible rate of missing values. Remaining missed values need to be imputed. Numerous



Station 42039

timestamp	WDIR	WSPD	WDIR_ERA5	WDIR_ERA5
00:00	v1	v2	v3	v4
01:00	v5	v6	v7	v8
02:00	v9	v10	v11	v12

Station 42040

timestamp	WDIR	WSPD	WDIR_ERA5	WDIR_ERA5
00:00	v1	v2	v3	v4
01:00	v5	v6	v7	v8
02:00	v9	v10	v11	v12

Station 42040

timestamp	WDIR	WSPD	WDIR_ERA5	WDIR_ERA5
00:00	v1	v2	v3	v4
01:00	v5	v6	v7	v8
02:00	v9	v10	v11	v12

Multi Location Model

timestamp	WDIR	WSPD	WDIR_ERA5	WDIR_ERA5	WDIR	WSPD	WDIR_ERA5	WDIR_ERA5	WDIR	WSPD	WDIR_ERA5	WDIR_ERA5
00:00	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
01:00	v5	v6	v7	v8	v5	v6	v7	v8	v5	v6	v7	v8
02:00	v9	v10	v11	v12	v9	v10	v11	v12	v9	v10	v11	v12

Single Station Unified Model

timestamp	WDIR	WSPD	WDIR_ERA5	WDIR_ERA5
00:00	v1	v2	v3	v4
01:00	v5	v6	v7	v8
02:00	v9	v10	v11	v12
00:00	v1	v2	v3	v4
01:00	v5	v6	v7	v8
02:00	v9	v10	v11	v12
00:00	v1	v2	v3	v4
01:00	v5	v6	v7	v8
02:00	v9	v10	v11	v12

Figure 4.1: Concept comparison of MLM and SSUM

techniques exist for this purpose and the choice of method heavily relies on the characteristics of the specific data. In preparation for this study, it was opted to evaluate the performance of three straightforward yet effective methods, along with two machine learning approaches:

Forward and Backward Fill: Missing values are imputed by replacing them with the most recent known value (forward fill) or the next known value (backward fill) in the dataset.

Mean and Median Imputation: Missing data is imputed by replacing it with either the mean or median value calculated from all known values associated with a specific feature.

K-NN Imputation: k-Nearest Neighbors is a well-known clustering algorithm. The parameter k defines the number of clusters. All data records are clustered into k groups where the Euclidian distance is used as distance

metric. The algorithm minimizes the distance to records of the same cluster and maximizes the distance to records of other clusters. To use this algorithm for data imputation, the missed values are replaced with the average of the same feature of all other records in the same cluster.

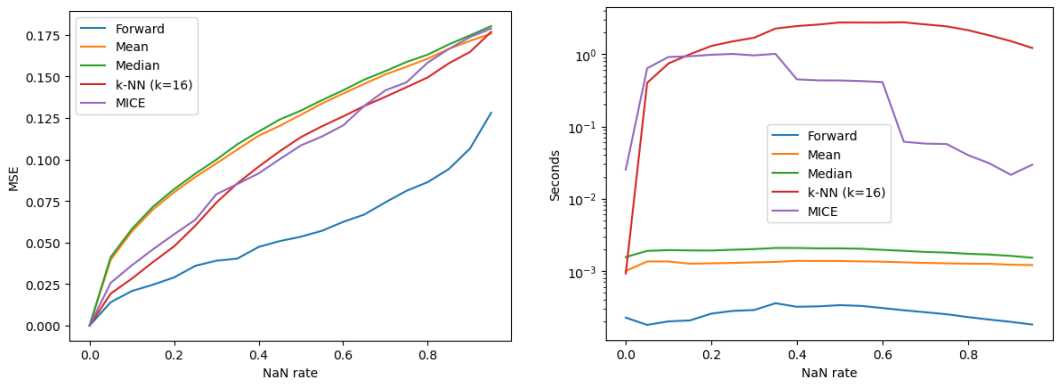
MICE Imputation: The Multiple Imputations by Chained Equations (MICE) method involves the imputation of missing values in a dataset through multiple iterations. It is an iterative algorithm that utilizes chained equations, where each variable is imputed using its own imputation model while considering the other variables as estimators. (Little & Rubin, 2019)

To assess the performance of the imputation techniques on the NDBC data, a dataset without any missing values was necessary to serve as ground truth for evaluation purposes. To select a suitable dataset, the annual data files of all stations in the Gulf of Mexico spanning from 2010 to 2022 were compared. The dataset with the lowest number of missing values was identified, which happened to be the buoy 42001 dataset from the year 2016, with only 0.23% of the values missing. To ensure data consistency, the instances corresponding to these missing values were removed by dropping them from the dataset. During each test, a specific rate of values within the selected dataset is randomly chosen and removed. The missing values are subsequently imputed, and those values are then compared to the ground truth dataset. To evaluate the performance of the imputation techniques, the mean square error (MSE) is utilized as the evaluation metric. Tests were carried out across all NaN rates, varying from 0% to 100%, in increments of 5%. The performance of all five mentioned imputation techniques was evaluated, including variations in the value of k from 2 to 256 for the k-Nearest Neighbors (k-NN) algorithm. The result, as illustrated in Figure 4.2, demonstrate that forward filling outperforms all other techniques in terms of both accuracy and execution time.

The implementation of a script to automatically create a dataset following the MLM approach can be found in subsubsection 4.4.2. The parameters chosen to create the dataset considered in the study can be found in subsubsection 5.1.

4.1.2 Station-Specific Unified Modelling

In the secondary methodology, called Station Specific Unified Modelling, emphasis is on producing a substantial number of records without the need of data imputation. Each instance encompasses measurement values and ERA5 counterparts exclusively from a specific location at a given timestamp.



a) Accuracy

b) Computation time

Figure 4.2: Comparison of different data imputation techniques on NDBC data

This approach offers a notable advantage by effectively removing instances with missing values, while simultaneously providing a larger quantity of instances. Additionally, it is well-suited for real-world applications as it enables the combination of training data from various locations and allows for operational utilization at any station.

Subsection 4.4.2 elaborates on the script implemented for automatic dataset creation under the SSUM paradigm and the choice of parameters for the dataset discussed in the sutdy is outlined in subsubsection 5.1.

4.2 Neural Network Design

Once created, the data representation that have been formulated is utilized as input for the neural networks. As mentioned in the requirements, the main objective of this thesis does not revolve around discovering the ideal hyperparameters. Instead, the aim is to explore how accuracy varies with different levels of physics-informed regulation. Therefore, the networks are intentionally designed to possess a more straightforward structure. A specialized network has been designed for each of the architecture types under consideration, as outlined in subsubsection 3.4. The layers and parameters chosen for the those neural networks are initially detailed, followed by an explanation of the custom loss function used in all of them.

4.2.1 Neural Network Architecture

Ahmed et al. (2022) offers a comprehensive study of deep sequential models, including sequential LSTM and TCN modelsin the field of weather pre-

diction. The distinction of a deep sequential model, as illuminated in this paper, is its ability to uncover hidden temporal patterns and recall data from preceding time points, unlike conventional statistical models. Additionally, the study notes that Python and Keras are the primary tools used for these models. Based on those findings, all model architectures are based on the Keras sequential model. In this framework, layers are sequentially added, ensuring that the output of one becomes the input for the following layer.

For each model, the initial layer is designed to accept an input shape defined by both the sequence length of individual data records and the count of features. Based on the chosen dataset, both parameters are dynamically assigned.

The output layer in every model is designed to yield predictions consistent with the target data's expected dimensions. This is realized using a Keras Dense layer (fully connected layer) with unit numbers corresponding to the desired output shape.

Each of the two RNNs incorporates four RNN layers with diminishing unit counts, a decision driven by the desire to maintain a deep network while still limiting its complexity. As it is common practice in RNN design, dropout layers are added after each RNN layer. Taking a cue from the Keras documentation (Keras, 2023), which exemplifies a 0.2 dropout rate, this value is applied. Consequently, each dropout layer zeroes out 20% of the input units, aiding in the prevention of overfitting. As illustrated in Table 4.1, the architecture of LSTM and GRU is the same, with the sole distinction lying in the specific LSTM and GRU layers utilized.

Layer Type	**LSTM**	**GRU**
Input LSTM Layer	LSTM (Units: 128)	GRU (Units: 128)
1st Dropout Layer	Dropout of 0.2	Dropout of 0.2
2nd LSTM Layer	LSTM (Units: 64)	GRU (Units: 64)
2nd Dropout Layer	Dropout of 0.2	Dropout of 0.2
3rd LSTM Layer	LSTM (Units: 32)	GRU (Units: 32)
3rd Dropout Layer	Dropout of 0.2	Dropout of 0.2
4th LSTM Layer	LSTM (Units: 16)	GRU (Units: 16)
Output Layer	Dense (Units: number of Features)	Dense (Units: number of Features)

Table 4.1: Layer-wise comparison between LSTM and GRU architectures

The implemented CNN model draws inspiration from the LeNet-5 architecture as outlined in subsection 2.2.2. As illustrated in Table 4.2 it consists of one-dimensional convolutional layers followed by a MaxPooling layer for down sampling and reducing dimensionality. Additionally, it incorporates a

Flatten layer to transform the output into a one-dimensional vector.

Layer	**CNN**
Input Conv1D Layer	Conv1D (Filters: 128)
1st MaxPooling Layer	MaxPooling1D (Pool Size: 1)
2nd Conv1D Layer	Conv1D (Filters: 64)
2nd MaxPooling Layer	MaxPooling1D (Pool Size: 1)
1st Flatten Layer	Flatten
1st Dense Layer	Dense (Units: 50)
Output Layer	Dense (Units: number of Features)

Table 4.2: CNN Architecture

The TCN was realized using the Keras-TCN package. The principles of this package are rooted in the paper by Bai et al. (2018), also referenced in subsubsection 2.2.2. As depicted in Table 4.3, three TCN layers are employed to strike a balance between ensuring a deep network structure and managing its complexity. Remy (2020) serves as the primary guide for selecting parameters. It is advised to employ a list of multiple of two for the `Dilations` parameter, which determines the depth of the TCN layer. To maintain a simpler network structure, `[1,2,4,8]` is chosen over the provided example of `[1,2,4,8,16,32]`.

Layer	**TCN**
Input TCN Layer	TCN (Filters: 64, kernel size: 3, dilations: `[1,2,4,8]`)
2nd TCN Layer	TCN (Filters: 64, kernel size: 3, dilations: `[1,2,4,8]`)
3rd TCN Layer	TCN (Filters: 64, kernel size: 3, dilations: `[1,2,4,8]`)
Output Layer	Dense (Units: number of Features)

Table 4.3: TCN Architecture

The Adam optimizer, prevalent in Keras documentation examples, has been implemented across all four network designs. This decision is also supported by Kingma and Ba (2014), describing the Adam optimizer as computationally efficient and well suited for problems that are large in terms of data or parameters. Training is conducted for 100 epochs, with a batch size of 64 and a validation data split of 10%. The choices made herein align with widely accepted values. A deep dive into the optimization of the parameters enumerated in this section, was beyond the purview of this project, yet it stands as a potential topic for future exploration.

4.2.2 Custom Loss Function

In order to maintain control over the features influencing the training process, a dedicated loss function that remains uniform across all architecture types is introduced.

The compile function of a Keras Model (TensorFlow, 2023) requires the specification of a loss function, which can be any callable that follows the signature:

$$\text{loss} = \text{fn}(y_{\text{true}}, y_{\text{pred}})$$

Here, y_{true} represents the ground truth values, and y_{pred} represents the predictions generated by the model. The expected return value is a float tensor. In the custom loss function, the first step involves splitting the y_{true} and y_{pred} tensors into separate parts corresponding to NDBC and ERA5. To quantify the disparity between the predictions and the ground truth values, the mean square error (MSE) is employed.

This metric allows to assess the average squared difference between the predicted and actual values. The MSE is separately calculated for the NDBC and ERA5 data components according to Equation 4.1. Subsequently, these two results are combined with their respective weight which can be set as a parameter $\alpha \in [0, 1]$ as illustrated in Equation 4.2. The final operational loss function, as depicted in Equation 4.3, integrates the NDBC and ERA5 terms' MSEs, which are weighted by the factor α.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_{\text{true}}^{(i)} - y_{\text{pred}}^{(i)} \right)^2 \tag{4.1}$$

Where:

n	:	is the number of samples or observations.
$y_{\text{true}}^{(i)}$	:	represents the true values of the target variable for the i-th sample.
$y_{\text{pred}}^{(i)}$	:	represents the predicted values of the target variable for the i-th sample.

$$\text{loss} = \alpha \times \text{MSE}_{\text{NDBC}} + (1 - \alpha) \times \text{MSE}_{\text{ERA5}} \tag{4.2}$$

Where:

$loss$	:	is the combined weighted loss.
α	:	is the weight parameter that determines the balance between NDBC and ERA5 contributions.
MSE_{NDBC}	:	is the Mean Squared Error for the NDBC data.
MSE_{ERA5}	:	is the Mean Squared Error for the ERA5 data.

$$\begin{aligned} \text{loss} = \alpha \times \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)}_{\text{true, NDBC}} - y^{(i)}_{\text{pred, NDBC}} \right)^2 \\ + (1 - \alpha) \times \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)}_{\text{true, ERA5}} - y^{(i)}_{\text{pred, ERA5}} \right)^2 \end{aligned} \tag{4.3}$$

Setting α to 1 results in assigning full importance to the observations from NDBC, while completely disregarding the influence of the simulated ERA5 data during the loss computation. Conversely, setting α to 0 reverses this effect, prioritizing the simulated ERA5 data and disregarding the observations from NDBC in the loss calculation.

By manipulating this parameter, the impact of the simulated data on the weight adjustments performed within the neural network can be controlled.

In essence, this parameter governs the degree of influence that the physics-informed data has on the forecast generated for the ocean buoy.

4.3 Toolkit Architecture

This section provides an introduction into the implementation of the integrated platform for dataset creation and forecast experiments. It is constructed using Python (Version 3.8[1]), primarily chosen for its diverse range of data analysis libraries and OS independence.

[1] https://www.python.org/downloads/release/python-380/

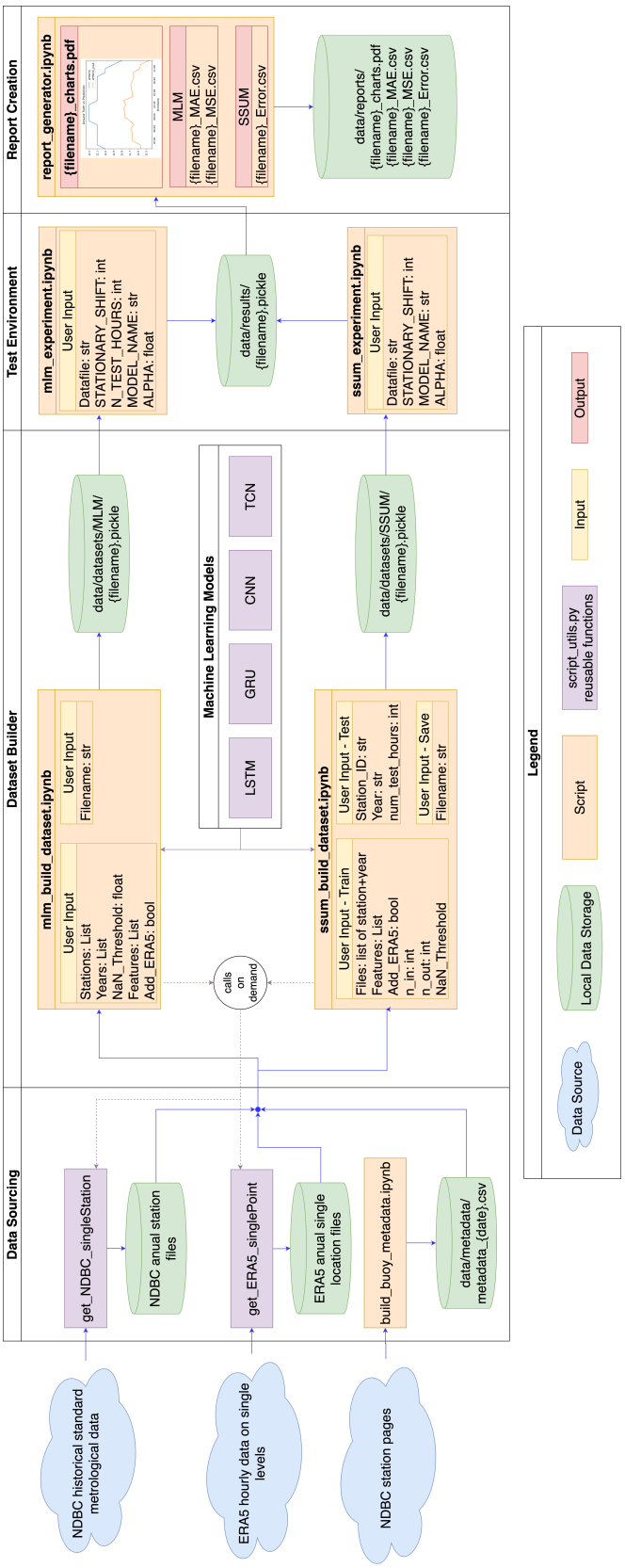


Figure 4.3: Architecture of the integrated platform for dataset creation and forecast experiments

The overarching architecture is illustrated in Figure 4.3. Six orange boxes depict Jupyter notebooks[2] which represent the core of the toolkit. Due to their interactive and collaborative characteristics, smooth execution and documentation can be unified within single files. The `script_utils.py` file, highlighted in purple, contains both commonly employed functions and more complex procedures, promoting greater code reusability. Across all files, several open-source libraries and packages were incorporated and the most relevant ones are listed in Table 4.4.

Library	**Usage**
IPyWidgets[1]	Basic user interface within Jupyter Notebook
Pandas[2]	Structured data handling
NumPy[3]	Management of arrays and handling of missing values
TensorFlow[4]	Training and testing of neural networks
Keras[5]	Definition of deep learning model architecture
Matplotlib[6]	Visualization

[1] https://ipywidgets.readthedocs.io/en/stable/
[2] https://pandas.pydata.org/
[3] https://numpy.org/
[4] https://www.tensorflow.org/
[5] https://keras.io/
[6] https://matplotlib.org/

Table 4.4: Libraries and their usages

Subsequently, an overview of the toolkit architecture is given by outlining each component while a more comprehensive explanation of the implementation details is available in the subsequent section.

Data Sourcing

The `script_utils.py` file contains two primary functions tasked with retrieving NDBC observations and modeled data from ERA5. They retrieve data for a particular station respectively a particular location and subsequently save the data for a specific year onto the disk. This ensures efficiency in terms of both disk space and network usage.

The purpose of the script `build_buoy_metadata.ipynb` is to scrape metadata from NBDC station pages. Once executed, the metadata is stored in a CSV file. Given the frequent updates to NDBC's data, it is important to

[2] https://jupyter.org/

note that if this toolkit is contemplated for future research, there might be a requirement to re-execute this script to accommodate the latest metadata.

Dataset Builder

The dataset builder component is divided into two separate scripts. Each of them implements one of the two methodologies for dataset creation as described in section 4.1. The essential parameters for constructing a new dataset can be configured through the user interface within the Jupyter notebooks. Both scripts utilize the coordinates obtained from the NDBC metadata to ensure geographical alignment between the two considered data sources. The required NDBC and ERA5 data is read from disk and the mentioned downloader functions are called on demand. As elucidated extensively in subsubsection 4.4.2, the integration of transforming the problem into a supervised format, along with the execution of train-test splitting, became imperative within the dataset building process of the SSUM approach. Once created, the dataset is saved to disk for future utilization.

Test Environment

Those files can subsequently be selected in the corresponding experiment scripts. The machine learning models as described in section 4.2 are predefined within the `script_utils.py` file. Including other models for future research endeavors requires minimal effort. Users have the option to choose one of predefined models and their corresponding Alpha values through the integrated user interface within the Jupyter notebooks of the test environment. Initially, the test scripts execute remaining preprocessing steps to ensure the dataset is structured in a stationary supervised format, partitioned into distinct train and test segments. Then, the script trains the chosen neural network using the designated dataset and then proceeds to test the network's forecasting capability. As a test outcome, the prediction as well as the ground truth values are saved to disk.

Report Creation

Finally, the report generating script reads the raw test results and creates a PDF report containing charts that depict each feature of the dataset. Furthermore, mean absolute error as well as mean square error metrics are written to disk as CSV files. These files provide a convenient means for conducting comparisons across different tests.

4.4 Development Details

The objective of this section is to provide an in-depth insight into the implementation. It describes the procedures employed within each script, outlines the expected input for each script, and describes how results are saved to the disc. By the end, the reader should grasp the full scope of the toolkit's data flow and user interactions.

4.4.1 Data Sourcing

This subsection begins by detailing the metadata file and its corresponding implementation. Furthermore, it details two procedures designed to retrieve NDBC and ERA5 data for a specific location. These procedures, implemented in `script_utils.py`, are called on demand by the two scripts dedicated to build datasets.

Metadata

The created metadata file gives an overview about which data is actually available. It provides an exhaustive inventory of the years for which a particular station supplies a data file as well as insights about the official operator and type of each station. Furthermore, the file contains the longitude and latitude coordinates of each station, which are required to obtain ERA5 data from the same location. Additional to the metadata file, a Python script named `build_buoy_metadata.ipynb` is provided. It was initially used to create the metadata file and can further be used to update it without requiring modifications, provided that NDBC maintain their current data and information delivery methods. Subsequently, attention will be directed to the methodical implementation of this script.

NDBC provides a directory[3] showcasing all the files contained in the standard meteorological historical data (STDMET). The HTML file is fetched and the body is parsed into a string. Using Regular Expressions (RegEx), a list of all file names is generated. NDBC uses the `{StationID}h{Year}.txt.gz` naming convention for those files, as exemplified by the filename `0y2w3h2012.txt.gz`. Thus, a RegEx is used again to extract the corresponding 5-character station-id and year. To enhance accessibility, the information is stored in a Pandas dataframe. The station-id serves as the index, and the years are represented by the columns. Each cell of the dataframe contains a binary value that indicates the presence or absence of the corresponding file. Given the interest in the ERA5 data at the same location, only stations with

[3] https://www.ndbc.noaa.gov/data/historical/stdmet/

known coordinates can be considered. NDBC provides those coordinates for each station on its dedicated station page. By substituting StationID with the specific ID of a station in a given URL pattern[4], one can access this page. In addition to the station coordinates the operator of the station as well as the station type are considered. Figure 4.4 displays a screenshot of an example station page, highlighting the specific data of interest for scraping.

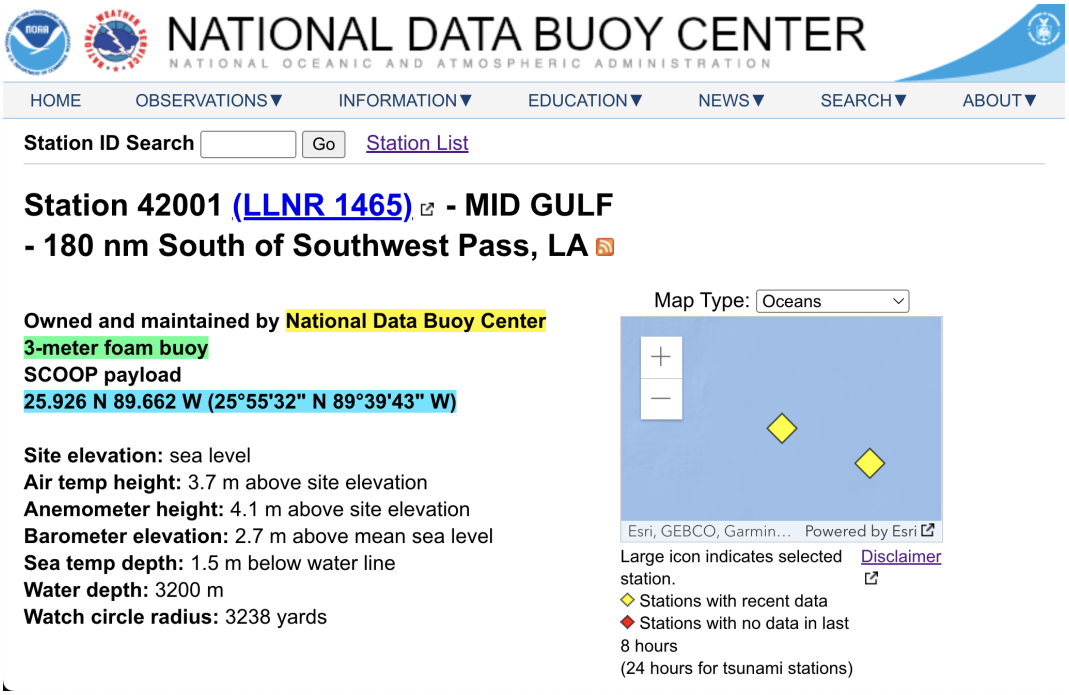


Figure 4.4: Screenshot of station page (yellow: station owner, green: station type, blue: coordinates)

To extract this information, two functions are created. The first fetches the html code and extracts the relevant strings and the second converts the string containing the coordinate into float values for longitude and longitude. In compliance with the WGS84 standard (Defense-Mapping-Agency, 1991) south and west coordinates are represented with negative values. Finally, the script stores the extracted metadata as a CSV file on the disk.

NDBC Single Location

When building a dataset, several STDMET datafiles are considered. Based on the five-digit station identifier and the corresponding year, a download

[4]URL format: https://www.ndbc.noaa.gov/station_page.php?station={StationID}

URL[5] is generated, enabling the retrieval of the required text file which is then read into a Pandas Dataframe. Subsequently, the acquired raw data file is persistently cached on the disk, thereby minimizing network traffic and execution time. In subsequent executions, the scripts check the disk for the existence of the required file, obtaining it from the NDBC website only if it is not already stored on the disk.

Prior to continuing, a brief review of three significant details covered in subsection 2.4.1 is warranted. Firstly, rows marked with '#' at the beginning serve as comments within the dataset. Secondly, the first five entries of each record encode the observation time. Lastly, missing data is indicated by the presence of varying numbers of 9's. The process of data cleaning encompasses the systematic removal of comments and the handling of missing values, whereby they are replaced with the `NumPy-NaN` value to signify their undefined nature. Additionally, the five time-related features are consolidated into a single column called `timestamp`, which now serves as a unique identifier for each instance.

An additional challenge encountered is the discrepancy in measuring intervals among different NDBC stations. Unlike the ERA5 dataset, which provides values solely at full hour timestamps, the NDBC data exhibits substantial variation in this regard. To construct a dataset where each record aligns data from both datasets at similar times, all NDBC records that correspond to timestamps other than full hours need to be excluded. Furthermore, for full-hour timestamps not covered by the considered data file, a record containing `NumPy-NaN` as value for each feature is added. Moreover, the features that were identified in subsection 2.4.3 as lacking corresponding variables in the ERA5 dataset are removed.

At this stage, a cleaned dataset has been successfully obtained that encompasses annual data from a specific station. The dataset has been meticulously prepared to ensure that, for each observation within it, an equivalent value can be obtained from the ERA5 dataset.

ERA5 Single Point

The process of collecting ERA5 data corresponding to a specific location of an NDBC station will now be elucidated. ECMWF provides the necessary data through the Climate Data Store (CDS). Accessing this data requires the utilization of the provided CDS API, for which a free CDS account and the corresponding CDS API credentials are required.(ECMWF, 2023)

[5]URL format: https://www.ndbc.noaa.gov/data/historical/stdmet/{STATION-ID}h{YEAR}.txt.gz

Every API request necessitates the inclusion of three parameters: the dataset name, which, in the particular case, is 'reanalysis-era5-single-levels'; a dictionary encompassing sub-selection parameters and the target file name. Within this dictionary, the request is further refined using parameters like product-type and format. Additionally, it serves as a filter by specifying variables, timestamps, and the geographical area. This filtering capability allows narrowing down the request to the specific data of interest. Listing 4.1 shows the structure of a request as utilized in the implementation. While most parameters remain static, the year and coordinates vary for each individual request.

```
c.retrieve(
    'reanalysis-era5-single-levels',
    {
        'product_type': 'reanalysis',
        'variable': [
            '10m_u_component_of_wind',
            '10m_v_component_of_wind',
            '2m_dewpoint_temperature',
            '2m_temperature',
            'mean_sea_level_pressure',
            'mean_wave_direction',
            'mean_wave_period',
            'sea_surface_temperature',
            'significant_height_of_total_swell',
        ],
        'year': year, # E.g.: '2022'
        'month': [
            '01', '02', '03', '04', '05', '06',
            '07', '08', '09', '10', '11', '12',
        ],
        'day': [
            '01', '02', '03', '04', '05', '06', '07', '08',
            '09', '10', '11', '12', '13', '14', '15', '16',
            '17', '18', '19', '20', '21', '22', '23', '24',
            '25', '26', '27', '28', '29', '30', '31',
        ],
        'time': [
            '00:00', '01:00', '02:00', '03:00', '04:00',
            '05:00', '06:00', '07:00', '08:00', '09:00',
            '10:00', '11:00', '12:00', '13:00', '14:00',
            '15:00', '16:00', '17:00', '18:00', '19:00',
            '20:00', '21:00', '22:00', '23:00',
        ],
        'area': coords, # E.g.: [45.0, 85.0, 45.0, 85.0]
        'format': 'netcdf',
    },
    path
)
```

Listing 4.1: Structure of CDS API request

Although it is technically feasible to download data for multiple years and an entire area using a single request, the decision was made to download and save annual files containing data for a single location. When defining the area in the sub-selection dictionary, identical values are utilized for the north and south boundaries, as well as for the west and east boundaries, effectively restricting the selection to a single point. This approach guarantees that there is a precise correspondence between each NDBC file and its corresponding ERA5 file which results in optimal storage usage.

The CDS API delivers data in the form of netCDF, a data format specifically designed for storing and retrieving scientific multidimensional data (Rew & Davis, 1990). Prior to merging the ERA5 data with the NDBC data, a conversion process is required to transform the data into a dataframe format. This conversion process also involves making adjustments to various features to ensure their conformity with the NDBC data representation as discussed in subsection 2.4.3. Initially, a conversion of the provided u- and v-components of wind into wind direction is performed, expressed in degrees north, similar to NDBC conventions. This conversion entails using the arctangent function on the quotient of u and v. Subsequently, the resulting angle is adjusted based on the signum (positive or negative) of the u- and v-components. This adjustment ensures that the angle is measured from the right axis, consistent with NDBC standards. Next, Pythagoras' theorem is employed to compute the speed at which the wind blows in the resulting direction. Moreover, the temperature and pressure values undergo unit conversion to align with the units used by NDBC. With this adjustment, the ERA5 data format precisely corresponds to the format employed by NDBC. As a result, the merging of these two datasets becomes feasible, employing either the MLM or SSUM approach.

4.4.2 Dataset Builder

Section 4.1 introduced the Multi-Location Model and the Station-Specific Unified Model as two approaches considered for building a dataset using NDBC and ERA5 data. In the next subsections, the focus is shifted to two scripts from the developed toolkit that enable the implementation of these models for dataset construction.

Build Multi-Location Model

The script `mlm_build_dataset.ipynb` is designed to construct a dataset following the multi-location model as introduced in subsection 4.1.1. The parameters required for dataset construction are described in Table 4.5.

These parameters can be set directly in the code or defined using the GUI elements within the Jupyter notebook, as illustrated in Figure 4.5.

Parameter	Datatype	Description
STATIONS	List of strings	5-digit station identifier
YEARS	List of strings	Subset of years since 1970
NAN_THRESHOLD	Float	$0 \leq$ NAN_THRESHOLD ≤ 1
FEATURES	List of strings	Subset of: WDIR, WSPD, WVHT, APD, MWD, PRES, ATMP, WTMP, DEWP
ADD_ERA5	Boolean	Default: True
FILENAME	String	Name of the output file without file extension

Table 4.5: mlm_build_dataset.ipynb parameters

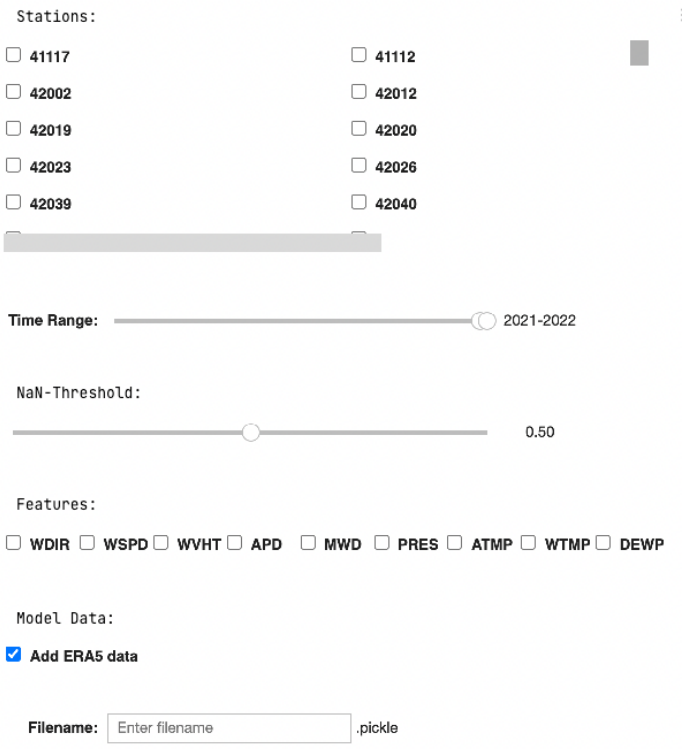


Figure 4.5: Screenshot: mlm_build_dataset.ipynb GUI

The script commences by reading NDBC files for all stations and years, adhering to the procedure outlined in subsubsection 4.4.1. Firstly, files pertaining to the same year are horizontally concatenated to form a unified dataframe. Subsequently, the data from all years is vertically concatenated, resulting in a comprehensive NDBC dataframe. The outcome is a dataframe

of NDBC data containing a singular instance for every full hour within the designated years, leaving no gaps in the temporal coverage. The column headers adhere to the naming convention of 'F_S,' where 'F' denotes the feature name and 'S' signifies the 5-digit station identifier associated with the respective station.

In accordance with subsection 2.4.3, the feature name 'F' is limited to a predetermined set of nine values: `WDIR`, `WSPD`, `WVHT`, `APD`, `MWD`, `PRES`, `ATMP`, `WTMP`, `DEWP`. To illustrate comprehension, Figure 4.6 presents a visual representation of an example header. It is important to note that the presented example, consisting of four columns, is derived from a dataset with only two features and two stations. However, it is crucial to recognize that the number of columns is significantly higher in most scenarios since it results by the number of considered stations times the available features.

Figure 4.6: Example of NDBC data header

Handling missing values: The real-world aspect of NDBC data results in missing values. This challenge is magnified when trying to align with the set criterion of only considering values measured at exact hourly intervals. Two primary approaches exist to address this issue. The first involves removing rows or columns with missing values, which improves model robustness but at the expense of losing information. Alternatively, data imputation can be employed to estimate the missing values within the dataset. The provided script effectively combines both strategies.

Once the NDBC datafiles are merged, the script calculates the missing value rate for each feature. Subsequently, features with a calculated rate surpassing the specified threshold, passed as a parameter, are removed from the dataset. Additionally, if the `features` parameter is employed, the dataframe's columns are filtered according to the provided list. As ensuring the availability of values for every timestamp is a fundamental objective of the Multi-Location Model, instances with remaining missing values cannot be simply removed from the dataset. Instead, these missing values need to be estimated or imputed in order to maintain the integrity of the dataset.

As discussed in subsection 4.1.1, forward filling is the strategie that showed the best performance on NDBC data. Given this finding, the implementation of the option to choose an imputation technique has been omitted, and forward filling has been exclusively implemented.

If the boolean parameter `ADD_ERA5` is set to false, the script omits the inclusion of ERA5 data and directly outputs the NDBC dataset. This flexibility allows users to leverage the toolkit for experiments exclusively involving observation data. However, if the parameter is set to `true`, the script follows the procedure outlined in subsubsection 4.4.1 to retrieve the required ERA5 data files.

Subsequently, these files are merged into a unified dataframe using the same methodology as previously described for the NDBC files. The columns equivalent to those removed during the feature selection process of the NDBC dataset are eliminated from the ERA5 dataset as well. The ERA5 data header follows the same naming convention as the NDBC data header, but with the addition of the postfix '_ERA5'. An example of the ERA5 data header, corresponding to the one depicted in Figure 4.6, is illustrated in Figure 4.7.

Figure 4.7: Example of ERA5 data header

With instances at precisely the same timestamps and equivalent features across the NDBC and ERA5 datasets, a column-wise concatenation of the datasets becomes viable. Extending from the initial data-header example, a final data header as illustrated in Figure 4.8 can be derived.

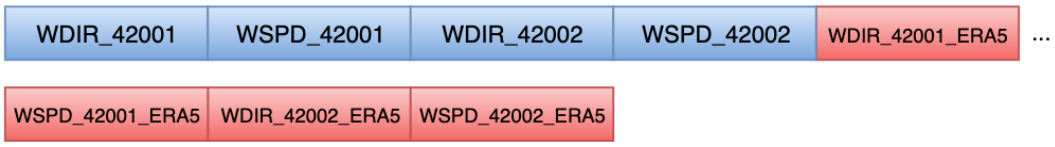


Figure 4.8: Example of final MLM data header

The generated dataset, along with the corresponding input parameters, is stored in a Python dictionary and subsequently saved to disk as a serialized `.pickle` file. This allows for convenient storage and future utilization of the dataset.

Build Station-Specific Unified Model

The script `ssum_build_dataset.ipynb` is designed for dataset construction following the Station-Specific Unified Model (SSUM) approach as introduced in subsection 4.1.2. This script incorporates additional parameters, as depicted in Table 4.6 and Figure 4.9. The inclusion of these additional

parameters stems from the nature of the SSUM approach, which allows forecasting of observations for any given station. Consequently, specifications for the test set are necessary to ensure accurate model evaluation.

Parameter	Datatype	Description
STATIONS	List of strings	5-digit station identifier
YEARS	List of strings	Subset of years since 1970
NAN_THRESHOLD	Float	$0 \leq$ NAN_THRESHOLD ≤ 1
FEATURES	List of strings	Subset of: WDIR, WSPD, WVHT, APD, MWD, PRES, ATMP, WTMP, DEWP
ADD_ERA5	Boolean	Default: True
TEST_STATION	String	5-digit station identifier
TEST_YEAR	String	
NUM_TEST_HOURS	Integer	
FILENAME	String	Name of the output file without file extension

Table 4.6: SSUM_Build_Dataset.ipynb parameters

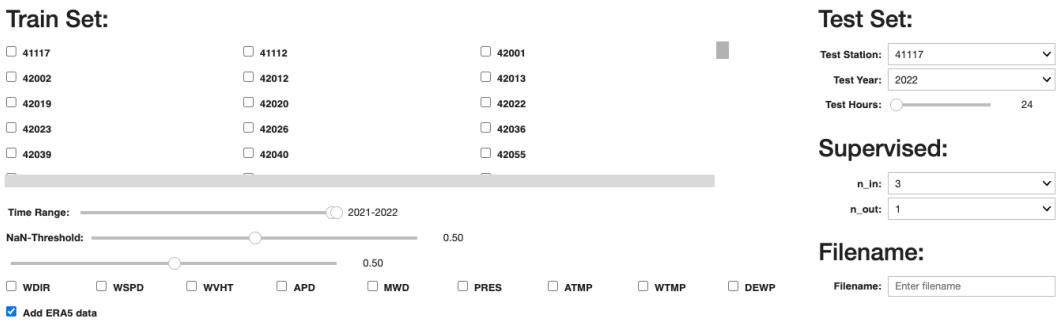


Figure 4.9: Screenshot: SSUM_Build_Dataset.ipynb GUI

The script initiates the construction of the training set, which can also be interpreted as the SSUM dataset, by iterating through the designated files determined by the STATIONS and YEARS parameters. The script then proceeds to utilize the methodology outlined in subsubsection 4.4.1 to extract a cleaned dataframe of NDBC data. Subsequently, the obtained NDBC data is filtered based on the custom feature list specified in the FEATURES parameter. Unlike the MLM script, the SSUM script handles ERA5 data separately for each location, provided that the `ADD_ERA5` parameter is set to `true`. This entails retrieving the ERA5 data for the current file using the

procedure described in subsubsection 4.4.1. The next stage involves merging the columns that correspond to the preserved features in the NDBC data with the NDBC columns, resulting in the creation of a merged dataframe. After performing this process for each file, all the resulting dataframes are concatenated vertically, forming a single merged dataframe. In the subsequent step, the script performs feature selection using the specified NaN threshold. The resulting dataframe contains a maximum of 18 features, representing each measurement and its corresponding ERA5 equivalent. The data header follows the same naming convention as MLM, excluding the station identifier. An illustrative example of the data header at this stage is depicted in Figure 4.10.

Figure 4.10: Example of SSUM data header before transforming to supervised problem

Supervised Problem: In contrast to the MLM script, which addressed the remaining NaN values through imputation, the SSUM approach aims to remove all NaN values. However, before proceeding with their removal, additional preprocessing steps are required. In order to train a neural network, the dataset needs to be transformed into a supervised problem, consisting of input and output values. In the context of time series forecasting, the neural network should learn a function that maps a sequence of past observations as input to either the subsequent observation or even a sequence of future observations. In the merged dataset, each instance corresponds to the weather conditions at a specific timestamp, which is encoded in the index of the dataframe. To transform this data into a supervised problem, a certain number of previous instances are considered as the input data, while the current instance and potentially subsequent instances are treated as the output data. During the execution of the data transformation into a supervised problem, it is crucial to ensure that instances be properly ordered and free from any gaps or interruptions. As the script proceeds to remove instances containing missing values in the subsequent step, it is essential to perform the conversion to a supervised problem within this script itself.

The parameters n_{in} and n_{out} govern the choice of instances to be employed as input and output in the supervised problem. As the data undergoes the conversion process, the number of instances decreases by $(n_{\text{in}} + n_{\text{out}}–1)$ due to the absence of valid instances preceding or succeeding the initial and final instances. Conversely, the number of features expands by the factor $(n_{\text{in}} + n_{\text{out}})$. To maintain a record of the corresponding timestamp for each feature, the relative timeshift is incorporated into the data header. Figure 4.11

depicts the data header of the continuous example after the conversion of the data into a supervised problem.

Figure 4.11: Example SSUM final data header with $n_{\text{in}} = 3$ and $n_{\text{out}} = 2$

Following this step, the script eliminates all rows in the supervised problem that include missing values, culminating in the creation of the final training set. In order to maintain comparability and avoid a reduction in the size of the test set due to missing values, the MLM approach is adopted when constructing the test dataset. However, in this case, the focus remains solely on one specific station, resulting in the coverage of only one location. Nonetheless, the implemented NaN imputation technique can still be leveraged. The parameters `TEST_STATION`, `TEST_YEAR`, and `NUM_TEST_HOURS` define the source and size of the test set, which is subsequently converted into a supervised problem. Lastly, akin to the MLM script, a dictionary is created, encompassing the utilized parameters along with the training and test sets. This dictionary is then serialized and saved as a `.pickle` file.

4.4.3 Test Environment

The primary objective of this project is to examine the influence of physics-informed regularization on the forecasting of ocean buoy data. Consequently, it is necessary to conduct tests with various levels of physics-informed regulation. To streamline this process, test scripts have been developed, which accept pertinent parameters as input, train a pre-defined neural network, perform forecasting on the test data, and save the results to a report file. To account for the disparities in the construction of MLM and SSUM datasets, the toolkit includes separate test scripts for each approach which are described in this subsection.

MLM Test Script

To assess the performance of a specific neural network architecture on a given MLM dataset, users are required to specify the parameters summarized

in Table 4.7. These parameters can be defined either by assigning specific values to the variables within the code or by utilizing the graphical user interface provided within the Jupyter-Notebook script, as depicted in Figure Figure 4.12.

Parameters	**Datatype**	**Note**
DATAFILE	String	Choose file in data/datasets/MLM
n_in	Integer	To create supervised problem
n_out	Integer	To create supervised problem
STATIONARY_SHIFT	Integer	For data-differencing
N_TEST_HOURS	Integer	Defines size of test set
Model	Integer	Choose from predefined models
Alpha	Float	$0 \leq \alpha \leq 1$

Table 4.7: mlm_experiment.ipynb parameters

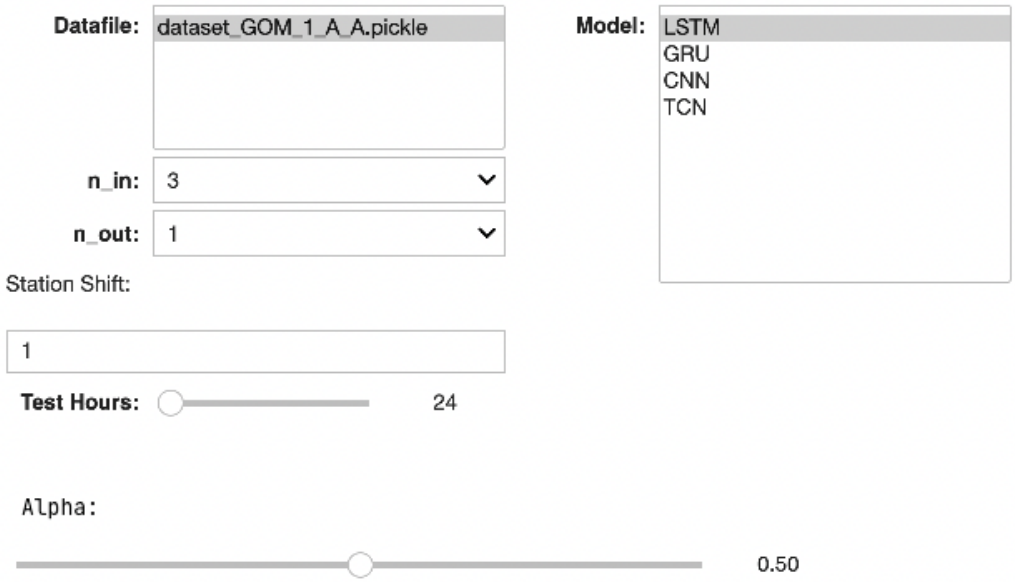


Figure 4.12: Screenshot: mlm_experiments.ipynb GUI

The script initiates by deserializing the data stored in the chosen pickle file. It then proceeds with preprocessing, starting with the conversion of the data into a stationary form. This step eliminates the influence of trends and seasonality, guaranteeing that the prediction of weather changes for future timestamps is based solely on the weather changes observed between

previous timestamps. To achieve this, the script replaces the actual values at each timestamp with the difference between the current value and the value recorded `STATION_SHIFT` timestamps earlier. Following this, the data is transformed into a supervised problem, adhering to the user-defined parameters n_{in} and n_{out}. The procedure mirrors the methodology described in paragraph 4.4.2.

Furthermore, the resulting supervised problem is divided into four distinct groups: a training and a test set, both of them further separated into input (X) and output (y) components. The number of records allocated to the test set is specified by the `N_TEST_HOURS` parameter, with the test set encompassing the most recent records in the dataset. The division between input and output components is based on the relative timestamp appended to the column header during the transformation process. Past timestamps are assigned to the input segment, while the output segment comprises the current and future timestamps.

Upon completion of the preprocessing steps, the data is now prepared for utilization by the neural network. The neural network architectures are defined within the Models class in the `script_utils.py` file, which consists of multiple methods. Each method within this class requires the input (X) and output (y) components of the training data, along with the Alpha value, to be provided as parameters. Within each method, a specific neural network architecture is defined, and a trained model is returned as the output. Within the test scripts, users can choose from a set of predefined models using the graphical user interface. Although the current predefined models align with the architectures detailed in section 4.2, the selection is designed to be extensible. It can be expanded to accommodate any neural network architecture, not limited to those with a custom loss function. This versatility ensures the script's adaptability for various potential future applications. Utilizing the trained model, the test script proceeds to generate predictions for the output corresponding to each input in the test set. The quantity of input and output timestamps is established during the creation of the supervised problem. Hence, it is crucial to note that the prediction horizon is determined by the parameter n_{out}. Before conducting a comparison between the predictions and the ground truth values, the script undertakes the task of converting the stationary data back to its absolute values. This process entails particular attention to be given to wind direction. While the stationary value and the previous observation can be directly added to yield the predicted absolute value, a modulo 360 operation is employed for wind speed to address potential overflow issues.

The major output of the script is a dataframe encompassing the ground truth value and the corresponding predictions for the complete test set.

Together with the chosen parameters and an optional textual description, the output is stored in a Python dictionary and saved to disc.

SSUM Test Script

The Jupyter notebook `ssum_build_dataset.ipynb` serves as a test environment for datasets created according to the SSUM approach. Notably, this script operates with a reduced parameter set, as indicated in Table 4.8 and illustrated in Figure 4.13.

Parameters	Datatype	Note
DATAFILE	String	Choose file in data/datasets/MLM
STATIONARY_SHIFT	Integer	For data-differencing
N_TEST_HOURS	Integer	Defines size of test set
Model	Integer	Choose from predefined models
Alpha	Float	$0 \leq \alpha \leq 1$

Table 4.8: SSUM_Experiment.ipynb parameters

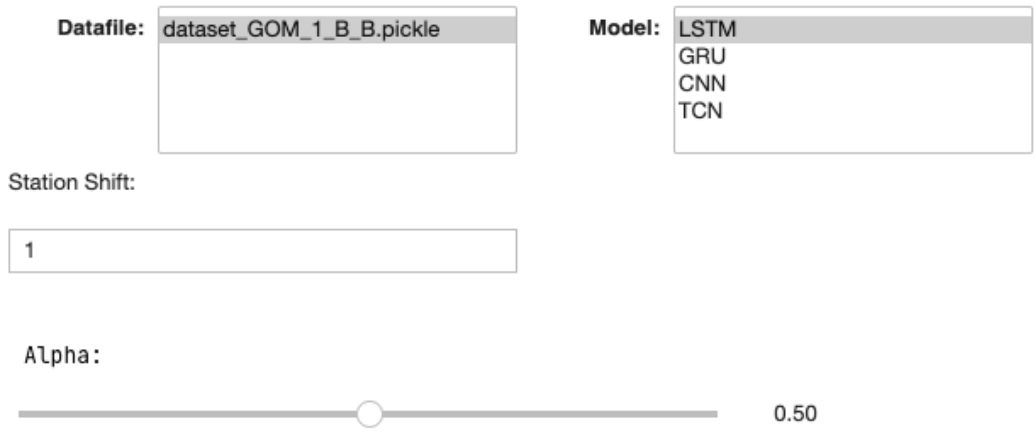


Figure 4.13: Screenshot: ssum_experiments.ipynb GUI

In contrast with the script for MLM datasets, the SSUM script utilizes data that is already organized as a supervised problem, featuring distinct train and test sets. Hence, there is no need for data transformation into a supervised problem or train and test sets; instead, the provided data is simply divided into input (X) and output (y) sections. These discrepancies

constitute the primary differences between the two scripts, while the remainder of the procedure remains unchanged, thereby ensuring an meaningful comparison of the results yielded by tests conducted using both scripts.

4.4.4 Report Creation

The decision to create reports in a separate file was motivated by the aim to enhance the flexibility and reusability of the toolkit. By doing so, the test scripts are able to store the raw results in a pickle file, which can subsequently be utilized for evaluation purposes, irrespective of the specific evaluation method employed. In the provided `report_generator.ipynb` script, users can select the desired files using the integrated selection widget, which presents a comprehensive list of all reports stored in the designated `data/reports` directory. Once the files are selected, the script proceeds to read the raw data and generates the report files mentioned in Table 4.9.

File	Filetype	Naming Convention	Created for
Report	PDF	`filename_charts.pdf`	All
MAE	CSV	`filename_MAE.csv`	MLM only
MSE	CSV	`filename_MSE.csv`	MLM only
Error	CSV	`filename_Error.csv`	SSUM only

Table 4.9: Overview of evaluation files created by report_generator.ipynb

The initial page of the generated report file provides a comprehensive overview of the metadata extracted from the read .pickle file. This includes essential information related to the dataset. Also, this page features a summary of the employed neural network architecture, offering insights into its configuration and key parameters. Subsequent pages within the report consist of charts showcasing the ground truth values of each feature plotted against the corresponding predictions over time. These visualizations enable a comparative analysis of the model's performance. Additionally, the report provides the Mean Absolute Error (MAE) and Mean Squared Error (MSE) values for each feature, offering quantitative measures of the model's accuracy and performance. Furthermore, these error metrics are saved as separate CSV files, allowing for detailed analysis. In the case of the Multi-Location Model (MLM), which has a variable number of features based on the considered stations, the error metrics are stored in individual files. Both, the MAE and the MSE file consist of a table with the measurements on the x-axis and the station ID on the y-axis. Conversely, for the Station-Specific Unified Model (SSUM), where the number of features is limited to 18, the MAE and MSE values can be represented in two columns of one table, with

the features listed on the y-axis. This approach enables both error metrics to be saved within a single file, simplifying the organization and interpretation of the results.

4.5 Summary

This chapter covered a detailed description regarding final design decisions and implementation details of the integrated toolkit for dataset creation and test execution.

First, two methodologies for integrating NDBC and ERA5 data were introduced: Multi-Location Modelling (MLM) and Station-Specific Unified Modelling (SSUM). MLM focuses on observing a specific area, encompassing features from various locations at a single timestamp. Several data imputation techniques were evaluated, and forward filling was found to be the most effective for the given data. In contrast, SSUM emphasizes generating a vast number of records without data imputation by collecting data from individual locations over time. This method handles missing values by eliminating such instances.

Second, the design of neural networks for the subsequently described study was discussed. Both the LSTM and GRU recurrent neural networks possess a comparable layout, consisting of four RNN layers paired with associated dropout layers. The LeNet-5 design informs the selected CNN architecture, and the TCN is anchored in the Keras-TCN package, a reflection of the insights from the paper by Bai et al. (2018). All these architectures utilize a custom loss function that permits the regulation of the influence of the physics-informed dataset based on a parameter called `Alpha`.

Following those specifications, the chapter delves into an overview of the systems architecture. The implemented toolkit encapsulates the full spectrum of tasks from data sourcing to report creation. It comprises one Python file with reusable functions and six main scripts, realized as Jupyter notebooks. Those scripts are designed for:

1. Retrieving updated metadata for the NDBC dataset, offering insights into data availability as well as station operator, type and location.
2. Automating dataset creation adhering to the MLM and SSUM methodologies.
3. Executing tests on varied neural network architectures and diverse Alpha values.
4. Automatically evaluating and visualizing results.

The final section of this chapter covers all development details, offering a

comprehensive understanding of data flow and user interaction.

First, the NDBC metadata collection process is described. Data about file availability, coordinates, operator, and type of each station is collected by scraping NDBC web pages. In addition to the resulting CSV file, a script for updating the metadata is also provided.

Furthermore, procedures for efficient data retrieval were described. By storing data from individual NDBC stations and their associated ERA5 counterparts locally, both network traffic and local storage usage are minimized. Within those procedures, inconsistencies between NDBC's data intervals and the regular hourly intervals of the ERA5 dataset are managed, ensuring data compatibility. Moreover, ERA5 data is harmonized with the NDBC data representation through unit conversion.

Subsequently, two scripts for dataset creation are described. These scripts access the stored data files, so the data sourcing procedures only need to be invoked as needed. Using the specified MLM or SSUM methodology, the scripts assemble a dataset determined by the input parameters available in a simple GUI in the Jupyter notebook. The inherent design of the SSUM approach mandated that certain preprocessing steps be conducted as part of the dataset creation.

Created datasets can subsequently be used for test execution in the two provided test environments. While the script developed specifically for MLM datasets involves more preprocessing operations, such steps are already undertaken during the creation of the SSUM dataset, marking the key contrast between them. In each of the two scripts, users have the capability to pick a dataset, opt for one of the pre-established machine learning models, and determine the Alpha value. Initially, the test scripts carry out the remaining preprocessing steps to ensure the dataset is formatted in a stationary supervised manner, divided into separate train and test segments. Then training and testing is executed and the raw results are saved.

Finally, users can use the implemented report generator to produce a PDF report and CSV files that display MAE and MSE outcomes.

In summary, this chapter reveals the extensive capabilities of the toolkit. While it was developed specifically for the study described in chapter 5, it furthermore promises to be a valuable resource for upcoming research.

5 Empirical Analysis of Physics-Informed Regularization

In order to showcase the utility of the newly created toolkit and examine the impact of physics-informed regularization (PIR), a study as detailed below was carried out. Theoretically, the toolkit has the capacity to construct a dataset by incorporating all accessible NDBC datafiles alongside their corresponding ERA5 counterparts. Beyond the sheer quantity of data, the selection process should prioritize data that carries substantive meaning and relevance. Consequently, a meticulous approach was taken in handpicking particular NDBC datafiles, a process explained in the initial subsection of this chapter. Moreover, within this subsection, the essential tests that need to be conducted to understand how accuracy varies with the application of varying levels of PIR are explored. The test results provide clarity on whether physics-informed regularization genuinely enhances forecast accuracy and pinpoint the most optimal PIR level. The second subsection of this chapter delineates a systematic analysis of the results, and the closing subsection is dedicated to a discussion of these discoveries.

5.1 Study Setup

Data Selection

The selection of an optimal subset of available datafiles for experiments is crucial to ensure both feasibility and efficient management of computational resources. The principles guiding the selection of particular stations is outlined subsequently.

Firstly, the analysis is limited to the geographical area of the Gulf of Mexico. Despite the availability of other regions with high-resolution NDBC stations as viable options, this choice was made based on the proximity to the GulfSCEI office in New Orleans. In pursuit of this objective, the region is limited by focusing on stations located within the latitude range of 21° N to 32° N and the longitude range of 82° W to 102° W. This delineated area, as depicted in Figure 5.1 encompasses a total of 206 stations, with the ultimate selection highlighted in red.

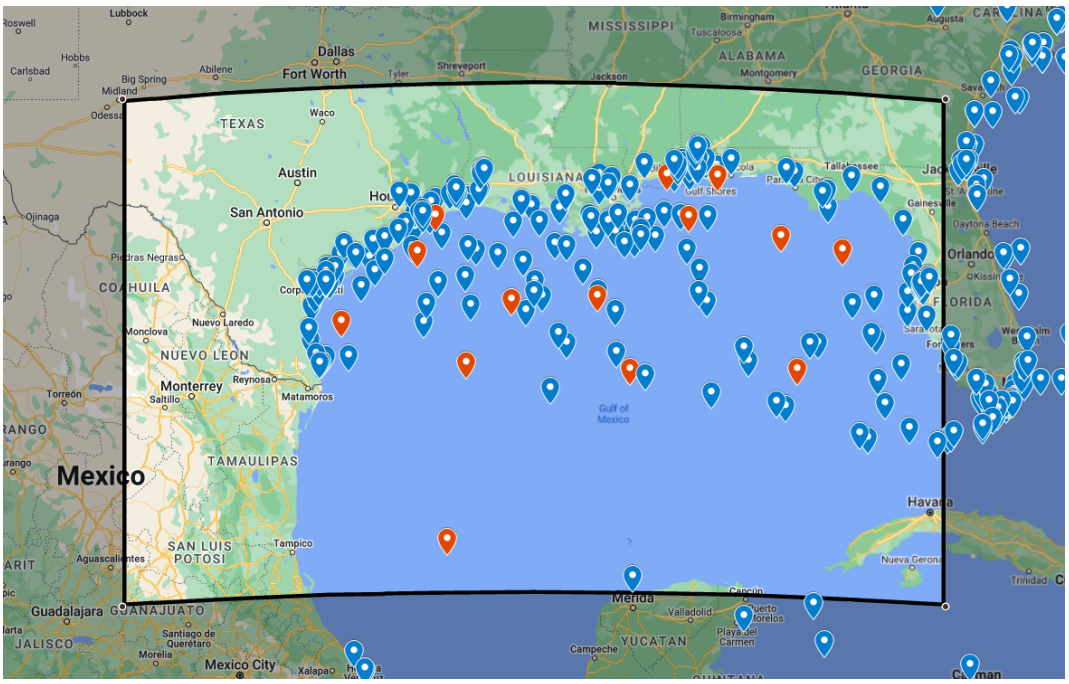


Figure 5.1: Map of considered area in Gulf of Mexico: Blue pins represent available stations, while red pins indicate the stations that have been chosen for the experiments.

Next, an analysis of the station operator organizations within this geographic area was conducted. Figure 5.2 demonstrates that the majority of stations within this area, comprising 27.2% of the total, are operated directly by NDBC. In order to maintain consistency in measurement methodologies across all stations, exlclusively those are utilized. Furthermore, as detailed in subsection 2.4.1, it is worth noting that data obtained directly from NDBC sources tends to exhibit greater accuracy compared to third-party data.

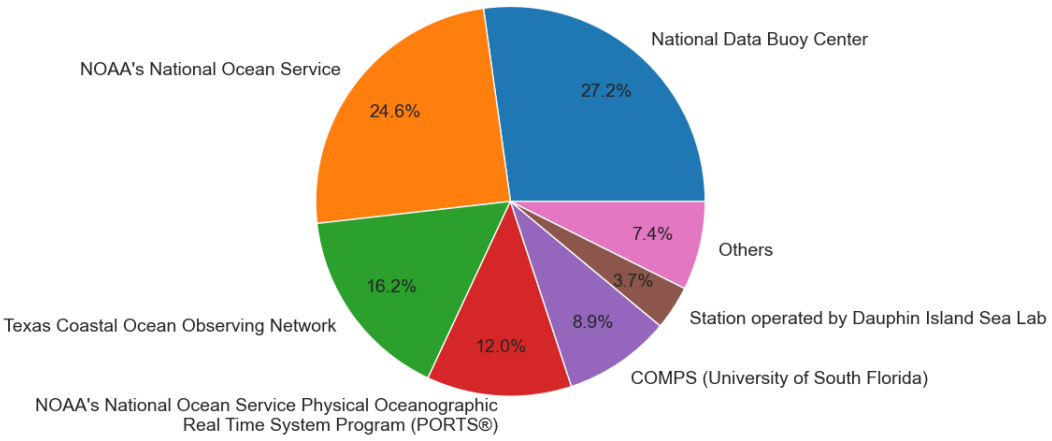


Figure 5.2: Distribution of station owners in the considered area

Following this, all stations that are not ocean buoys, such as weather stations located on shoresides or oil platforms, were excluded. Upon completion of this filtration, 28 ocean buoys operated by NDBC within the Gulf of Mexico area were identified and retained.

As depicted in Figure 5.3, a noticeable trend emerges showing a decrease in the number of stations providing data when moving further back in time. Notably, there is no year in which NDBC provides more than 13 files.

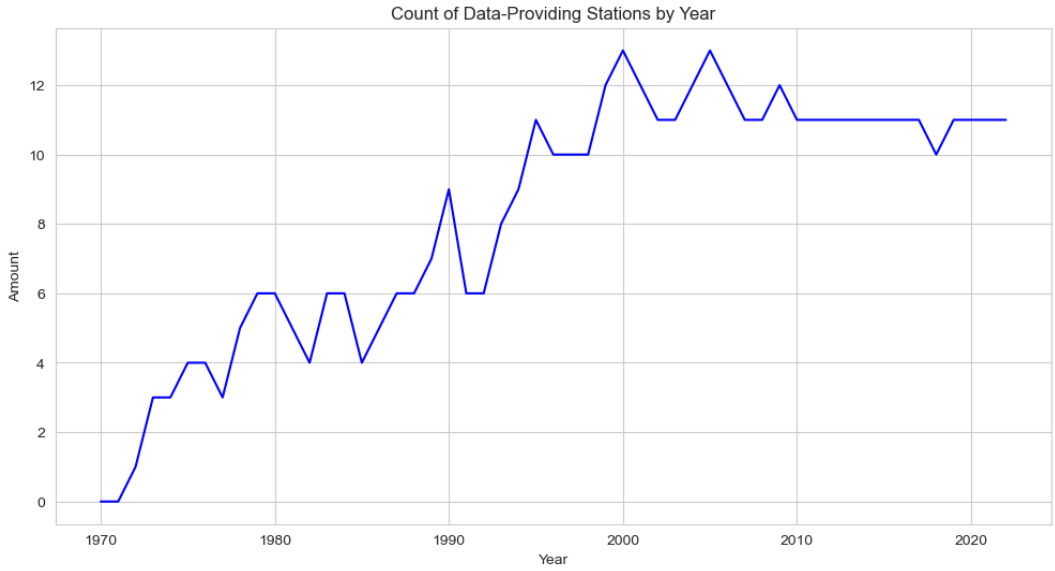


Figure 5.3: Number of available NDBC data files in area of interest by year

To minimize the occurrence of missing files, the analysis was limited to data from the past two decades. After excluding stations with no data available for the specified 20-year period, a subset of 14 stations meeting the data availability criteria was obtained. Figure 5.4 presents a heat map showcasing the distribution of the 235 available files from the selected stations across the considered years.

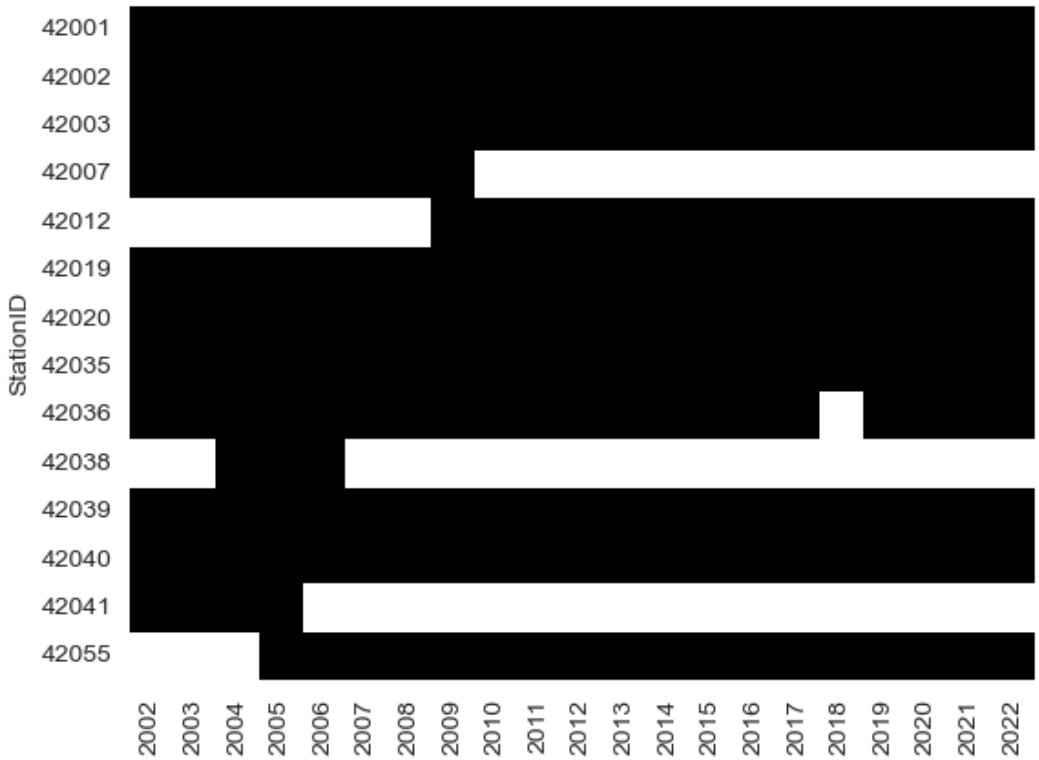


Figure 5.4: File availability distribution (Files available in black)

Test Scenarios

As elaborated in subsubsection 3.4 and further detailed in section 4.2, the toolkit already contains four predefined machine learning models specially designed for this study. All of these models possess the capability to control the influence of the ERA5 features depending on a parameter referred to as Alpha. To get insights about the behavior of those models on different levels of PIR, the Alpha value was systematically varied from 0 to 1 with increments of 0.1. This resulted in 44 different machine learning models and each of them needed to be tested with both built datasets resulting in 88 experiments as also shown in Figure 5.5.

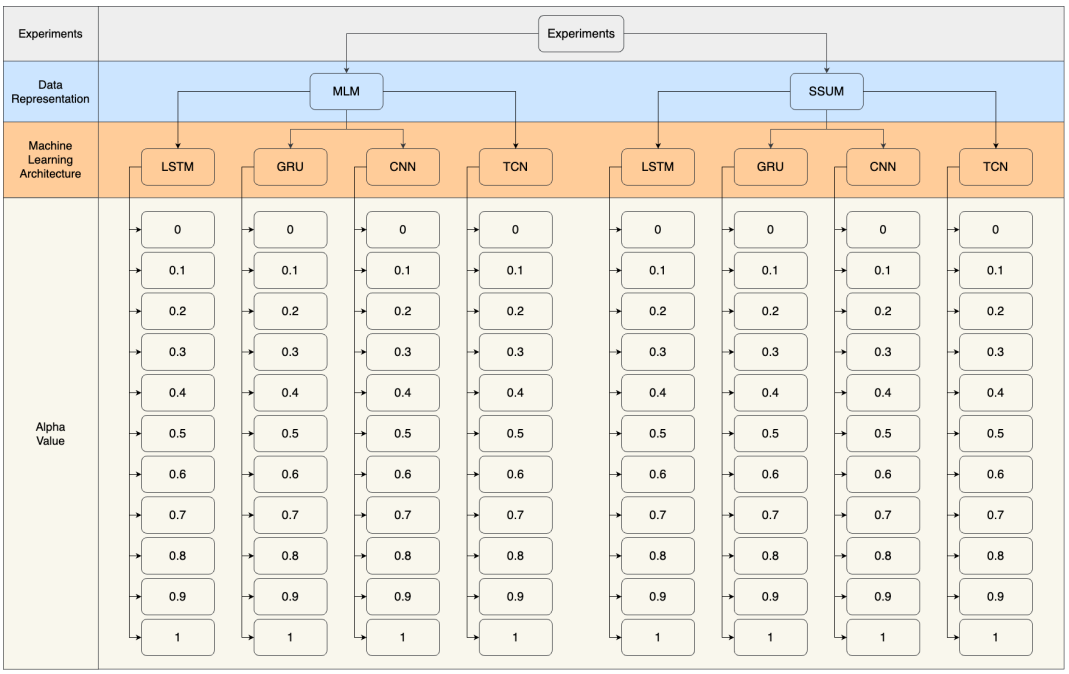


Figure 5.5: Tree diagram of test scenarios

Toolkit Utilization

Building upon the selection of NDBC datafiles, two datasets were generated adhering to the MLM and SSUM data representation methodologies. Therefore, the Jupyter notebooks `mlm_build_dataset.ipynb` and `ssum_build_dataset.ipynb` were executed. Subsequently, the two created datasets are utilized as input for the Jupyter notebooks `mlm_experiment.ipynb` and `ssum_experiment.ipynb`. The selection of the necessary parameters is depicted in Table 5.1 and Table 5.2 and expounded upon in this subsection.

Despite the station selection being motivated by the objective of minimizing missing values, it is important to note that a significant number of missing values persist within these files. This circumstance can undeniably be identified as a challenge or limitation within this project. The implemented NaN-rate based feature selection eliminates features with a NaN-rate exceeding the specified threshold. However, setting this value to 0.5 resulted in the absence of any available features in the SSUM approach. Therefore, a NaN-threshold of 0.66 was chosen, implying that all features with less than 66% missing values are considered. This resulted in a MLM dataset consisting of 6 and a SSUM dataset comprising 3 features. Within the SSUM script, the latest 24 records of a specific datafile from the most recent year were designated to serve as test data. As delineated in para-

Parameter	mlm_build_dataset.ipynb	ssum_build_dataset.ipynb
Stations	42001, 42002, 42003, 42007, 42012, 42019, 42020, 42035, 42036, 42038, 42039, 42040, 42041, 42055	42001, 42002, 42003, 42007, 42012, 42019, 42020, 42035, 42036, 42038, 42039, 42040, 42041, 42055
Time Range	2002 - 2022	2002 - 2022
NaN-Threshold	0.66	0.66
Features	WDIR, WSPD, WVHT, APD, MWD, PRES, ATMP, WTMP, DEWP	WDIR, WSPD, WVHT, APD, MWD, PRES, ATMP, WTMP , DEWP
Add ERA5 data	TRUE	TRUE
Test Station	-	42001
Year	-	2022
Num_test_hours	-	24
n_in	-	3
n_out	-	1
Filename	dataset_GOM_1_A	dataset_GOM_1_B

Table 5.1: Selected parameters for dataset builder

Parameter	mlm_experiment.ipynb	ssum_experiment.ipynb
datafile	dataset_GOM_1_A.pickle	dataset_GOM_1_B.pickle
Model	LSTM, GRU, CNN, TCN	LSTM, GRU, CNN, TCN
n_in	3	-
n_out	1	-
Station Shift	1	1
Test Hours	24	-
Alpha	`[0, 0.1, ..., 0.9, 1]`	`[0, 0.1, ..., 0.9, 1]`

Table 5.2: Selected parameters for test environment

graph 4.4.2, the parameters related to the transformation into a supervised problem are required to be specified within the dataset creation script for the SSUM data representation, whereas these values are defined within the experiment script for MLM. Three records were chosen as input and one as output, employing these values consistently across all scenarios. To compute the data differences, they were calculated relative to the preceding record, a configuration specified by setting the parameter station shift to one. Moreover, the number of test records in the MLM experiment script was aligned to 24, consistent with the configuration established during the SSUM dataset creation.

Subsequent to executing all 88 tests, their output was used as input for the report generator script. The generated error metrics were then subjected to further analysis in a spreadsheet, as detailed below.

Report Data Processing

In the process of creating the MLM dataset, several features were eliminated through the NaN-rate based feature selection process. Out of the initial 9 considered features and 14 stations, only 6 features and 9 stations remained after the selection process. Additionally, the feature `DEWP` from station 42039 and the feature `WTMP` from station 42036 have been excluded. The report generator script produces two CSV files as output: one for the MAE result and another for the MSE result of the associated experiment. As an example, Figure 5.6 presents the unprocessed MAE results of a LSTM experiment.

Experiment output: MAE - MLM - LSTM - 0.0												
	ATMP	DEWP	PRES	WDIR	WSPD	WTMP	ATMP _ERA5	DEWP _ERA5	PRES _ERA5	WSPD _ERA5	WTMP _ERA5	WDIR _ERA5
42001	0.62	0.63	0.42	46.14	1.18	0.09	0.24	0.22	0.40	0.47	0.02	16.44
42002	0.21	0.04	0.40	29.89	0.75	0.02	0.15	0.13	0.35	0.58	0.02	23.75
42003	0.20	0.11	0.03	0.27	0.15	0.05	0.26	0.16	0.49	0.59	0.01	9.92
42019	0.24	0.51	0.39	0.10	0.59	0.11	0.18	0.15	0.40	0.55	0.01	21.83
42020	0.45	0.57	0.44	28.53	0.71	0.41	0.13	0.25	0.43	0.38	0.01	24.42
42035	0.11	0.09	0.05	0.20	0.05	0.07	0.17	0.15	0.36	0.49	0.04	22.27
42036	0.41	0.48	0.57	24.95	1.28	-	0.16	0.21	0.41	0.67	-	8.72
42039	0.69	-	0.61	37.70	2.20	0.21	0.38	-	0.43	0.81	0.02	8.71
42040	0.43	0.23	0.57	11.65	1.17	0.14	0.22	0.24	0.35	0.98	0.02	15.37

Figure 5.6: Unprocessed MAE Results of MLM experiment with LSTM architecture and Alpha=0

In contrast, the SSUM approach considers features across all stations when executing the NaN-based feature selection. Only three features passed the filtering with the considered threshold of 66%. In the case of SSUM, the report generator creates only one CSV file that contains both error metrics as exemplified in Figure 5.7.

Experiment output: SSUM - LSTM - 0.0		
Feature	**MAE**	**MSE**
PRES(t)	0.46	0.28
WSPD(t)	1.30	3.04
WDIR(t)	46.05	3963.54
PRES_ERA5(t)	0.43	0.23
WSPD_ERA5(t)	7.94	172.95
WDIR_ERA5(t)	15.88	813.11

Figure 5.7: Unprocessed error rates of SSUM experiment with LSTM architecture and Alpha=0

For evaluating the collected data, MLM and SSUM were investigated separately for each considered architecture in separate spreadsheets. First, two tables representing the MAE and MSE of each feature across all Alpha values were created. Given this project's exclusive focus on forecasting NDBC observations, ERA5 features were omitted from the evaluation. In the context of MLM, the average values of similar features across various stations were incorporated. This streamlined the assessment of the predictability of a specific feature and enabled a comparison with SSUM results.

Additionally, the absolute improvement of each value compared to the equivalent experiment with the Alpha value of 0 were created. Experiments with this particular Alpha value serve as baselines, representing the absence of any physics-informed regulation. The poorest outcome is typically observed when Alpha is set to 0, and this is quite evident since this experiment completely excludes the use of actual observations. As mentioned in subsection 2.4.3, care should be taken when making comparisons between ERA5 wind measurements and actual observations. The validity of this statement is evident from the initial observations of the `WDIR` forecasts as this feature is the only one showing most worst result with other Alpha values than 1. Furthermore, it includes many outliers. However, this is not the case for `WSPD`. To prevent potential misinterpretations due to the influence of experiments based solely on ERA5 data or the `WDIR` feature, this data was excluded during model improvement evaluation.

As these improvements represent absolute values within the context of the specific feature's scale, it becomes challenging to interpret the overall enhancement of a particular Alpha value. Hence, also the relative improvement across all features in percentage based on the baseline models was calculated for each considered Alpha value.

The performance evaluation of the models across all Alpha values is grounded in the MAE, MSE, absolute improvement of each feature, and the relative improvement across all features. Tables and visuals detailing these

can be found in subsequent section, segmented by dataset and model.

5.2 Results and Findings

This section delves into the results of the conducted experiments by analyzing tables that display feature-wise outcomes and charts that illustrate both the absolute improvement of each feature and the overall relative enhancement, as described in subsubsection 5.1. Improvements are benchmarked against the baseline experiment, which is characterized by an Alpha value of 1, signifying an experiment without PIR. Results are separately discussed for each dataset-model pair. The presented findings form the foundation for addressing the question of whether physics-informed regularization can indeed enhance the forecasting accuracy of real-world observations and identifying the level of regularization that offers the most promising results.

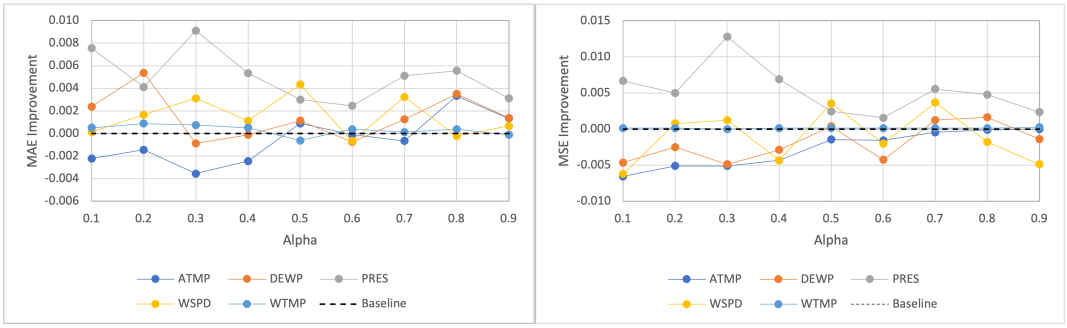
5.2.1 MLM-LSTM

Average MAE of LSTM Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	0.3729	0.3309	0.3869	0.8973	0.1383
0.1	0.3036	0.2953	0.3708	0.8822	0.0548
0.2	0.3028	0.2923	0.3742	0.8807	0.0544
0.3	0.3049	0.2985	0.3692	0.8792	0.0545
0.4	0.3038	0.2978	0.3730	0.8812	0.0548
0.5	0.3004	0.2965	0.3753	0.8780	0.0559
0.6	0.3014	0.2984	0.3759	0.8830	0.0549
0.7	0.3020	0.2964	0.3732	0.8791	0.0551
0.8	0.2980	0.2941	0.3728	0.8826	0.0549
0.9	0.3000	0.2963	0.3752	0.8817	0.0554
1	0.3013	0.2976	0.3783	0.8823	0.0553

Average MSE of LSTM Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	0.3307	0.3508	0.3033	2.1114	0.0436
0.1	0.3029	0.3451	0.2814	2.1027	0.0193
0.2	0.3014	0.3430	0.2831	2.0957	0.0193
0.3	0.3014	0.3454	0.2753	2.0952	0.0194
0.4	0.3007	0.3434	0.2812	2.1008	0.0193
0.5	0.2978	0.3401	0.2857	2.0929	0.0193
0.6	0.2979	0.3448	0.2866	2.0984	0.0193
0.7	0.2968	0.3393	0.2826	2.0928	0.0193
0.8	0.2964	0.3389	0.2833	2.0982	0.0193
0.9	0.2963	0.3419	0.2858	2.1013	0.0191
1	0.2963	0.3405	0.2881	2.0964	0.0194

Figure 5.8: Average MAE and MSE of LSTM architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted.

In Figure 5.8, the Alpha value of 0, representing the experiment solely considering ERA5 data, yields the least favorable outcome. Furthermore, this figure shows that the MSE of `ATMP` reaches its finest performance in the absence of Physics-Informed Regularization (Alpha=1). In contrast, the superior outcomes for other features are reached using Alpha values that vary between 0.2 and 0.9.



a) MAE b) MSE

Figure 5.9: MLM-LSTM: Absolute improvement in relation to baseline experiment (Alpha=1)

Figure 5.9 displays the absolute MAE and MSE improvement of each feature compared to the experiment that only considered NDBC data. The feature PRES consistently shows improvement across all Alpha values. However, it is worth noting that 13 out of 45 data points exhibit a deterioration in MAE and 19 in MSE. The MSE for ATMP worsens for all Alpha values except when set to 0.9, where it experiences a slight improvement. Furthermore, it is important to emphasize that the alterations in accuracy, whether improvement or deterioration, are generally minor. The most significant enhancement in a single feature's performance is evident when Alpha is set to 0.3, leading to a decrease of 0.128 in MSE for PRES. In contrast, all other outcomes exhibit variations of less than one-hundredth. The Alpha value of 0.7 stands out as it results in an improvement in every feature, except for ATMP. Furthermore, it is noticeable that the feature WTMP exhibits minor variations in terms of MAE. However, the MSE appears to be almost unaffected by changes in the Alpha value.

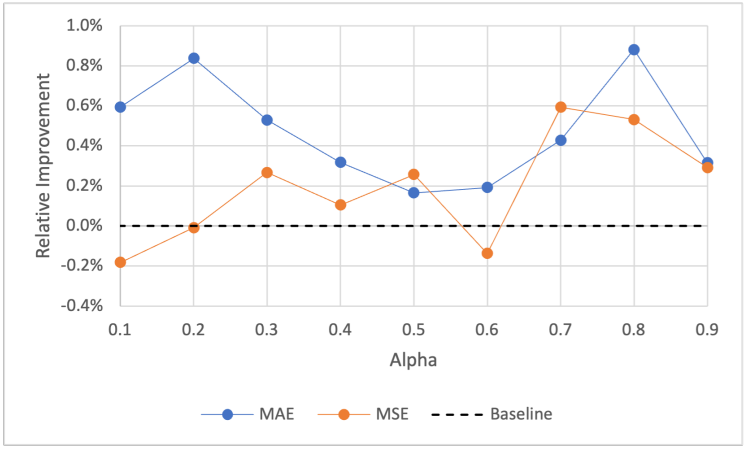


Figure 5.10: MLM-LSTM: Average relative MAE and MSE improvement in percent across all features compared to baseline

The overall improvements achieved with each alpha value compared to the experiment solely relying on NDBC data are illustrated in Figure 5.10. As illustrated, every level of physics-informed regularization results in an overall MAE improvement. However, the relative enhancement remains below 1% for each Alpha value. Alpha values of 0.1, 0.2, and 0.6 lead to an average MSE distortion across all features. Furthermore, all variations in the average MSE remain within a 1% range. Concerning MAE, Alpha value of 0.2 exhibits the best average improvement, which is 0.837%. In the context of MSE, Alpha=0.7 displays the most substantial average improvement, standing at 0.593%. Additionally, this specific Alpha value illustrates a satisfactory average MAE improvement of 0.428%.

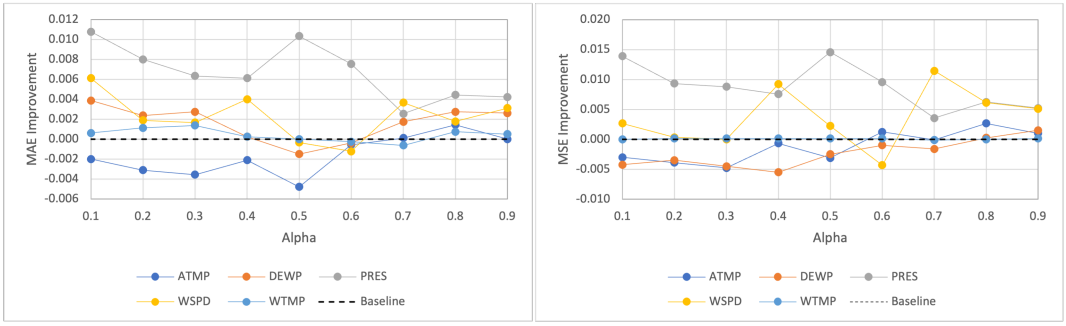
5.2.2 MLM-GRU

Average MAE of GRU Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	0.6053	0.4438	0.4757	1.0514	0.5194
0.1	0.3022	0.2948	0.3706	0.8771	0.0553
0.2	0.3033	0.2963	0.3733	0.8813	0.0548
0.3	0.3038	0.2959	0.3750	0.8816	0.0545
0.4	0.3023	0.2984	0.3752	0.8792	0.0556
0.5	0.3050	0.3001	0.3710	0.8836	0.0559
0.6	0.3008	0.2990	0.3738	0.8844	0.0561
0.7	0.3001	0.2969	0.3788	0.8796	0.0565
0.8	0.2988	0.2959	0.3769	0.8814	0.0551
0.9	0.3002	0.2960	0.3771	0.8801	0.0554
1	0.3002	0.2986	0.3813	0.8832	0.0559

Average MSE of GRU Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	0.5998	0.4299	0.4284	2.3840	0.3950
0.1	0.3001	0.3446	0.2796	2.0972	0.0194
0.2	0.3010	0.3439	0.2841	2.0996	0.0193
0.3	0.3019	0.3449	0.2847	2.0999	0.0193
0.4	0.2978	0.3459	0.2859	2.0907	0.0193
0.5	0.3002	0.3429	0.2789	2.0977	0.0193
0.6	0.2959	0.3414	0.2839	2.1042	0.0193
0.7	0.2972	0.3420	0.2899	2.0884	0.0195
0.8	0.2944	0.3401	0.2872	2.0938	0.0194
0.9	0.2961	0.3389	0.2882	2.0948	0.0193
1	0.2971	0.3404	0.2934	2.0999	0.0194

Figure 5.11: Average MAE and MSE of GRU architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted.

In Figure 5.11, it is evident that when Alpha is set to 0, the performance of each feature is at its lowest because the actual observations are disregarded. Furthermore, the Alpha value of 1 which solely relies on NDBC data never attains the best result. It is noticeable, that Alpha values within the range of 0.2-0.6 and 0.9 yield identical MSE results for `WTMP`, which also represents the superior outcome. Nevertheless, the Alpha value of 0.3 stands out as the one showcasing the best MAE result. For `WSPD`, the optimal MAE is achieved with an Alpha value of 0.1. However, for the other features, the best MAE and the optimal MSE are observed at the lowest PIR levels.



a) MAE

b) MSE

Figure 5.12: MLM-GRU: Absolute improvement in relation to baseline experiment (Alpha=1)

Improvement and deterioration of all features at various Alpha values, compared to the GRU experiment that solely considers NDBC data, is shown

in Figure 5.12. The feature PRES demonstrates enhancements in both, MAE and MSE across every Alpha value. Conversely, ATMP displays a decline with intense PIR levels for both MAE and MSE metrics. With Alpha values of 0.7 or lower, DEWP faces a decrease in MSE, though a decrease in MAE is only evident at Alpha values of 0.5 and 0.6. It is significant to mention that for Alpha values of 0.8 and 0.9, there is an improvement for all features in terms of both metrics. Overall, the discernible variations in performance, whether advantageous or detrimental, remain marginal.

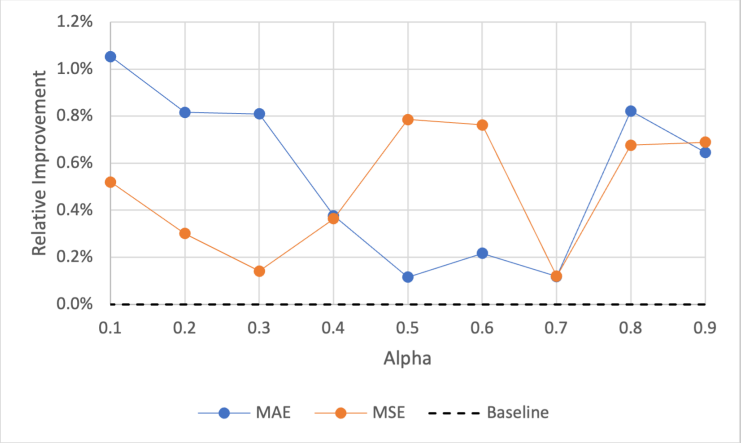


Figure 5.13: MLM-GRU: Average relative MAE and MSE improvement in percent across all features compared to baseline

Figure 5.13 indicates that the overall accuracy improves with any level of PIR compared to the baseline experiment that did not incorporate any physics-informed data. However, Alpha of 0.1 is the sole value that results in an improvement exceeding 1% in terms of MAE. The best MSE result is observed with an Alpha value of 0.5. It is also important to highlight that the Alpha value of 0.8, which yields improvements in each individual feature, achieves the second-best average MAE and ranks fourth in terms of MSE, with a minimal deviation of approximately 0.1% from the top-performing result.

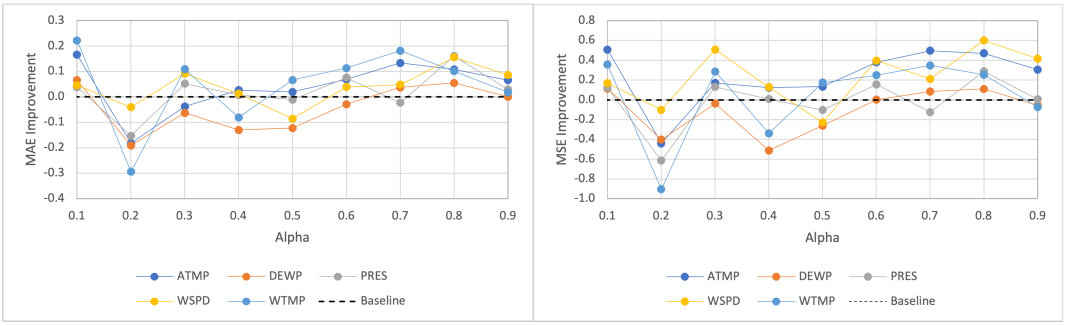
5.2.3 MLM-CNN

Average MAE of CNN Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	23.283	23.567	23.848	24.641	22.048
0.1	0.398	0.416	0.550	1.051	0.170
0.2	0.746	0.672	0.741	1.137	0.687
0.3	0.602	0.544	0.536	1.005	0.284
0.4	0.538	0.611	0.580	1.084	0.474
0.5	0.544	0.604	0.599	1.182	0.327
0.6	0.495	0.510	0.512	1.057	0.280
0.7	0.431	0.444	0.611	1.049	0.212
0.8	0.456	0.427	0.427	0.941	0.292
0.9	0.498	0.481	0.559	1.011	0.375
1	0.564	0.481	0.588	1.096	0.393

Average MSE of CNN Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	1590.464	1728.229	1845.293	1705.007	1374.208
0.1	0.413	0.453	0.527	2.683	0.095
0.2	1.362	0.969	1.264	2.953	1.356
0.3	0.751	0.604	0.517	2.342	0.166
0.4	0.799	1.078	0.641	2.722	0.791
0.5	0.789	0.827	0.751	3.079	0.277
0.6	0.543	0.565	0.493	2.452	0.204
0.7	0.426	0.481	0.773	2.640	0.104
0.8	0.451	0.457	0.360	2.249	0.198
0.9	0.616	0.621	0.642	2.434	0.523
1	0.921	0.566	0.648	2.850	0.452

Figure 5.14: Average MAE and MSE of CNN architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted.

In Figure 5.14, the mean error rates across all Alpha values within the CNN architecture are compared to those of the CNN baseline experiment that only considers NDBC data. Mirroring observations from the previously discussed recurrent neural networks, each feature's least favorable outcome occurs with an Alpha value of 0. The best values for each feature are evident in only two different experiments. Specifically, for the features `ATMP`, `DEWP`, and `WTMP`, an Alpha value of 0.1 yields the most superior results. Conversely, for `PRES` and `WSPD`, the lowest error rates are observed at an Alpha value of 0.8. Furthermore, it should be highlighted that within the CNN architecture, the Alpha value associated with the best MAE consistently also produces the best MSE.



a) MAE

b) MSE

Figure 5.15: MLM-CNN: Absolute improvement in relation to baseline experiment (Alpha=1)

As illustrated in Figure 5.15, within the CNN framework, the patterns of MAE and MSE are strikingly similar with MSE exhibiting greater variation.

When contrasted with the experimented recurrent neural networks, the influence of PIR on both MAE and MSE is more pronounced, with differences reaching up to 0.3 for MAE and 0.9 for MSE. No single feature consistently improves or deteriorates across the entire range of Alpha values. Notably, an Alpha value of 0.2 leads to a decline in accuracy for all features, whereas Alpha values of 0.1 and 0.8 enhances accuracy across the board.

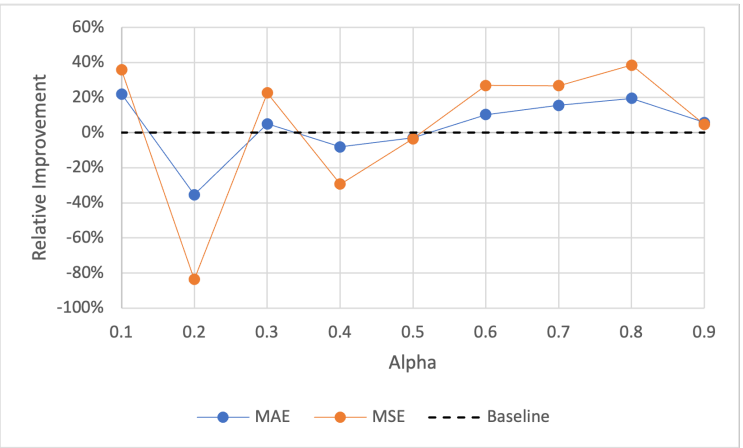


Figure 5.16: MLM-CNN: Average relative MAE and MSE improvement in percent across all features compared to baseline

Figure 5.16 displays the cross-feature improvement of MLM-CNN experiments in comparison to the MLM-CNN experiment solely based on observation data. It is particularly noticeable that the range is defined by a potential increase of up to 35.73% and a reduction exceeding 80%. It is important to keep in mind that an improvement represents an decrease of the error which means cutting the error in half corresponds to a relative improvement of 50%, while doubling the error leads to a relative deterioration of 100%. Across both metrics, the average error consistently moves in the same direction, either improving or declining. Furthermore, it's worth noting that any Alpha value above 0.5 exhibits an enhancement in the mean error across all features, indicating that minimal PIR levels result in stable enhancement.

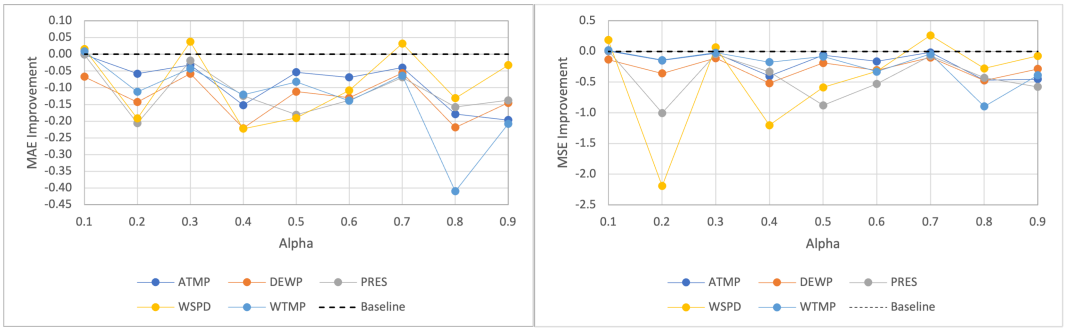
5.2.4 MLM-TCN

Average MAE of TCN Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	11.835	9.917	10.127	8.801	12.237
0.1	0.395	0.421	0.445	0.971	0.142
0.2	0.452	0.497	0.650	1.179	0.264
0.3	0.426	0.412	0.463	0.949	0.194
0.4	0.546	0.575	0.568	1.210	0.272
0.5	0.447	0.466	0.625	1.178	0.234
0.6	0.463	0.484	0.582	1.095	0.290
0.7	0.433	0.410	0.513	0.955	0.215
0.8	0.572	0.573	0.602	1.118	0.561
0.9	0.590	0.499	0.582	1.020	0.359
1	0.394	0.354	0.444	0.987	0.151

Average MSE of TCN Architecture					
Alpha	**ATMP**	**DEWP**	**PRES**	**WSPD**	**WTMP**
0	1279.276	605.550	905.812	544.298	1028.985
0.1	0.409	0.501	0.365	2.233	0.059
0.2	0.561	0.723	1.379	4.610	0.219
0.3	0.445	0.475	0.421	2.351	0.098
0.4	0.818	0.884	0.707	3.621	0.251
0.5	0.474	0.554	1.254	3.002	0.160
0.6	0.576	0.672	0.902	2.743	0.407
0.7	0.430	0.465	0.448	2.157	0.120
0.8	0.886	0.840	0.804	2.695	0.973
0.9	0.867	0.651	0.953	2.490	0.465
1	0.413	0.365	0.375	2.417	0.076

Figure 5.17: Average MAE and MSE of TCN architecture over all Alpha values using MLM dataset: Values of a certain feature represent the average result across all stations. The best (green) and worst (red) result of each feature are highlighted.

In line with previously discussed test results, Figure 5.17 shows that the TCN model also exhibits the poorest performance in terms of MAE and MSE across all features when based solely on ERA5 data. Contrary to other models, the TCN achieves top performance for three out of the five features in terms of MAE and for one feature in terms of MSE when utilizing the base model devoid of PIR. Conversely, three peak MSE outcomes are attained with the most pronounced level of PIR, set at an Alpha value of 0.1.



a) MAE

b) MSE

Figure 5.18: MLM-TCN: Absolute improvement in relation to baseline experiment (Alpha=1)

Figure 5.18 shows the absolute deviation of each feature at each Alpha value compared to the MLM-TCN experiment that only considers NDBC data. It can be noticed that almost all PIR levels across all features lead to a reduction in accuracy. Solely four values in MAE and five in MSE show a positive outcome.

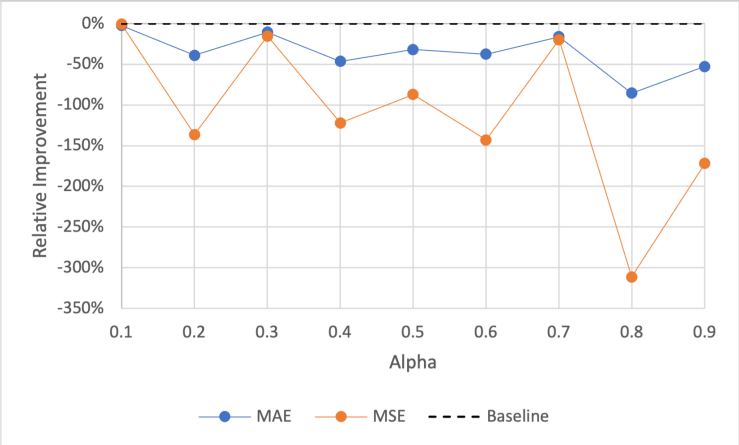


Figure 5.19: MLM-TCN: Average relative MAE and MSE improvement in percent across all features compared to baseline

The relative cross-feature improvement of MLM-TCN experiments, compared to the MLM-TCN baseline that does not consider any level of PIR, is illustrated in Figure 5.19. It demonstrates that with every Alpha value, the average error across all features sees an accuracy decrease, in many instances significantly.

5.2.5 SSUM-LSTM

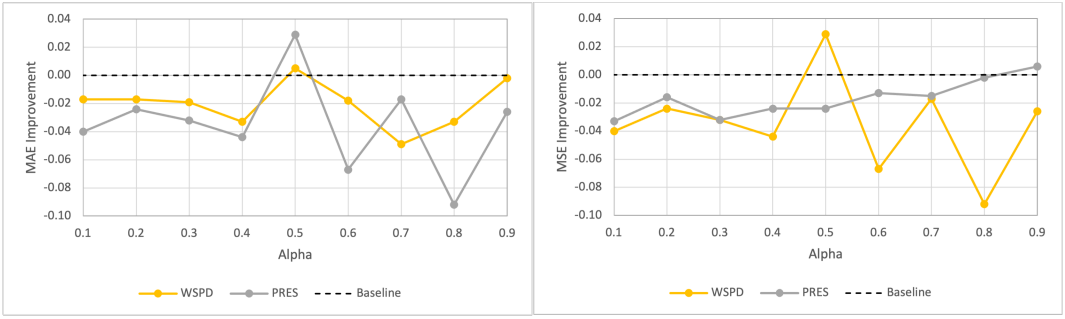
SSUM - LSTM - MAE		
Alpha	**WSPD**	**PRES**
0	1.298	3.042
0.1	1.235	2.849
0.2	1.235	2.833
0.3	1.237	2.841
0.4	1.251	2.853
0.5	1.213	2.780
0.6	1.236	2.876
0.7	1.267	2.826
0.8	1.251	2.901
0.9	1.220	2.835
1	1.218	2.809

SSUM - LSTM - MSE		
Alpha	**WSPD**	**PRES**
0	3.042	0.275
0.1	2.849	0.239
0.2	2.833	0.222
0.3	2.841	0.238
0.4	2.853	0.230
0.5	2.780	0.230
0.6	2.876	0.219
0.7	2.826	0.221
0.8	2.901	0.208
0.9	2.835	0.200
1	2.809	0.206

Figure 5.20: MAE and MSE of LSTM architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted.

As shown in Figure 5.20, the least favorable outcomes consistently occur with an Alpha value of 0. The MAE of PRES as well as both metrics of the

feature WSPD reach the best result with an Alpha value of 0.5 while the top result of PRES in terms of MSE is reached with an Alpha value of 0.9.



a) MAE b) MSE

Figure 5.21: SSUM-LSTM: Absolute improvement in relation to baseline experiment (Alpha=1)

Figure 5.21 illustrates the feature-wise improvement of SSUM-LSTM experiments for each Alpha value, compared to the experiment with an Alpha value of one, which results in no physics-informed regulation. It shows that the majority of PIR experiments lead to a decrease in accuracy. For MAE, the only notable enhancement comes with an Alpha value of 0.5, affecting both features. The MSE only improves for WSPD at an Alpha of 0.5 and for PRES at an Alpha of 0.9. All remaining results are negative.

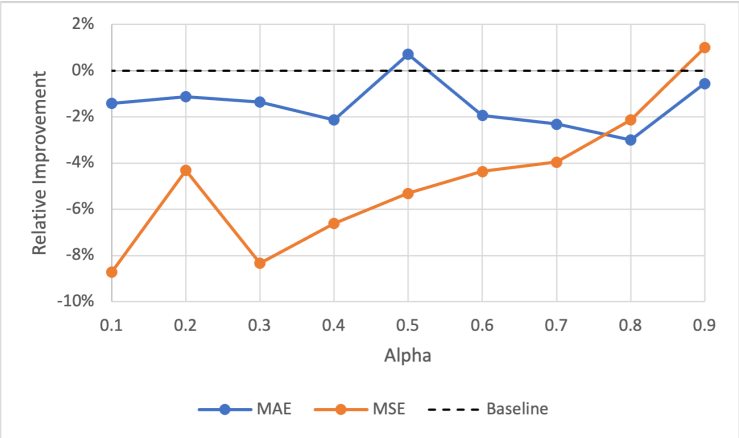


Figure 5.22: SSUM-LSTM: Average relative MAE and MSE improvement in percent across all features compared to baseline

Figure 5.22 confirms that most PIR experiments lead to a deterioration. It illustrates the cross-feature deviation compared to the baseline experiment, which does not incorporate any level of PIR. Both MAE and MSE see one

marginal improvement of less than 1%, with the former at an Alpha of 0.5 and the latter at 0.9.

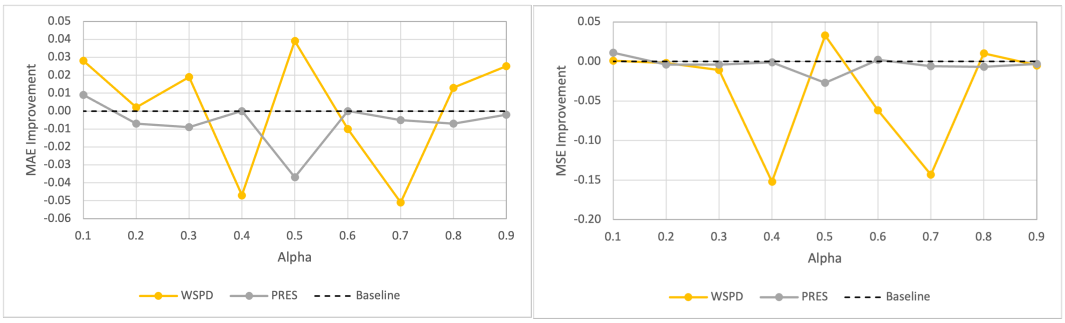
5.2.6 SSUM-GRU

SSUM - GRU - MAE		
Alpha	**WSPD**	**PRES**
0	1.228	1.865
0.1	1.212	0.393
0.2	1.238	0.409
0.3	1.221	0.411
0.4	1.287	0.402
0.5	1.201	0.439
0.6	1.250	0.402
0.7	1.291	0.407
0.8	1.227	0.409
0.9	1.215	0.404
1	1.240	0.402

SSUM - GRU - MSE		
Alpha	**WSPD**	**PRES**
0	2.901	3.813
0.1	2.820	0.202
0.2	2.823	0.217
0.3	2.832	0.217
0.4	2.973	0.214
0.5	2.788	0.240
0.6	2.883	0.211
0.7	2.964	0.219
0.8	2.811	0.220
0.9	2.826	0.216
1	2.821	0.213

Figure 5.23: MAE and MSE of GRU architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted.

As shown in Figure 5.23, the GRU model does not register the poorest performance for the WSPD feature at an Alpha value of 0. The worst MAE is observed at 0.7, while the most suboptimal MSE is at 0.4. The optimal performance for this feature, for both metrics, occurs at an Alpha value of 0.5. For the PRES feature, the peak accuracy is achieved at an Alpha value of 0.1, while the least favorable result is observed at an Alpha value of 0.



a) MAE

b) MSE

Figure 5.24: SSUM-GRU: Absolute improvement in relation to baseline experiment (Alpha=1)

Figure 5.24 depicts the absolute improvement of the considered Alpha values in relation to the SSUM-GRU experiment that excludes physics-informed data. It shows that the Alpha value of 0.1 stands out as the sole value where both features exhibit improvement. Additionally, it is the only Alpha setting where MAE and MSE enhance simultaneously for any single feature. The value 0.5 stands out as it corresponds to the best result for `WSPD`, while simultaneously representing the least favorable outcome for `PRES`. Furthermore, it is observed that the variations in improvement and deterioration are limited to the range of tenths and thousandths.

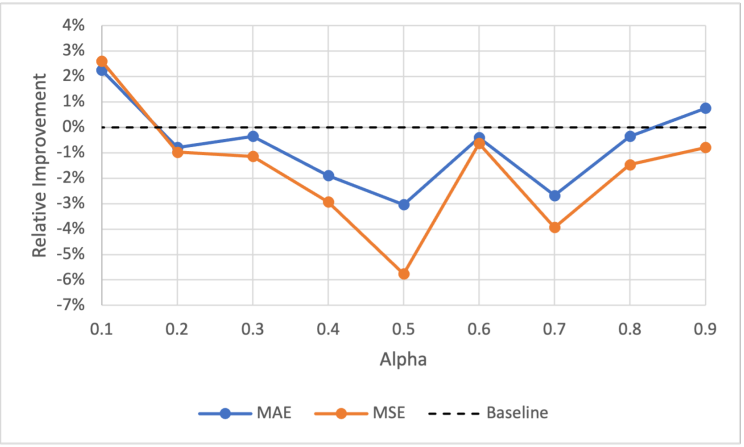


Figure 5.25: SSUM-GRU: Average relative MAE and MSE improvement in percent across all features compared to baseline

In Figure 5.25 the relative error across both features of each SSUM-GRU experiment is compared to the baseline experiment, which is solely based on NDBC data. An Alpha value of 0.1 emerges as the only configuration yielding enhancements in both error metrics. Additionally, there is a discernible improvement in MAE at the Alpha value of 0.9. The remaining settings show a clear decline in accuracy.

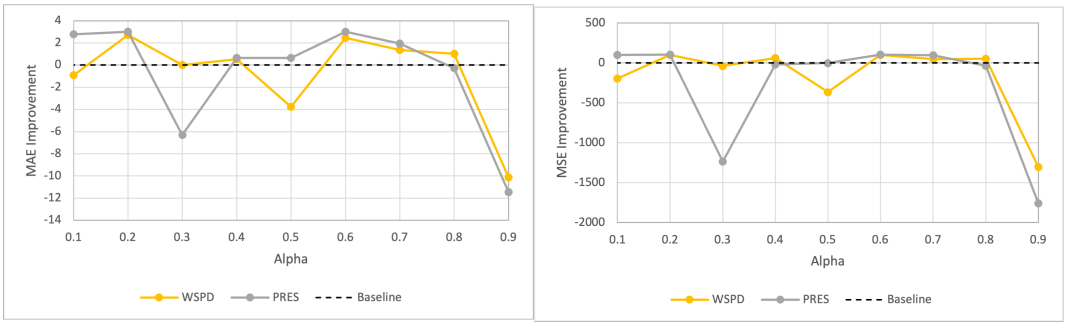
5.2.7 SSUM-CNN

SSUM - CNN - MAE		
Alpha	**WSPD**	**PRES**
0	137.152	136.287
0.1	5.374	1.252
0.2	1.746	1.033
0.3	4.465	10.321
0.4	3.937	3.403
0.5	8.230	3.390
0.6	2.009	1.017
0.7	3.095	2.104
0.8	3.436	4.312
0.9	14.589	15.477
1	4.466	4.034

SSUM - CNN - MSE		
Alpha	**WSPD**	**PRES**
0	93249.087	76515.863
0.1	303.961	9.263
0.2	6.698	2.886
0.3	144.922	1343.809
0.4	47.818	127.207
0.5	473.114	110.104
0.6	6.724	2.921
0.7	57.936	11.769
0.8	54.563	142.359
0.9	1412.946	1866.826
1	108.546	109.748

Figure 5.26: MAE and MSE of CNN architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted.

As delineated in Figure 5.26, the least favorable outcomes are observed at an Alpha value of 0.6. For the PRES feature, the MAE is optimized at an Alpha value of 0.6. The MAE of WSPD and the MSE for both features yield their best outcomes at an Alpha value of 0.2.



a) MAE

b) MSE

Figure 5.27: SSUM-CNN: Absolute improvement in relation to baseline experiment (Alpha=1)

The absolute improvement exhibited by the CNN architecture and benchmarked against the CNN baseline experiment which excludes any physics-informed data is illustrated in Figure 5.27. The minimal level of PIR yields the least favorable outcomes for both MAE and MSE. A significant disparity in the scales between MAE and MSE is evident. To illustrate, while the largest drop in MAE amounts to a decline of 11.44, the most substantial deterioration in MSE is 1757.1 units below the baseline. Notably, Alpha values of 0.6 and 0.7 demonstrate the ability to enhance forecast accuracy in both metrics.

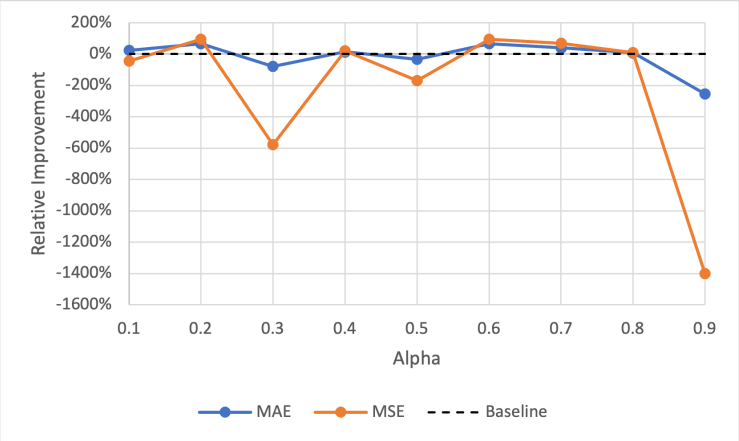


Figure 5.28: SSUM-CNN: Average relative MAE and MSE improvement in percent across all features compared to baseline

The disparity in scale between MAE and MSE is similarly evident in Figure 5.28. This figure illustrates the relative cross-feature deviation of error rates for all Alpha values in comparison to the experiment completely excluding ERA5 data. For Alpha values of 0.3, 0.5, and 0.9, a decline surpassing 100% is observed in terms of MSE, but this substantial drop only pertains to one MAE value. In contrast, the most pronounced improvements are seen at 67.65% for MAE and 95.6% for MSE, both achieved with an Alpha value of 0.2. Notably, while some Alpha values markedly degrade results, it's significant that half of the evaluated Alpha values lead to improvements.

5.2.8 SSUM-TCN

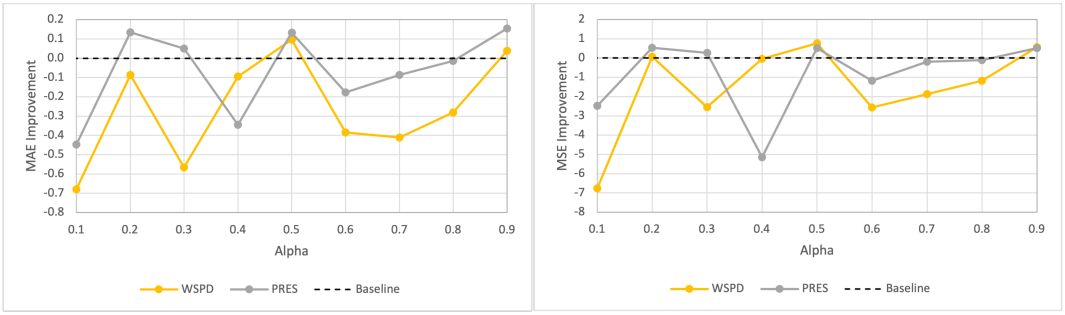
SSUM - TCN - MAE		
Alpha	**WSPD**	**PRES**
0	73.768	115.003
0.1	1.998	1.037
0.2	1.404	0.455
0.3	1.884	0.539
0.4	1.412	0.935
0.5	1.221	0.458
0.6	1.702	0.767
0.7	1.729	0.677
0.8	1.599	0.604
0.9	1.279	0.436
1	1.318	0.590

SSUM - CNN - MSE		
Alpha	**WSPD**	**PRES**
0	68379.164	155323.237
0.1	10.506	3.299
0.2	3.675	0.282
0.3	6.294	0.538
0.4	3.789	5.961
0.5	2.974	0.297
0.6	6.307	1.990
0.7	5.612	1.002
0.8	4.921	0.922
0.9	3.172	0.297
1	3.742	0.814

Figure 5.29: MAE and MSE of TCN architecture over all Alpha values using SSUM dataset: The best (green) and worst (red) result of each feature are highlighted.

In Figure 5.29, it can be observed that also the TCN architecture yields its poorest result when the model relies solely on ERA5 data. In both metrics,

the feature WSPD attains its optimal performance at an Alpha value of 0.5. For the PRES feature, the best MAE is observed at an Alpha value of 0.9, whereas the best MSE value is achieved with an Alpha value of 0.2.



a) MAE b) MSE

Figure 5.30: SSUM-TCN: Absolute improvement in relation to baseline experiment (Alpha=1)

Figure 5.30 depicts the absolute improvement of PIR-based experiments using the TCN architecture on the SSUM dataset. The baseline is defined by the SSUM-TCN experiment executed with an Alpha value of one, as this setting signifies the absence of physics-informed influence. A strong correlation is evident between the two metrics under consideration. Only Alpha values of 0.5, and 0.9 lead to improvements in both MAE and MSE across the features. Additionally, there's a tenfold difference in the scale between MAE and MSE.

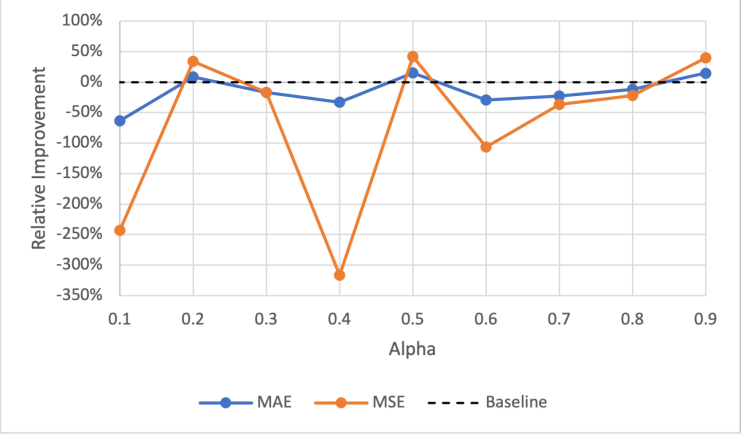


Figure 5.31: SSUM-TCN: Average relative MAE and MSE improvement in percent across all features compared to baseline

The relative cross-feature improvement of all SSUM-TCN experiments in comparison to the result from the SSUM-TCN experiment based solely

on NDBC data is illustrated in Figure 5.31. It indicates that only the Alpha values 0.2, 0.5 and 0.9 are capable to result in an improved accuracy. While remaining values result in a deterioration, those show an significant improvement in both error metrics. Notably, some Alpha values significantly degrade the results, with MSE being particularly impacted. The most adverse outcome occurs with an Alpha value of 0.4, leading to an MSE deterioration of 316.8%.

5.3 Discussion

This section begins by discussing most important findings from the previous section. Furthermore, it addresses the research question and points out the contribution of this thesis.

Scope and Limitations

The empirical research is limited to a selection of 14 stations situated within the Gulf of Mexico. Due to feature selection processes, primarily influenced by missing values and the exclusion of wind speed data, the MLM dataset encompasses five features, while the SSUM dataset is restricted to two. The study further confines its exploration to four specific neural network models. Additionally, there was no optimization of hyperparameters, and the Alpha values were adjusted in discrete increments of 0.1.

Main Discoveries

As demonstrated in the preceding section, the outcomes vary significantly depending on the chosen dataset representation and the selected architecture. Subsection 5.2.6 outlines that the SSUM-GRU experiment stands out as the Alpha value of 0 does not produce the least favorable results for every feature. Specifically, within this experiment, `WSPD` registers its least favorable outcomes at Alpha values of 0.7 for MAE and 0.4 for MSE. The difference across all Alpha values for this feature is notably limited to less than one-tenth for MAE and two-tenths for MSE. The system, when set with an Alpha value of 0, exclusively utilizes the modelled ERA5 data, bypassing the NDBC observations during weight adjustments in the loss function as detailed in section 4.2. The predominance of negative results with this value indicates that relying only on modelled data to predict ocean buoy observations is not recommended.

While most tests indicated enhancements across all features with various Alpha values, the LSTM analysis with the MLM dataset observed a consistent

drop in MSE for the feature ATMP, irrespective of the Alpha setting. Moreover, the MAE of ATMP, DEWP and PRES as well as the MSE of DEWP consistently decline across all Alpha values smaller than one when considering the TCN network and the MLM dataset. In contrast, in those experiments, all other features and indeed all features in other tests reached their peak performance with a certain degree of PIR. Based on this, one can infer that PIR offers advantages for time series forecasting. However, the optimal Alpha value depends on the chosen mode and dataset as well as on the feature of interest.

A noteworthy difference in the significance of improvement and deterioration is noticed. The recurrent neural networks mostly displayed results with variations under 5%, in contrast to the more notable fluctuations, both upward and downward, seen in CNN and TCN architectures.

Considering the MLM approach for dataset creation, the stable improvement within the CNN architecture at low level PIR must be emphasized. For all Alpha values greater than 0.6, there is a consistent relative improvement across all features. A marked increase in accuracy, reaching up to 19.54% for MAE and 38.42% for MSE is observed. Additionally, every single feature registers a boost in MAE for Alpha values greater than 0.7 and the MSE improvement for all features at an Alpha of 0.8.

Recurrent neural networks also appear to have the potential for improvements when the discussed PIR method is implemented. Nevertheless, the executed experiments rarely indicate an enhancement beyond 1% for individual features. Furthermore, identifying a solid Alpha value can be challenging as it depends on the feature of interest and whether it is more important to improve the MAE or the MSE in the specific problem. On the one hand, if the primary concern is to minimize the impact of outliers and provide a more robust model, the focus should be on MAE. On the other hand, if the goal is to heavily penalize large errors and ensure that the model is sensitive to such deviations, then optimizing for MSE is more appropriate.

Test results suggest that the integration of PIR fails to yield improvements for a TCN that has been trained using the MLM dataset.

Similar to the MLM approach, the recurrent neural networks using the SSUM dataset strategy yield only slight variations in results. However, only specific Alpha values are capable of improving the accuracy of individual features. The fact that the SSUM-GRU experiment yielded the best result for the feature PRES by giving just a 10% weight to the actual observation data is difficult to explain. Without diving deeper, this is assumed to be unique to the given dataset. Hence, the MLM approach takes precedence over the SSUM method when it comes to using GRU or LSTM architectures. Another

similarity between both dataset creation strategies is that the CNN displays the most significant overall improvement. Disregarding outcomes where Alpha is set to 0, the least favorable result from the CNN using the MLM dataset still outperforms the best from the same feature and metric with the SSUM dataset. Consequently, the MLM strategy appears to be the more optimal choice for CNN as well.

Using the SSUM dataset with a TCN model and an Alpha value of 0.5 or 0.9 results in a solid and significant improvement while most remaining Alpha values result in a significant decrease. Drawing conclusions purely from the considered experimental data, the SSUM-based TCN model seems to benefit from PIR, in contrast to the MLM-based model.

It is crucial to emphasize that, to the best of our knowledge, this is the first study applying PIR for refining ocean buoy forecasts based on modeled data. From the comprehensive review conducted, one particular paper by Daw et al. (2017) emerged, adopting a comparable methodology. This research focuses on incorporating physical equations into the loss function to enhance lake temperature forecasts. However, similar to the approach presented in this thesis, physics-informed data is incorporated into a machine learning model that serves as a baseline. In contrast to this work, only one physics-based feature is considered and its influence in the training process is not weighted. In alignment with the presented experiments, the paper points out that considering physics-based data increases accuracy. Consequently, the remaining insights and conclusions drawn are purely derived from the conducted tests and a deeper analysis is required before general statements can be made.

Key Finding and Guidance

It was the aim of this study to address the question whether the idea of physics-informed regularization by incorporating modelled data leads to an improvement of accuracy in ocean-buoy observation prediction. Based on the presented experiments, this question can be answered with 'yes'. However, even though the experiments proved that it is possible, it must be highlighted, that this is definitely not a universal exploration. While one setting resulted in an improvement of almost 40%, also many other settings showed enormous loss of accuracy. Using the MLM dataset, LSTM and GRU models showed minor improvements and TCN was not even capable to improve a single feature at any Alpha value. In contrast, CNN showed great results using the MLM dataset, especially when using low levels of physics-informed regularization. CNN also showed promising results with the SSUM dataset as well. However, for LSTM, GRU and CNN, choosing MLM over SSUM is highly suggested. Table 5.3 provides an overview

Model	MLM	SSUM
LSTM	Minor improvement	Not suggested
GRU	Minor improvement	Not suggested
CNN	Significant Improvement, especially with low PIR level	Significant Improvement
TCN	Not suggested	Not suggested

Table 5.3: Comparison and suggestion of investigated settings

showing suggestions and expected results. Those are solely based on results presented in section 5.2. While this table represents a general impression, the selection of a specific architecture, dataset representation, and PIR level should be made depending on the feature of interest and based on insights from the charts in section 5.2.

Contribution

Besides the implementation of a toolkit for general utilization for a wide range of related research endeavors, this thesis contributes to the fields of marine weather prediction and physics-incorporated machine learning by providing a study investigating physics-informed regularization.

Overall, the study, limited to 14 stations in the Gulf of Mexico and four specific models, demonstrates that the considered methodologies can lead to accuracy improvement in specific scenarios. It is hoped that the findings presented contribute to research on ocean-buoy forecasting by drawing attention to this alternative approach for physics-informed neural networks and motivating further research in this direction.

5.4 Summary

Chapter 5 showcased the use of the developed toolkit in conducting an experimental study on physics-informed regularization. This not only highlighted the toolkit's capabilities but also offered insights into PIR's performance and addressed the research question.

Initially, this chapter outlined the meticulous process of station selection. The primary objective was striking a balance between data quantity and quality. Out of 206 stations in the Gulf of Mexico, 14 were chosen based on criteria such as station ownership, type, and data availability. The final dataset comprises 235 NDBC data files and their corresponding ERA5 counterparts.

The chapter then elaborated on the test scenarios. Four unique deep learning models were utilized, with each permitting the adjustment of the ERA5 features' impact through the Alpha parameter. With one dataset using the MLM approach and another the SSUM approach, combined with 11 levels of physics-informed regularization, the total number of experiments reached 88. The process of dataset creation, test execution, and report generation was carried out using the toolkit highlighted in chapter 4.

All the error metrics produced were compiled into spreadsheets. Within these, MLM features from various stations were averaged, and charts were constructed to depict both the individual and overall improvements of features. Experiments with an Alpha value set to one acted as the benchmark as they displayed outcomes without the integration of physics-informed ERA5 data. Detailed discussions were carried out for each dataset-model pairing, drawing from the constructed tables and charts.

Delving deeper into the discussion, it emerged that the choice of dataset representation and architecture significantly influenced the results. Across all experiments, relying solely on modeled data to predict ocean buoy observations proved less than ideal. While TCN does not show any improvement when based on the MLM dataset, this approach should be favored for all other models. Significant enhancements were most evident in the CNN model.

The research question of whether physics-informed regularization can enhance marine weather prediction is affirmatively answered. However, although some experiments demonstrated significant improvements, others observed a decline in accuracy. The outcome of the study offers an initial insight and underscores the importance of continued research in this area.

Overall, this chapter underscored the toolkit's role while drawing attention to the significant potential of physics-informed regularization.

6 Lessons Learned

In this chapter, significant takeaways and insights gained throughout the research journey of this master's thesis are presented. Through the lens of reflection, both the hurdles surmounted, and the victories achieved in the literature research, development phase and during study execution are explored. Additional, personal insights and developments are shared.

6.1 Literature Research

In the initial research phase, a profound understanding of the techniques employed in weather forecasting emerged, along with insights into its historical development. The extensive dependence of various fields on accurate weather predictions also became evident. It is now apparent that weather forecasting is another example of an achievement that shapes our daily lives and owes its feasibility to developments in computer science.

While CNNs were predominantly known as architectures for image processing, the literature research revealed that they are also suitable and frequently employed for time series forecasting. The major challenge in this phase was finding work related to the implemented physics-informed regularization approach. This involved sifting through numerous papers that ultimately proved unsuitable. Only through significant time and effort, coupled with the assistance of colleagues from GulfSCEI, was this issue addressed to a certain extent. The takeaway is that not every concept is well-documented in the literature, presenting a valuable opportunity for contribution.

6.2 Development

At the beginning of the development phase, a critical decision was required: to tailor the implementation strictly for this study's objectives or to configure it for a more general applicability. The decision was inclined towards the latter approach. Whether it paid off will be determined by the subsequent utilization and adaptation of the software, an outcome that is anticipated

with considerable interest. While developing the toolkit, experience and knowledge in data science could be expanded. This particularly encompassed managing real-world data with missing values, crafting custom loss functions, and forecasting spatial-temporal time series.

6.3 Study

While designing the study, the challenge of appropriate data selection was faced and addressed by handpicking relevant data. An unexpected revelation during evaluation was the pronounced variability in the behavior of different architectures; a consistent trend based on the Alpha value, initially anticipated across all experiments, proved elusive. Moreover, analyzing the results led to the understanding that even the results from 88 experiments may not be suitable for broad generalization. Overall, conducting the study offered a deeper understanding of the behavior of neural network architectures and the impact of physics-informed regularization.

6.4 Personal

The journey of this master's thesis included the unique opportunity to spend five months in the United States. Within that time, professional growth occurred alongside personal development. By delving into an unfamiliar environment, forging connections with a myriad of people, and experiencing the vibrant culture of New Orleans, the understanding for global perspectives was deepened and skills in intercultural communication and adaptability were sharpened. Working on this master's thesis not only expanded the horizons of cross-cultural scientific collaboration but also ignited a newfound passion for research, a realm previously unconsidered for a career path. Furthermore, this journey provided a holistic understanding of the intricacies involved in high-quality scientific endeavors. Recognizing the significance of adhering to the established research process became evident: in future endeavors, there will be a greater emphasis on concluding the research phase before delving into development, meticulous phrasing of requirements, and ensuring a robust conceptual architecture. Such refinements, as gleaned from this experience, are anticipated to streamline and expedite subsequent research undertakings.

7 Conclusion and Future Work

This chapter concludes the key elements and characteristics of this project. Beyond this reflection on what has been accomplished, it also provides a forward-looking perspective, outlining potential avenues and directions for subsequent research and developments in the future.

7.1 Conclusion

On land as well as on the sea, current operative weather forecasts mainly conduct numerical weather prediction. However, a wide range of papers highlight the potential of deep learning in this domain.

The National Data Buoy Center (NDBC) provides a collection of historical records of ocean buoy measurements. This data is mainly collected with ocean buoys from NDBC but also third parties.

The aim of this thesis was to forecast those measurements. In the broader perspective, achievements in this domain further result in an improved marine weather prediction. A more accurate prediction of the weather on the sea can safeguide marines from extreme weather events and also protect coastal residents by providing timely warnings for meteorological calamities. Furthermore, this allows ship route optimization yielding both economic benefits and decreased emissions.

A common path to improve the forecast would have been to develop a physics-informed neural network and incorporate physical laws, mostly represented as partial differential equations into the loss function.

The European Center for Medium-Range Weather Forecasts (ECMWF) offers coherent worldwide historical weather data created through NWP model simulations and data assimilation. This dataset, called European Reanalysis 5 (ERA5) is the fifth and latest version and is based on the operative forecasting system used in 2016. As this data is generated following a theory-driven approach it is coherent with meteorological physical laws.

Therefore, the decision was made to investigate whether it is beneficial in terms of accuracy to incorporate this physics-informed data provided

by ECMWF into a deep learning model for forecasting NDBC ocean buoys measurements.

To answer this research question, first a toolkit for dataset creation and test execution was developed and further utilized for an extensive study. This toolkit is not tailored to a single use case, but is designed to be flexible, so that the toolkit can be used for a variety of similar studies.

The toolkit automatically sources the required data directly from NDBC and ERA5 and manages data storage in a smart way to keep data traffic and disk usage low.

It provides two scripts for dataset creation following two developed data representation strategies. First, Multi-Location Modelling (MLM) is based on assumption that representing a whole area instead of just a single position allows neural networks to detect additional patterns and lead to an increased forecast capability. Second, Station-Specific Unified Modelling aims to create a high number of instances by combining data from several stations in one dataset.

Furthermore, the toolkit is equipped with a test environment with the capability to train and test neural networks for time series forecasting using any dataset created with one of the described approaches. It is equipped with four predefined deep learning models but designed to be extensible to any neural network. Those predefined models are rather simple implementations of Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Convolutional Neural Network (CNN), and Temporal Convolutional Network (TCN) architectures. All of these are equipped with a custom loss function that allows regulating the impact of the physics-informed ERA5 data on the training process by changing a parameter called Alpha.

The experiment output can further be used as input for the report generator to create a detailed report as well as MAE and MSE performance metrics.

Following the toolkit development, it was utilized to explore the performance of the predefined deep learning models depending on the Alpha parameter. Considering both dataset creation approaches, four models and 11 Alpha levels, in total 88 experiments were executed.

The result allowed to answer the research question with 'yes', this approach is capable of improving the forecasting accuracy. However, while CNN showed significant improvement, especially when trained on the MLM dataset with low level of physics-informed regularization, many other experiments resulted in an accuracy decrease. It emerged that the choice of dataset representation and architecture significantly influenced the results.

Therefore, dataset representation, architecture but also the Alpha value should be selected based on the specific use case and the feature of interest.

Clearly, it can be said that there is a lot of potential in this approach, but it also requires further investigation. The developed toolkit can serve as a relevant tool for this while the presented study provides a general impression as the first investigation of utilizing ERA5 data for physics-informed regularization on NDBC ocean buoy forecasts.

7.2 Future Work

Providing a general overview of the capability of the presented approach, the conducted study sets the stage for in-depth future research stemming from its foundational insights. In various potential future projects, the presented toolkit can be utilized or further developed. Therefore, the source code is publicly available on GitHub[1]. While this section suggests a number of possible research paths, it is by no means an exhaustive list of all possibilities.

The study at hand does not provide grounds for broad generalizations. The absence of a clear pattern attributed solely to the Alpha value suggests further refinement is needed. A subsequent study, focusing on models that are fine-tuned in both architecture and hyperparameters and which incorporates a more elaborate preprocessing pipeline, might help in minimizing these variations.

Additionally, by broadening the scope and undertaking a detailed study spanning multiple geographic areas, the findings from this investigation can either be validated or challenged.

Working alongside meteorological experts can provide domain-specific interpretations and yield further insights. Additionally, this collaboration could facilitate a closer look into how the results align with established physical laws.

Lastly, the strategy of integrating real-world data with modelled counterparts combined with the presented regularization approach could also be applied in other domains. Potential applications of this approach include energy grid management, price forecasting, and traffic optimization.

When sticking to the domain of marine weather prediction, reaching out to the GulfSCEI lab in New Orleans may be beneficial. At this lab, Austin Schmidt, who co-supervised this thesis, is currently exploring a method that

[1] https://github.com/elsandner/Master_Project/

allows the Alpha value to self-adjust during training. To this end, he utilizes the Hybrid Coordinate Ocean Model (HYCOM, n.d.).

Bibliography

Agrawal, S., Barrington, L., Bromberg, C., Burge, J., Gazen, C., & Hickey, J. (2019). Machine learning for precipitation nowcasting from radar images. *arXiv preprint arXiv:1912.12132* (cit. on p. 25).

Ahmed, D. M., Hassan, M. M., & Mstafa, R. J. (2022). A review on deep sequential models for forecasting time series data. *Applied Computational Intelligence and Soft Computing*, *2022* (cit. on p. 51).

Aizenberg, I., Aizenberg, N. N., & Vandewalle, J. P. (2000). *Multi-valued and universal binary neurons: Theory, learning and applications*. Springer Science & Business Media. (Cit. on p. 12).

Ajit, A., Acharya, K., & Samanta, A. (2020). A review of convolutional neural networks. *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, 1–5. https://doi.org/10.1109/ic-ETITE47903.2020.049 (cit. on p. 20)

Anderson, D., Sheinbaum, J., & Haines, K. (1996). Data assimilation in ocean models. *Reports on Progress in Physics*, *59*(10), 1209 (cit. on p. 8).

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (cit. on pp. 23, 24, 53, 74).

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. *Proceedings of ICML workshop on unsupervised and transfer learning*, 37–49 (cit. on p. 25).

Baldi, P., Sadowski, P., & Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, *5*(1), 4308 (cit. on p. 12).

Bandara, K., Bergmeir, C., & Smyl, S. (2020). Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach. *Expert systems with applications*, *140*, 112896 (cit. on p. 17).

Bannister, R. (2017). A review of operational methods of variational and ensemble-variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, *143*(703), 607–633 (cit. on p. 8).

Bauer, P., Thorpe, A., & Brunet, G. (2015). The quiet revolution of numerical weather prediction. *Nature*, *525*(7567), 47–55 (cit. on pp. 10, 27).

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. *Proceedings of ICML workshop on unsupervised and transfer learning*, 17–36 (cit. on p. 12).

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166 (cit. on p. 15).

Berrisford, P., Guillory, A., Giusti, M., & Marsh, K. (2020). The family of ERA5 datasets [Last modified: Mai 05, 2023. Accessed on Mai 21, 2023]. %7Bhttps://confluence.ecmwf.int/x/icrLCg%7D. (Cit. on p. 34)

Beucler, T., Rasp, S., Pritchard, M. S., & Gentine, P. (2019). Achieving conservation of energy in neural network emulators for climate modeling. *ArXiv*, *abs/1906.06622* (cit. on p. 27).

Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., & Ghayvat, H. (2021). Cnn variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, *10*(20), 2470 (cit. on p. 20).

Bhimji, W., Farrell, S. A., Kurth, T., Paganini, M., Racah, E., et al. (2018). Deep neural networks for physics analysis on low-level whole-detector data at the lhc. *Journal of Physics: Conference Series*, *1085*(4), 042034 (cit. on p. 12).

Bjerknes, V. (1904). Das problem der wettervorhersage, betrachtet vom standpunkte der mechanik und der physik [Translated by Y. Mintz: The problem of weather forecasting as a problem in mechanics and physics. Los Angeles, 1954. Reprinted in Shapiro and Grønås (1999) 1–4.]. *Meteorologische Zeitschrift*, *21*(1), 1–7 (cit. on pp. 6, 8).

Bonavita, M., Hólm, E., Isaksen, L., & Fisher, M. (2016). The evolution of the ecmwf hybrid data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, *142*(694), 287–303 (cit. on p. 32).

Borovykh, A., Bohte, S., & Oosterlee, C. W. (2017). Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691* (cit. on p. 20).

Bouktif, S., Fiaz, A., Ouni, A., & Serhani, M. A. (2018). Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies*, *11*(7), 1636 (cit. on p. 17).

Brown, M. E. (2008). *Famine early warning systems and remote sensing data*. Springer. (Cit. on p. 10).

C3S CDS. (2021). Copernicus Climate Change Service (C3S) Climate Data Store (CDS) [Accessed on 20-May-2023]. https://cds.climate.copernicus.eu/cdsapp#!/dataset/10.24381/cds.67e8eeb7?tab=overview. (Cit. on p. 32)

C3S CDS. (2023). ERA5 hourly data on single levels from 1940 to present [Accessed on 20-May-2023]. https://doi.org/10.24381/cds.adbb2d47. (Cit. on pp. xxiii, 32–37)

Cai, M., Pipattanasomporn, M., & Rahman, S. (2019). Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. *Applied energy*, *236*, 1078–1088 (cit. on p. 22).

CALANCA, P., BOLIUS, D., WEIGEL, A. P., & LINIGER, M. A. (2011). Application of long-range weather forecasts to agricultural decision problems in europe. *The Journal of Agricultural Science*, *149*(1), 15–22. https://doi.org/10.1017/S0021859610000729 (cit. on p. 5)

Chan, J. C. L., & Kepert, J. D. (2010). *Global perspectives on tropical cyclones: From science to mitigation*. World Scientific. (Cit. on p. 9).

Charney, J. G., Fjörtoft, R., & Neumann, J. v. (1950). Numerical integration of the barotropic vorticity equation. *Tellus*, *2*(4), 237–254 (cit. on p. 6).

Chen, J.-H., Lin, S.-J., Magnusson, L., Bender, M., Chen, X., Zhou, L., Xiang, B., Rees, S., Morin, M., & Harris, L. (2019). Advancements in hurricane prediction with noaa's next-generation forecast system. *Geophysical Research Letters*, *46*(8), 4495–4501 (cit. on p. 9).

Chen, Y., Kang, Y., Chen, Y., & Wang, Z. (2020). Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, *399*, 491–501 (cit. on p. 24).

Chen, Z., Pang, M., Zhao, Z., Li, S., Miao, R., Zhang, Y., Feng, X., Feng, X., Zhang, Y., Duan, M., Huang, L., & Zhou, F. (2020). Feature selection may improve deep neural networks for the bioinformatics problems [Cited by: 45; All Open Access, Hybrid Gold Open Access]. *Bioinformatics*, *36*(5), 1542–1552. https://doi.org/10.1093/bioinformatics/btz763 (cit. on p. 12)

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (cit. on p. 18).

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (cit. on pp. 15, 20).

Cong, S., & Zhou, Y. (2023). A review of convolutional neural network architectures and their optimizations. *Artificial Intelligence Review*, *56*(3), 1905–1969 (cit. on pp. xviii, 22).

Constantin, P. (2006). Navier-stokes equations. In *The millennium prize problems* (pp. 75–96). American Mathematical Society. (Cit. on p. 8).

Courtier, P., Thépaut, J.-N., & Hollingsworth, A. (1994). A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, *120*(519), 1367–1387 (cit. on p. 33).

Cox, J. D. (2002). *Storm watchers: The turbulent history of weather prediction from franklin's kite to el niño*. John Wiley & Sons. (Cit. on p. 6).

Crippa, M., Guizzardi, D., Banja, M., Solazzo, E., Muntean, M., Schaaf, E., Pagani, F., Monforti-Ferrario, F., Olivier, J., Quadrelli, R., et al. (2022). Co2 emissions of all world countries. *Luxembourg: Publications Office of the European Union. doi*, *10*, 730164 (cit. on p. 40).

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., & Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, *92*(3), 88 (cit. on p. 45).

Daw, A., Karpatne, A., Watkins, W. D., Read, J. S., & Kumar, V. (2017). Physics-guided neural networks (pgnn): An application in lake temperature modeling. In *Knowledge-guided machine learning* (pp. 353–372). Chapman; Hall/CRC. (Cit. on pp. 27, 101).

Defense-Mapping-Agency. (1991). Department of defense world geodetic system 1984: Its definition and relationships with local geodetic systems. *Defense Technical Information Center* (cit. on p. 60).

Dreossi, T., Ghosh, S., Sangiovanni-Vincentelli, A., & Seshia, S. A. (2017). Systematic testing of convolutional neural networks for autonomous driving. *arXiv preprint arXiv:1708.03309* (cit. on p. 20).

ECMWF. (2023). Climate Data Store (CDS) documentation [Accessed: July 07, 2023, Last modified on Mai 23, 2023 15:32]. https://confluence.ecmwf.int/display/CKB/Climate+Data+Store+%28CDS%29+documentation. (Cit. on p. 61)

Farag, W., & Saleh, Z. (2018). Behavior cloning for autonomous driving using convolutional neural networks. *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 1–7 (cit. on p. 20).

Fefferman, C. L. (2000). Existence and smoothness of the navier-stokes equation. *The millennium prize problems*, *57*, 67 (cit. on p. 8).

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, *270*(2), 654–669 (cit. on pp. 17, 18).

Frisinger, H. H. (1972). Aristotle and his "meteorologica". *Bulletin of the American Meteorological Society*, *53*(7), 634–638 (cit. on p. 1).

Frnda, J., Durica, M., Nedoma, J., Zabka, S., Martinek, R., & Kostelansky, M. (2019). A weather forecast model accuracy analysis and ecmwf enhancement proposal by neural network. *Sensors*, *19*(23), 2087–2095. https://doi.org/10.3390/s19235144 (cit. on pp. 26, 27)

Fu, H., Liao, J., Ding, N., Duan, X., Gan, L., Liang, Y., Wang, X., Yang, J., Zheng, Y., Liu, W., et al. (2017). Redesigning cam-se for peta-scale climate modeling performance and ultra-high resolution on sunway

taihulight. *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 1–12 (cit. on p. 27).

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, *1*(2), 119–130 (cit. on p. 20).

Gamboa, J. C. B. (2017). Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887* (cit. on p. 13).

Gassmann, A. (2018). Discretization of generalized coriolis and friction terms on the deformed hexagonal c-grid. *Quarterly Journal of the Royal Meteorological Society*, *144*(716), 2038–2053 (cit. on p. 9).

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, *3*, 189–194 (cit. on p. 15).

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, *12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015 (cit. on p. 15)

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning* [http://www.deeplearningbook.org]. MIT Press. (Cit. on pp. xviii, 10).

Grover, A., Kapoor, A., & Horvitz, E. (2015). A deep hybrid model for weather forecasting, 379–386. https://doi.org/10.1145/2783258.2783275 (cit. on p. 26)

Haiden, T., Janousek, M., Vitart, F., Bouallègue, Z. B., Ferranti, L., Prates, F., & Richardson, D. (2018). *Evaluation of ecmwf forecasts, including the 2018 upgrade*. European Centre for Medium Range Weather Forecasts Reading, UK. (Cit. on p. 9).

Hall, C., & Jensen, R. E. (2021). Utilizing data from the noaa national data buoy center (cit. on p. 29).

Harper, K., Uccellini, L. W., Kalnay, E., Carey, K., & Morone, L. (2007). 50th anniversary of operational numerical weather prediction. *Bulletin of the American Meteorological Society*, *88*(5), 639–650 (cit. on p. 6).

Heaslip, E. (2023). *Here's how we can get weather forecasts while offshore* [Accessed: 2023-09-25]. https://www.sofarocean.com/posts/heres-how-we-can-get-weather-forecasts-while-offshore. (Cit. on p. 2)

Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Muñoz Sabater, J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., & Thépaut, J.-N. (2023). ERA5 hourly data on single levels from 1940 to present [Accessed data between February and July 2023]. https://doi.org/10.24381/cds.adbb2d47. (Cit. on p. 33)

Hersbach, H., Bell, B., Berrisford, P., Hirahara, S., Horányi, A., Muñoz-Sabater, J., Nicolas, J., Peubey, C., Radu, R., Schepers, D., et al. (2020).

The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 1999–2049 (cit. on pp. 32, 33).

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504–507 (cit. on p. 12).

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, *91*(1) (cit. on p. 15).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780 (cit. on p. 15).

Holton, J. R. (2004). *An introduction to dynamic meteorology, volume 1*. Academic Press. (Cit. on p. 9).

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, *148*(3), 574 (cit. on pp. 12, 20).

HYCOM. (n.d.). Hycom overview [Accessed: 27-09-2023]. https://www.hycom.org/hycom/overview. (Cit. on p. 109)

Isaksen, L., Bonavita, M., Buizza, R., Fisher, M., Haseler, J., Leutbecher, M., & Raynaud, L. (2010). Ensemble of data assimilations at ecmwf (cit. on p. 33).

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. *Proceedings of the 44th annual international symposium on computer architecture*, 1–12 (cit. on p. 27).

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., & Wu, Y. (2016). Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* (cit. on p. 20).

Kamyshanska, H., & Memisevic, R. (2015). The potential energy of an autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(6), 1261–1273. https://doi.org/10.1109/TPAMI.2014.2362140 (cit. on p. 25)

Kanamitsu, M. (1989). Description of the nmc global data assimilation and forecast system. *Weather and forecasting*, *4*(3), 335–342 (cit. on p. 8).

Kashinath, K., Mustafa, M., Albert, A., Wu, J., Jiang, C., Esmaeilzadeh, S., Azizzadenesheli, K., Wang, R., Chattopadhyay, A., Singh, A., et al. (2021). Physics-informed machine learning: Case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200093 (cit. on pp. 26, 27).

Keras. (2023). *Dropout layer* [Accessed: 2023-09-18]. https://keras.io/api/layers/regularization_layers/dropout/#dropout-class. (Cit. on p. 52)

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (cit. on p. 53).

Koprinska, I., Wu, D., & Wang, Z. (2018). Convolutional neural networks for energy time series forecasting. *2018 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (cit. on pp. 21, 22).

Kuan, L., Yan, Z., Xin, W., Yan, C., Xiangkun, P., Wenxue, S., Zhe, J., Yong, Z., Nan, X., & Xin, Z. (2017). Short-term electricity load forecasting method based on multilayered self-normalizing gru network. *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, 1–5 (cit. on p. 20).

Kuo, P.-H., & Huang, C.-J. (2018). A high precision artificial neural networks model for short-term energy load forecasting. *Energies*, *11*(1), 213 (cit. on p. 22).

Lara-Benıétez, P., Carranza-Garcıéa, M., Garcıéa-Gutiérrez, J., & Riquelme, J. C. (2020). Asynchronous dual-pipeline deep learning framework for online data stream classification. *Integrated Computer-Aided Engineering*, *27*(2), 101–119 (cit. on p. 13).

Lara-Benıétez, P., Carranza-Garcıéa, M., Luna-Romera, J. M., & Riquelme, J. C. (2020). Temporal convolutional networks applied to energy-related time series forecasting. *applied sciences*, *10*(7), 2322 (cit. on p. 24).

Lara-Benıétez, P., Carranza-Garcıéa, M., & Riquelme, J. C. (2021). An experimental review on deep learning architectures for time series forecasting. *International Journal of Neural Systems*, *31*(03), 2130001 (cit. on pp. 13, 14, 17, 19, 22).

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791 (cit. on p. 20)

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551 (cit. on p. 20).

Leslie, L., & Dietachmayer, G. (1992). Real-time limited area numerical weather prediction in australia- a historical perspective. *Australian Meteorological Magazine*. http://www.bom.gov.au/jshess/docs/1992/leslie2.pdf (cit. on p. 7)

Little, R. J., & Rubin, D. B. (2019). *Statistical analysis with missing data* (Vol. 793). John Wiley & Sons. (Cit. on p. 50).

Liu, H., Mi, X., & Li, Y. (2018). Smart deep learning based wind speed prediction model using wavelet packet decomposition, convolutional neural network and convolutional long short term memory network. *Energy Conversion and Management*, *166*, 120–131 (cit. on p. 22).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440 (cit. on p. 23).

Lynch, P. (2007). The origins of computer weather prediction and climate modeling. *Journal of Computational Physics*, *227*(7), 3431–3444. https://doi.org/10.1016/j.jcp.2007.02.029 (cit. on pp. 1, 5, 6)

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015a). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, *54*, 187–197 (cit. on p. 17).

Ma, X., Tao, Z., Wang, Y., Yu, H., & Wang, Y. (2015b). Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies*, *54*, 187–197. https://doi.org/https://doi.org/10.1016/j.trc.2015.03.014 (cit. on p. 18)

May, R., Soroka, D., Presnell, W., & Garcia, B. (2014). Marine weather forecasting in the national weather service (nws) [NOAA's National Weather Service, Silver Spring, MD. Port Meteorological Officer - South Florida. NOAA's National Weather Service, Eureka, CA. Accessed on September 27, 2023]. *Mariners Weather Log*, *58*(3). https://www.vos.noaa.gov/MWL/201412/forecasting.shtml#:~:text=Official%20three%2Dday%20marine%20weather,received%20on%20ship%20at%20sea. (cit. on p. 1)

Mazzarella, V., Milelli, M., Lagasio, M., Federico, S., Torcasio, R. C., Biondi, R., Realini, E., Llasat, M. C., Rigo, T., Esbrié, L., et al. (2022). Is an nwp-based nowcasting system suitable for aviation operations? *Remote Sensing*, *14*(18), 4440 (cit. on p. 5).

Mitchell, T. (1997). *Machine learning*. McGraw Hill. (Cit. on p. 11).

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, *73*, 1–15. https://doi.org/https://doi.org/10.1016/j.dsp.2017.10.011 (cit. on p. 28)

Morone, L., & Carmeyia Gillis, N. N. W. S. (2007). The history of numerical weather prediction [Accessed: March 31, 2023]. https://www.vos.noaa.gov/MWL/dec_07/weatherprediction.shtml. (Cit. on p. 6)

NDBC. (n.d.-a). How are spectral wave data derived from buoy motion measurements? (Cit. on p. 36).

NDBC. (n.d.-b). Measurement Descriptions and Units [Accessed May 22, 2023, n.d.]. (Cit. on pp. xxiii, 29, 30, 34–37).

NDBC. (n.d.-c). National Data Buoy Center [Accessed on May 22, 2023]. (Cit. on p. 29).

NDBC. (n.d.-d). What averaging procedures are performed on the wind measurements? [Accessed May 22, 2023, n.d.]. (Cit. on p. 35).

Nitta, T., & Saito, K. (2004). Early history of the operational numerical weather prediction in japan. *Symp. on the 50th Anniversary of Operational Numerical Weather Prediction* (cit. on p. 7).

NOAA. (2021). The first climate model [Accessed in February 2023]. https://celebrating200years.noaa.gov/breakthroughs/climate_model/welcome.html. (Cit. on p. 7)

Nor, K. M., Shaaban, M., & Abdul Rahman, H. (2014). Feasibility assessment of wind energy resources in malaysia based on nwp models. *Renewable Energy*, *62*, 147–154. https://doi.org/https://doi.org/10.1016/j.renene.2013.07.001 (cit. on p. 5)

Olah, C. (2015). Understanding lstm networks [Accessed on June 7, 2023]. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (cit. on pp. 14, 15, 17)

Ongsulee, P. (2017). Artificial intelligence, machine learning and deep learning. *2017 15th international conference on ICT and knowledge engineering (ICT&KE)*, 1–6 (cit. on pp. 11, 12).

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (cit. on p. 23).

Pan, C., Tan, J., Feng, D., & Li, Y. (2019). Very short-term solar generation forecasting based on lstm with temporal attention mechanism. *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, 267–271 (cit. on p. 17).

Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al. (2019). Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, *572*(7767), 745–754 (cit. on p. 27).

Penny, S. G., Akella, S., Buehner, M., Chevallier, M., Counillon, F., Draper, C., Frolov, S., Fujii, Y., Karspeck, A., & Kumar, A. (2017). *Coupled data assimilation for integrated earth system analysis and prediction: Goals, challenges, and recommendations* (tech. rep.). (Cit. on p. 33).

Pielke, R. A. (2002). *Mesoscale meteorological modeling*. Academic Press. (Cit. on p. 8).

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045 (cit. on p. 27)

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, *378*, 686–707 (cit. on p. 45).

Ravanelli, M., Brakel, P., Omologo, M., & Bengio, Y. (2018). Light gated recurrent units for speech recognition. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*(2), 92–102 (cit. on p. 18).

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., & Carvalhais, N. (2019). Deep learning and process understanding for data-driven earth system science. *Nature*, *566*(7743), 195–204 (cit. on p. 27).

Remy, P. (2020). Temporal convolutional networks for keras. (Cit. on p. 53).

Ren, X., Li, X., Ren, K., Song, J., Xu, Z., Deng, K., & Wang, X. (2021). Deep learning-based weather prediction: A survey. *Big Data Research*, *23*, 100178. https://doi.org/https://doi.org/10.1016/j.bdr.2020.100178 (cit. on pp. xviii, 27, 28)

Ren, X., Zhao, J., Li, X., Ren, K., Song, J., & Sun, D. (2019). Pagcm: A scalable parallel spectral-based atmospheric general circulation model. *Concurrency and Computation: Practice and Experience*, *31*(20), e5290 (cit. on p. 27).

Rew, R., & Davis, G. (1990). Netcdf: An interface for scientific data access. *IEEE Computer Graphics and Applications*, *10*(4), 76–82. https://doi.org/10.1109/38.56302 (cit. on p. 63)

Reyes, O., & Ventura, S. (2019). Performing multi-target regression via a parameter sharing-based deep network. *International journal of neural systems*, *29*(09), 1950014 (cit. on p. 13).

Richardson, L. F. (1922). *Weather prediction by numerical process*. University Press. (Cit. on p. 6).

Ringler, T. D., Thuburn, J., Klemp, J. B., & Skamarock, W. C. (2010). A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured c-grids. *Journal of Computational Physics*, *229*(9), 3065–3090 (cit. on p. 9).

Runge, J., Petoukhov, V., Donges, J. F., Hlinka, J., Jajcay, N., Vejmelka, M., Hartman, D., Marwan, N., Paluš, M., & Kurths, J. (2015). Identifying causal gateways and mediators in complex spatio-temporal systems. *Nature communications*, *6*(1), 8502 (cit. on p. 26).

Sagheer, A., & Kotb, M. (2019). Time series forecasting of petroleum production using deep lstm recurrent networks. *Neurocomputing*, *323*, 203–213 (cit. on pp. 17, 18).

Satoh, M., Tomita, H., Yashiro, H., Miura, H., Kodama, C., Seiki, T., Noda, A. T., Yamada, Y., Goto, D., Sawada, M., et al. (2014). The non-hydrostatic icosahedral atmospheric model: Description and development. *Progress in Earth and Planetary Science*, *1*(1), 1–32 (cit. on p. 9).

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, *61*, 85–117 (cit. on p. 13).

Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., Mozaffari, A., & Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200097 (cit. on pp. 1, 7, 9, 24).

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., & Tkatchenko, A. (2017). Quantum-chemical insights from deep tensor neural networks. *Nature communications*, *8*(1), 1–8 (cit. on p. 12).

Sene, K. (2008). *Flood warning, forecasting and emergency response*. Springer Science & Business Media. (Cit. on p. 5).

Shen, Z., Zhang, Y., Lu, J., Xu, J., & Xiao, G. (2020). A novel time series forecasting model with deep learning. *Neurocomputing*, *396*, 302–313 (cit. on p. 22).

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, *28*, 802–810 (cit. on pp. 25, 26).

Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W.-k., & Woo, W.-c. (2017). Deep learning for precipitation nowcasting: A benchmark and a new model. *Advances in neural information processing systems*, *30* (cit. on p. 25).

Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, 1–6 (cit. on p. 11).

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*(1), 75–85 (cit. on p. 17).

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., & Khudanpur, S. (2018). X-vectors: Robust dnn embeddings for speaker recognition. *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 5329–5333 (cit. on p. 12).

Sommerville, I. (2011). *Software engineering*. Pearson. https://books.google.at/books?id=loegcQAACAAJ. (Cit. on p. 41)

Sønderby, C. K., Espeholt, L., Heek, J., Dehghani, M., Oliver, A., Salimans, T., Agrawal, S., Hickey, J., & Kalchbrenner, N. (2020). Metnet: A neural weather model for precipitation forecasting. *arXiv preprint arXiv:2003.12140* (cit. on pp. 25, 27).

Srivastava, N., Mansimov, E., & Salakhudinov, R. (2015). Unsupervised learning of video representations using lstms. *International conference on machine learning*, 843–852 (cit. on p. 26).

Steele, K. E., & Mettlach, T. (1993). Ndbc wave data—current and planned. *Ocean Wave Measurement and Analysis*, 198–207 (cit. on p. 36).

Steffen, T. (n.d.). Numerical weather prediction (weather models) [Accessed on May 25, 2023]. *NOAA NWS Juneau*. (Cit. on p. 5).

Strikwerda, J. C. (2004). *Finite difference schemes and partial differential equations*. SIAM. (Cit. on p. 8).

Sun, Y., Shi, Z., Zhang, J., Qi, Y., Hu, H., & Shen, Z. M. (2022). Improving accuracy without losing interpretability: A ml approach for time series forecasting. *arXiv preprint arXiv:2212.06620* (cit. on p. 24).

TensorFlow. (2023). Tensorflow [Last updated 2023-05-26 UTC]. (Cit. on p. 54).

Tian, C., Ma, J., Zhang, C., & Zhan, P. (2018). A deep neural network model for short-term load forecast based on long short-term memory network and convolutional neural network. *Energies*, *11*(12), 3493 (cit. on pp. 20, 22).

Tian, Y., & Pan, L. (2015). Predicting short-term traffic flow by long short-term memory recurrent neural network. *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*, 153–158 (cit. on p. 17).

Tomita, H., & Satoh, M. (2004). A new dynamical framework of nonhydrostatic global model using the icosahedral grid. *Fluid Dynamics Research*, *34*(6), 357–400 (cit. on p. 8).

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Forecasting stock prices from the limit order book using convolutional neural networks. *2017 IEEE 19th conference on business informatics (CBI)*, *1*, 7–12 (cit. on p. 22).

Ugurlu, U., Oksuz, I., & Tas, O. (2018). Electricity price forecasting using recurrent neural networks. *Energies*, *11*(5), 1255 (cit. on pp. 19, 20).

Ullrich, P. A., & Jablonowski, C. (2012). Mcore: A non-hydrostatic atmospheric dynamical core utilizing high-order finite-volume methods. *Journal of Computational Physics*, *231*(15), 5078–5108 (cit. on pp. 8, 9).

Wan, R., Mei, S., Wang, J., Liu, M., & Yang, F. (2019). Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics*, *8*(8), 876 (cit. on p. 24).

Wang, Y., Liao, W., & Chang, Y. (2018). Gated recurrent unit network-based short-term photovoltaic forecasting. *Energies*, *11*(8), 2163 (cit. on p. 20).

Widén, J., Carpman, N., Castellucci, V., Lingfors, D., Olauson, J., Remouit, F., Bergkvist, M., Grabbe, M., & Waters, R. (2015). Variability assessment and forecasting of renewables: A review for solar, wind, wave and tidal resources. *Renewable and Sustainable Energy Reviews*, *44*, 356–375 (cit. on p. 40).

Wood, N., Staniforth, A., White, A., Allen, T., Diamantakis, M., Gross, M., Melvin, T., Smith, C., Vosper, S., Zerroukat, M., et al. (2014). An inherently mass-conserving semi-implicit semi-lagrangian discretization of the deep-atmosphere global non-hydrostatic equations. *Quarterly Journal of the Royal Meteorological Society*, *140*(682), 1505–1520 (cit. on p. 9).

Xue, W., Yang, C., Fu, H., Wang, X., Xu, Y., Gan, L., Lu, Y., & Zhu, X. Enabling and scaling a global shallow-water atmospheric model

on tianhe-2 [Cited by: 45]. In: Cited by: 45. 2014, 745–754. https://doi.org/10.1109/IPDPS.2014.82 (cit. on p. 27).

Zargar, S. (2021). Introduction to sequence learning models: Rnn, lstm, gru. *no. April* (cit. on pp. 17, 19).

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE geoscience and remote sensing magazine*, *5*(4), 8–36 (cit. on p. 25).

Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., & Perdikaris, P. (2019). Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, *394*, 56–81. https://doi.org/https://doi.org/10.1016/j.jcp.2019.05.024 (cit. on p. 27)

Zis, T. P., Psaraftis, H. N., & Ding, L. (2020). Ship weather routing: A taxonomy and survey. *Ocean Engineering*, *213*, 107697 (cit. on p. 40).