Master of Science in Computer Security
2024-10-04

Comparing Deep Learning Models for Lightning Strike Prediction in a Changing Climate

An Empirical Study

NAME REDACTED

Faculty of Computing. Blekinge Institute of Technology. 371 79 Karlskrona, Sweden.

Contact Information:

Author:
NAME REDACTED
aiai18@student.bth.se

University advisor:
Senior lecturer NAME REDACTED
Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet	:	www.bth.se
Phone	:	+46 455 38 50 00
Fax	:	+46 455 38 50 57

Abstract

The accurate prediction of lightning strikes is crucial for disaster preparedness, risk mitigation, and resource allocation. Traditional numerical weather simulations, while effective, are computationally intensive and complex. Recent advancements in deep learning offer a promising alternative for more efficient and precise weather forecasting.

This thesis aims to evaluate the effectiveness of various deep learning models in predicting lightning strikes. The study focuses on identifying the optimal model architectures, time frames, features, and hyperparameters to enhance prediction accuracy and lead-time.

The research utilizes datasets from the Swedish Meteorological and Hydrological Institute, including the Lightning Archive and MESAN (AROME) datasets. Extensive pre-processing techniques such as filtering, balancing, binning, extraction, and imputation are applied. Four deep learning models– Dense Neural Networks (DNN), Simple Recurrent Neural Networks (SRNN), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) are evaluated using stratified k-fold cross-validation. Metrics such as Training Time, F1 Score, Wilson Score, and Mean Absolute Error are used for model comparison. Hyperparameters are optimized using a genetic algorithm.

The study demonstrates that deep learning models can accurately predict lightning strikes with overall accuracies exceeding 75%. The LSTM and GRU models show higher performance for shorter time frames, while the DNN and SRNN models exhibit more stable and consistent performance across various time frames. The GRU model outperforms the LSTM model in all metrics, and the DNN model is the fastest in terms of training time.

Deep learning models offer a viable and efficient alternative to traditional numerical weather simulations for lightning strike prediction. The findings highlight the importance of selecting appropriate lookback and lookahead values, as well as the need for high-quality, diverse datasets. Future work should focus on further fine-tuning the models, incorporating more data, and evaluating their performance in real-world scenarios.

Keywords: Deep Learning, Lightning Prediction, Hyperparameter Tuning, Data Analysis, Model Comparison.

Sammanfattning

Den exakta förutsägelsen av blixtnedslag är avgörande för katastrofberedskap, riskreducering och resursallokering. Traditionella numeriska vädersimuleringar är, även om de är effektiva, beräkningsintensiva och komplexa. De senaste framstegen inom djupinlärning erbjuder ett lovande alternativ för mer effektiv och exakt väderprognos.

Denna avhandling syftar till att utvärdera effektiviteten hos olika modeller för djupinlärning för att förutsäga blixtnedslag. Studien fokuserar på att identifiera de optimala modellarkitekturerna, tidsramar, ingångsparametrar och hyperparametrar för att förbättra prediktionsnoggrannheten och ledtiden.

Forskningen använder datauppsättningar från Sveriges meteorologiska och hydrologiska institut, inklusive datauppsättningarna Lightning Archive och MESAN (AROME). Omfattande förbearbetningstekniker som filtrering, balansering, binning, extraktion och imputering tillämpas. Fyra modeller för djupinlärning– Dense Neural Network (DNN), Simple Recurrent Neural Network (SRNN), Long Short Term Memory (LSTM) och Gated Recurrent Unit (GRU) utvärderas med hjälp av stratifierad k-faldig korsvalidering. Mätvärden som träningstid, F1-poäng, Wilson-poäng och genomsnittligt absolut fel används för modelljämförelse. Hyperparametrar optimeras med hjälp av en genetisk algoritm.

Studien visar att djupinlärningsmodeller kan förutsäga blixtnedslag exakt med en total noggrannhet som överstiger 75%. LSTM- och GRU-modellerna visar högre prestanda för kortare tidsramar, medan DNN- och SRNN-modellerna uppvisar mer stabil och konsekvent prestanda över olika tidsramar. GRU-modellen överträffar LSTM-modellen i alla mätvärden, och DNN-modellen är snabbast när det gäller träningstid.

Modeller för djupinlärning erbjuder ett hållbart och effektivt alternativ till traditionella numeriska vädersimuleringar för att förutsäga blixtnedslag. Resultaten understryker vikten av att välja lämpliga tillbakablicks- och framtidsvärden, såväl som behovet av högkvalitativa, olika datauppsättningar. Framtida arbete bör fokusera på att ytterligare finjustera modellerna, införliva mer data och utvärdera deras prestanda i verkliga scenarier.

Sökord: Djupinlärning, Blixförutsägning, Hyperparameteroptimisering, Dataanalys, Modelljämförelse.

Acknowledgments

This study was conducted in collaboration with *Zenon*, a consulting company located in Linköping, Sweden, providing products in the line of lightning and storm prediction. The author would like to express gratitude to *Zenon* for providing the necessary hardware and research environment, as well as valuable expertise in the field of weather and climate prediction.

Additionally, the author would like to extend their sincere appreciation to Senior lecturer NAME REDACTED, who supervised the project and provided valuable guidance and improvement suggestions. His expertise and insights greatly contributed to the success of this study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The analysis of storms and lightning patterns holds immense significance for society, as it impacts multiple important sectors including infrastructure, agriculture, transportation, and public safety. Accurate prediction of lightning strikes can greatly contribute to disaster preparedness, risk mitigation, and resource allocation. Historically, meteorologists have heavily relied on numerical models and simulations to make forecasts. These methods require massive computational resources. They are centralized and extremely complex, operating a global network of sensors causing communication overhead. However, the emergence of artificial intelligence (AI), machine learning (ML), and specifically deep learning (DL) has opened up new avenues for more precise, efficient, and effective prediction methods.

In recent years, the field of deep learning has witnessed remarkable advancements, revolutionizing various domains, including weather prediction. Deep learning models, inspired by the structure and functioning of the human brain, have demonstrated exceptional capabilities in processing and analyzing complex data patterns. By leveraging large datasets and powerful computational resources, deep learning models can automatically learn intricate relationships and patterns from the data, making them suitable for accurate meteorological predictions.

One of the key advantages of deep learning models is their ability to extract high-level patterns from raw data, eliminating the need for manual development. This pattern extraction process allows the models to capture subtle and non-linear relationships between different meteorological variables, which are often challenging to identify using traditional numerical models. Consequently, deep learning models have the potential to enhance the accuracy and reliability of lightning prediction.

1.1.1 Significance

Accurate prediction of storms and lightning strikes holds great significance due to its wide-ranging impact on various stakeholders. Severe storms can cause significant damage to communities, resulting in property damage and loss of life.

Early Warnings

Early notification of approaching storms and lightning strikes can be life-saving for governments, emergency responders and individuals. This advance warning provides a valuable opportunity to take proactive measures, such as initiating evacuations, reinforcing infrastructure, and mobilizing resources.

Furthermore, accurate storm and lightning prediction can also benefit various industries. For example, the aviation industry heavily relies on weather forecasts to ensure the safety of flights and mitigate disruptions caused by severe weather conditions. Accurate prediction models can enable airlines to make informed decisions regarding flight routes and schedules, reducing the risk of accidents and delays.

Efficient Resource Allocation

Many governments, companies, and organizations rely on weather forecasts to plan and allocate resources effectively. Access to improved and more accurate predictions enables better decision-making regarding the timing, method, and location of resource allocation.

This can not only reduce cost but also save lives. Lightning strikes can damage power infrastructure, disrupt communication networks, and pose a threat to individuals engaged in outdoor activities. By accurately forecasting lightning strikes, these industries can take proactive measures and develop better strategies that minimizes the affected infrastructure to ensure the safety of their employees and customers.

Environmental Impact

Accurate storm and lightning forecasts can have a severe impact on environmental preservation and sustainability. By understanding storm behavior and patterns, it is possible to implement measures to manage potential environmental impacts such as:

Stormwater runoff which can lead to flooding and damage infrastructure. Accurate predictions allows for preparation and implementation of strategies to mitigate runoff, such as proper drainage systems and green infrastructure.

Soil erosion which is when soil is moved or displaced. Predicting storm behavior and patterns can help implement erosion control measures like contour plowing and cover crops.

Lightning-induced wildfires which is a severe problem in geographically hot and dry locations. Through predicting thunderstorms it is possible to implement fire management strategies to minimize the impact on biodiversity.

Contribution to the Field

This research also contributes to the broader field of meteorology by evaluating cutting-edge artificial intelligence and machine learning techniques. It demonstrates how these technologies can greatly enhance our understanding of weather patterns and climate change. By improving storm and lightning prediction models, this research makes the way for further advancements in the field, leading to more accurate and reliable forecasts. This, in turn, can help mitigate the impact of severe weather events and improve overall preparedness and response strategies.

1.1.2 Current Methods for Lightning Forecasting

At the time of writing, the vast majority of weather forecasting operations uses numerical methods [1] for weather forecasting. In these methods a set of non-linear equations are calculated to simulate the global environmental system in entirety. From a theoretical standpoint this is the optimal approach, utilizing established rules of physics to accurately predict weather phenomena. In practice however, there are many factors that limits the efficiency of such methods. Small variations in the input data has a drastic effect on the resulting outcome [2]. An incorrect measurement, a temperature fluctuation not picked up by sensors or differences in sensor hardware can render predictions highly unstable, especially in the long term. The data collection and simulation also has to be performed on a globally large scale, requiring massive computational resources and a comprehensive network of distributed sensor arrays to provide useful results in real-time.

Because of these restrictions artificial intelligence and machine learning models are seen as promising alternatives. Their ability to automatically extract patterns from complex data makes them useful in scenarios where correlation between features and labels is obvious but hard to identify [2].

1.2 Scope

The scope of this thesis is centered on the application of deep learning models for the prediction of lightning strikes, with a particular focus on the Swedish region. The study aims to explore the potential of various deep learning architectures, including Dense Neural Networks (DNN), Simple Recurrent Neural Networks (SRNN), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) models, in accurately forecasting lightning strikes based on meteorological data. The datasets utilized in this research are provided by the Swedish Meteorological and Hydrological Institute (SMHI), specifically the Lightning Archive dataset and the MESAN (AROME) dataset.

The scope is deliberately narrowed to ensure a comprehensive and in-depth analysis of the problem. By focusing on a specific geographic region, a defined set of deep learning models and a specific set of data the study aims to provide detailed insights into the effectiveness of these models in lightning strike prediction. This approach allows for a thorough examination of the pre-processing techniques, model configurations, and hyperparameter tuning required to optimize the performance of the models.

The following research questions will guide this study in order to achieve the objective:

1. How do the selected models perform across different time frames?
2. Which selected models are the most effective for predicting the occurrence of lightning strikes, and how do they compare to each other?
3. What are the optimal features for predicting lightning strikes?
4. What are the optimal hyperparameters and model configurations for lightning strike prediction?

These questions are designed to clarify the scope by providing a clear framework for the research. The study aims to fill the research gap identified in the field of lightning strike prediction using deep learning models, which has not been extensively explored compared to other weather prediction tasks.

1.3 Outline

This thesis consists of six chapters that address different aspects of deep learning models for lightning strike prediction. The chapters are as follows:

1. Introduction: The current chapter provides background information on the importance of accurate lightning strike prediction and the potential of deep learning in weather forecasting. It outlines the research questions and scope of the study.

2. Related Work: Reviews existing literature on deep learning models for weather prediction, with a focus on lightning strike prediction. Discusses different model types, challenges, and previous studies. Covers feature selection and hyperparameter tuning techniques.

3. Methodology: Details the research design, including dataset selection, preprocessing methods, and model development. Describes the hyperparameter tuning process and evaluation metrics used.

4. Results and Analysis: Presents the results of the study, including an analysis of the LIGHT and MESAN datasets. Evaluates model performance across different time frames and discusses the impact of lookback and lookahead values.

5. Discussion: Interprets the results in the context of the research questions and the field of lightning prediction. Compares the performance of different model types and addresses challenges and limitations. Explores implications for future research and applications.

6. Conclusions and Future Work: Summarizes the key findings and contributions of the study. Provides recommendations for future work, including dataset selection, model architecture, and hyperparameter tuning. Reflects on the overall impact and significance of the study.

Chapter 2

Related Work

2.1 Deep learning for lightning strike prediction

Hewage, Behera, Trovati, *et al.* [3] explores the prediction of lightning strikes using deep learning (DL) models. To gather the necessary data, the author sets up custom weather stations that collect common weather parameters such as air pressure, temperature, and wind speed.

The study employs two types of DL models: Long Short-Term Memory (LSTM) and Temporal Convolutional Neural (TCN) models. LSTM models have an advantage over simple recurrent neural networks (RNNs) as they overcome the vanishing gradient problem. This problem arises when the importance of past time steps diminishes over time, resulting in older time points having minimal impact on predictions [4]. LSTM layers address this issue by incorporating forget, input, and output gates, which determine the relevance of information for future predictions. The TCN model is an improvement over the LSTM model, combining convolutional and recurrent capabilities to enable simpler autoregressive modeling and the utilization of longer memory sequences.

Before training the models, the author applies several pre-processing techniques to prepare the original observations. The data is collected over an 8-month period at 15-minute intervals. To handle missing values, linear interpolation is used for data imputation. Additionally, all parameters are scaled to a range between 0 and 1. To extract training labels, a sliding-window procedure is employed, using the last 7 days of data as the feature sequence and predicting the label two hours ahead.

To evaluate the model's performance, the author compares the Multiple-Input Multiple-Output (MIMO) and Multiple-Input Single-Output (MISO) metrics. The resulting model demonstrates accurate predictions for 10 different parameters while maintaining computational efficiency, enabling it to be run on stand-alone PCs.

In a separate study Sulaiman, Mohamed, and Mustaffa [5] explores the use of deep learning techniques, specifically Feed-Forward Neural Networks (FFNN), to predict lightning occurrences within a 100 km radius from University Malaysia Pahang Al-Sultan Abdullah (UMPSA) in Pekan, Pahang, Malaysia. The study leverages data recorded by the Malaysia Meteorology Department (MET Malaysia) and aims to develop a model that can predict the latitude and longitude of lightning occurrences using historical data.

The article provides a comprehensive review of previous work in lightning prediction and compares different machine learning and deep learning approaches. The methodology section explains the use of FFNN, activation functions, optimization techniques, and data preparation. The results and discussion section evaluates the performance of the FFNN model using various metrics and compares it with an LSTM model. The article also includes detailed simulation results and visualizations to demonstrate the effectiveness of the FFNN model.

The study concludes that the FFNN model is effective in predicting lightning occurrences and suggests future work to improve the model's accuracy. The article's key contributions include a novel application of FFNN for lightning prediction, a comprehensive evaluation of the model's performance, and practical implications for improving lightning prediction accuracy.

In a similar study Essa, Hunt, and Ajoodha [6] made an attempt to use machine learning techniques for short-term prediction of lightning strikes. They collected a comprehensive dataset of 20,000,000 lightning strikes, covering the entire country of South Africa. The dataset was pre-processed using a binning technique with a temporal bin size of 3 hours, and the parameters were scaled to fall in between 0 and 1. The dataset was divided into a training set and a testing set, with a ratio of 70% for the training and 30% for the testing data. Three different machine learning models were evaluated and compared: an AutoRegressive (AR) model, an AutoRegressive Integrated Moving Average (ARIMA) model, and a Long Short Term Memory (LSTM) model. Among these models, only the LSTM model is a deep learning model. The results demonstrated that the LSTM model considerably outperformed AR and ARIMA in terms of Mean Absolute Percentage Error (MAPE), although all models showed high error rates.

2.2 Model Types

Schultz, Betancourt, Gong, *et al.* [7] and Wang, Hu, Wu, *et al.* [8] provide a comprehensive overview of the current state of deep learning in weather prediction and compare the results with traditional numerical methods. One particularly successful type of DL model is the convolutional neural network (CNN) model [9]. CNNs are designed to process two-dimensional data, such as satellite imagery or weather model output, by applying a series of filters to extract increasingly complex patterns. By learning higher-level spatial behavior from the input data, CNNs can effectively predict weather phenomena.

Another type of DL model that has been explored is the recurrent neural network (RNN) [10] [11] [12]. RNNs are essentially dense neural networks with additional functionality built into the layers, allowing them to analyze input data in temporal sequences. Unlike CNNs, RNNs can take the dimension of time into account. They not only extract patterns based on the provided feature sequence but also learn from the changes in the data over time. This might make RNNs particularly suitable for capturing the dynamics of weather patterns.

In addition to CNNs and RNNs, more advanced model types such as variational auto-encoders (VAEs) and generative adversarial networks (GANs) have also been attempted [13] [14]. VAEs encode and decode the input data in specific ways to extract the important features for the task at hand. GANs, on the other hand, use two competitive neural networks: one that generates predictions and another that learns to differentiate between artificially generated predictions and true predictions. By leveraging the interplay between these two networks, GANs can improve the accuracy of the predictions.

Essa, Ajoodha, and Hunt [15] also explores the use of LSTM recurrent neural networks to predict short-term lightning flash densities in Southern Africa. The study focuses on two specific areas with different lightning flash densities. The authors evaluated the prediction ability of the LSTM model using historical data from the South African Lightning Detection Network. The model was trained using four years of data and predictions were made for one-year intervals. The results showed good correlation between predicted and actual lightning flashes, but the model tended to under-predict in one area. The model also showed good repeatability and consistent mean absolute error values. However, it did not predict about 60% of major lightning events. The study suggests combining the LSTM model with a weather data model to improve accuracy.

Bao, Zhang, Ma, *et al.* [16] presents a deep learning-based lightning prediction system that utilizes atmospheric electric field (EF) observations to predict the occurrence and location of lightning events. The study collected data from EF measurement stations and lightning locators in Guangzhou city. The data were clustered into one-minute intervals, and thunderstorm and non-thunderstorm samples were created for analysis.

To extract features from the EF time series data, a Sparse Autoencoder (SAE) was employed, which compresses the dimensionality of the input data while retaining essential information. The time positioning module utilized an improved ResNet50 model to classify weather samples and predict the occurrence of lightning. The model achieved an accuracy of 88.2%, with a precision of 92.2%, recall of 81.5%, and F1-score of 86.4%. The length of the EF time series was found to impact the model's accuracy, with a 60-minute time series yielding the best results.

Data augmentation techniques were applied to increase the training data and improve the model's performance. The improved ResNet50 model outperformed other models, such as CNN and LSTM, in terms of precision, recall, F1-score, and accuracy. The model also demonstrated robustness against noise interference, maintaining high accuracy even under noisy conditions.

For spatial positioning, an MLP model was used to predict the location of lightning occurrences based on EF data. The model achieved satisfactory results, accurately predicting the location of lightning flashes with most errors within acceptable limits.

The study evaluated the effect of different lengths of EF time series on the accuracy of lightning spatial localization and found that the length had little impact on accuracy. Additionally, real-world case studies were presented to demonstrate the model's performance in practical scenarios, showcasing its ability to accurately predict lightning occurrences and provide valuable information for lightning protection measures.

2.3 Feature Selection

Verikas and Bacauskiene [17] introduces a novel approach for feature selection in classification tasks using neural networks. The method involves training neural networks with an augmented cross-entropy error function to improve generalization and robustness. The authors propose a feature selection procedure that ranks features based on their impact on classification error and selects the subset with the highest accuracy. The proposed method is compared to five other techniques and consistently outperforms them in terms of classification accuracy. The five other feature selection techniques were Neural-Network Feature Selector (NNFS), Signal-to-Noise Ratio (SNR) Based Technique, Neural Network Output Sensitivity Based Feature Ranking, Fuzzy Entropy Based Feature Ranking, and Discriminant Analysis (DA) Based Feature Ranking.

Experimental investigations on artificial and real-world datasets demonstrate the effectiveness and robustness of the approach. The selected feature subsets also yield the best performance when tested with the k-Nearest Neighbors classifier.

Howley, Madden, O'Connell, *et al.* [18] provides a potential method for finding the optimal features. The authors evaluate the use of Principal Component Analysis (PCA) to improve the performance of machine learning methods in classifying data of higher dimensions, such as Raman spectra used for identifying narcotics. The presence of redundant or highly correlated attributes in such data can degrade classification accuracy and model performance.

The experiments show that PCA can improve the performance of machine learning in classifying high dimensional data. The study also compares the performance of five well-known machine learning techniques (Support Vector Machines, k-Nearest Neighbors, C4.5 Decision Tree, RIPPER and Naive Bayes) along with classification by Linear Regression on a Raman spectral dataset.

The results show that Support Vector Machines perform better than other methods on raw and normalized data, while pre-processing techniques such as normalization and first derivative can improve the classification accuracy of these methods. The use of PCA in combination with machine learning methods also improves performance, with most methods requiring no more than six principal components to achieve the lowest error. This method could aid in the identification and selection of the optimal parameters for lightning prediction.

However, the study also finds that the performance of rule-based learners, such as C4.5 and RIPPER, can be adversely affected by the use of PCA. The paper concludes that the use of NIPALS PCA in combination with machine learning appears to be a promising approach for the classification of high dimensional spectral data.

2.4 Hyperparameter Tuning

Choosing the appropriate hyperparameters for a model is a common problem in machine learning. A model typically consists of two types of parameters: internal and external hyperparameters. The internal parameters are automatically optimized during the training process. On the other hand, the external hyperparameters are part of the model and cannot be changed once the model has been built.

To find the optimal hyperparameters, different strategies and methods have been proposed in literature. Alibrahim and Ludwig [19] presents an extensive overview and comparison of the most common strategies and methods for finding the optimal hyperparameters are provided. The described strategies include random search, grid search, Bayesian search and genetic algorithms, however only the latter three are evaluated and compared. A brief overview of the strategies follows:

Random Search: A simple strategy for hyperparameter tuning where hyperparameters are randomly chosen from a predefined set. The model is trained and evaluated using the selected hyperparameters, and this process is repeated for a specific number of iterations or until a sufficient performance is achieved. Random search is computationally efficient but may not always find the optimal set of hyperparameters due to its random nature.

Grid Search: A systematic approach to hyperparameter tuning where a predefined set of hyperparameters is specified, and the model is trained and evaluated for each combination of hyperparameters in the grid. Grid search fully explores all possible hyperparameter combinations, making it guaranteed to find the optimal combination.

However, it can be extremely computationally expensive, especially with a large number of hyperparameters.

Bayesian Optimization: A sequential optimization strategy that uses a probabilistic model to guide the search for optimal hyperparameters. It combines prior knowledge and observed performance to update a probability distribution over the hyperparameters. Bayesian optimization can handle noisy and expensive-to-evaluate objective functions and adapts its search based on observed performance. However, it may require more iterations to converge compared to other strategies.

Genetic Algorithms: Genetic algorithms generate a population of potential candidates (combinations of hyperparameters). These candidates are evaluated, and the best-performing ones produce offspring for the next generation through crossover and mutation. This process is repeated for a specified number of generations or until sufficient performance is achieved. Genetic algorithms can explore a large search space and potentially find global optima but can be computationally expensive.

The evaluated hyperparameters were layer number and size, optimizer, loss function, activation function, dropout and validation dataset split. The results were similar, however the Bayesian algorithm performed better than the grid search, with the genetic algorithm showing the highest efficiency.

Additionally, Yu and Zhu [20] provides a comprehensive review of Hyper-Parameter Optimization (HPO) in deep neural networks. It discusses key hyper-parameters, such as learning rate and optimizer, and explores various search algorithms and trial schedulers for HPO, including grid search, random search and Bayesian optimization. The paper also compares different toolkits and services for HPO, including Google Vizier, Amazon SageMaker, NNI, and Ray.Tune. It concludes by discussing the comparability of different algorithms and evaluation methods for HPO. The paper emphasizes the importance of computational efficiency and effective evaluation in HPO for deep learning networks.

2.5 Model Evaluation and Metrics

Mehdiyeva, Enkec, Fettkea, *et al.* [21] provide a detailed description of accuracy measures and evaluate the performance of the prediction models using a dataset from the UCI Machine Learning Repository. They argue that there is no universally accepted single metric for choosing the appropriate forecasting method, and that different approaches can perform differently depending on the chosen metric. Therefore, they propose a framework that considers various accuracy metrics concurrently, providing a more robust comparability of classification methodologies.

The authors conclude by suggesting that the proposed multidimensional framework provides more robust rankings than one-dimensional analysis, and can be further improved by considering other aspects of classification algorithms such as ease of use, computational costs, and flexibility. In the context of this study, it is used to highlight the importance of including multiple metrics for accuracy and efficiency, and provides an overview of which ones to choose.

2.6 Research Gap

The art of lightning prediction using deep learning models has made significant progress, yet several critical research gaps remain that would require further exploration.

Firstly, there is a lack of comprehensive model comparison and benchmarking. While various architectures, such as LSTM, TCN, and FFNN have been utilized, most studies focus on the performance of a single model. A systematic comparison of multiple models under consistent conditions would provide valuable insights into their practical usability.

Secondly, the challenge of imbalanced datasets is significant, as lightning strike events are rare compared to non-strike events. Current literature often overlooks strategies to address this imbalance, such as oversampling, undersampling, or specialized loss functions, which could enhance model performance and reliability.

Additionally, there is a need for more in-depth analysis of feature selection and the optimal hyperparameters. While some studies touch upon these topics, they often lack a thorough examination of which meteorological parameters or hyperparameters are most predictive of lightning strikes. Understanding this relevance could lead to more effective models.

Chapter 3

Methodology

3.1 Datasets

3.1.1 Data Source

All data used in this project has been collected and generously provided by the Swedish Meteorological and Hydrological Institute (SMHI), a Swedish governmental agency operating under the Ministry of the Environment. SMHI plays a pivotal role in the collection, analysis, and dissemination of data in the domains of meteorology, hydrology, and oceanography [22]. All data procured from SMHI is made openly accessible to the public.

The utilized datasets comprise the *Lightning Archive* (hereafter referred to as the LIGHT dataset) and the *Meteorological Analysis Model Data* (hereafter referred to as the MESAN dataset). The former encompasses detailed information of all lightning strikes in Sweden since January 2, 2012, while the latter serves as a repository of the meteorological data required for the predictive modeling.

Using the Swedish Meteorological and Hydrological Institute (SMHI) as source of data ensures the reliability of the data and provides transparency and repeatability of the research. The ensuing sections will outline the process of constructing, analyzing, and pre-processing the datasets using a systematic workflow.

3.1.2 Data Overview

When developing machine learning models, the selection and curation of datasets play a pivotal role in the efficacy of the models [23]. In pursuit of constructing a robust predictive model for lightning strikes, two distinct types of datasets are required.

The primary LIGHT dataset, representing the labels (i.e. the target variables or outcomes that the models are trained to predict) for the models, consists exclusively of lightning strike data. This dataset contains two vital parameters: the timestamp and positional coordinates of each lightning strike. As this dataset solely serves as labels during the training and testing phases, these two parameters are the only ones required.

Complementing the LIGHT dataset is the MESAN dataset, which serves as the features (i.e. the parameters used as basis for the models prediction) for the models. This data encompasses meteorological and climate-related parameters from Scandinavia. In other countries, the specific parameters that are included in the meteorological dataset may vary depending on the geographical location, sensor accessibility, and data provisioning policies of the relevant organizations. Variations of the meteorological data may manifest in:

Included parameters: The specific meteorological parameters included in the dataset may vary to reflect the diverse climatic conditions of different regions.

Derived parameters: Certain parameters might be compound or highly correlated parameters, such as "Total Cloud Cover" when height-specific cloud covers are already included.

Sensor type and accuracy: Variability may arise in terms of the type and precision of sensors employed to capture meteorological data, influencing the overall quality and reliability of the dataset.

Unit of measurement: The units in which meteorological parameters are measured can vary, introducing considerations for standardization during data pre-processing.

The variations in the meteorological datasets require a flexible and adaptable model that can handle diverse data while maintaining accurate predictions. Additionally, by understanding the complexities of these datasets, it is possible to further improve the pre-processing and model configuration.

LIGHT Dataset

The LIGHT dataset has been thoroughly curated by aggregating raw sensor data on a centralized server and subjecting it to a series of calculations to obtain precise measurements [24]. SMHI has undertaken initial data processing and cleaning procedures as part of the dataset preparation, such as applying a Chi-Square test [25] to discard outliers.

Notably, while the underlying lightning data is inherently static (i.e. it follows a consistent pattern over time), the observations within the lightning archive exhibit dynamism. This dynamism occurs due to ongoing enhancements and modifications to the lightning detection system and sensors employed by SMHI. The most recent significant system upgrade occurred in 2014, leading to the exclusion of lightning strike data predating this upgrade throughout this study. The dataset encompasses a geographical volume of 5,847,709 km^3 (an area of 2,781,974 km^2), spanning latitudes between 55 and 70, longitudes between 10 and 25, and altitudes ranging from -2 to 2100 meters. Table 3.1 provides an overview of all included parameters within the dataset.

MESAN Dataset

The (MESAN) is a substantial dataset provided by SMHI that contains a wide range of weather and climate-related information [26]. The dataset includes various parameters, as shown in Table 3.2, each associated with a two-dimensional grid with a spatial resolution of 2.5^2 km. Some parameters, such as `snowfall in the last 24 hours`, are only available at specific time points. The grid covers a geographical area from latitude 52.3 to 71.5 and longitude -9.5 to 39.7, encompassing a region of 11,679,839 km^2 that includes Scandinavia, the Baltic States, and the northern part of Europe. It is important to note that the dataset is large, exceeding a terabyte of data.

SMHI has performed data cleaning procedures to rectify technical anomalies and improbable outliers to some extent. The dataset is available only from December 1, 2014, onward, following a major system upgrade in December 2014. According to SMHI, the dataset consists of a total of 29 parameters [27], which are detailed in Table 3.2

Name	**Unit**
Timestamp	ISO timestamp, timezone UTC
Position	Latitude/longitude
Current	kA
Multiplicity	Categorical (0 for stroke data, 1-99 for flash)
Number of sensors	Count
Degrees of freedom	Count
Ellipse angle	Degrees
Semi major axis	km
Semi minor axis	km
Chi square value	Fraction
Rise time	μs
Peak to zero time	μs
Max rate of rise	$kA/\mu s$
Cloud indicator	Boolean (1 if cloud discharge, 0 if cloud-to-ground)
Angle indicator	Boolean (1 if yes, 0 if no)
Signal indicator	Boolean (1 if yes, 0 if no)
Timing indicator	Boolean (1 if yes, 0 if no)

Table 3.1: Parameter overview of the LIGHT dataset.

3.1.3 Data Formats

LIGHT Dataset

The LIGHT dataset is distributed in a standardized Universal ASCII Lightning Format (UALF). It is available for download in the form of JSON files provided on a daily basis. Each JSON file comprises a singular list named `values`, encompassing all lightning strikes that occurred on that specific day. JSON is a widely used and flexible format, known for its simplicity in both understanding and parsing. However, due to the rewriting of parameter names for each lightning strike, significant storage space can be conserved by converting the dataset to a CSV format.

MESAN Dataset

The MESAN dataset is provided in the form of GRIB files. The GRIB (Gridded Binary) format stands as a ubiquitous method for storing both historical and forecasted meteorological data. It has evolved through three distinct versions: GRIB 0, GRIB 1, and GRIB 2. While GRIB 0 served as a proof of concept, GRIB 1 emerged as the most widely adopted and well-established version. Currently, GRIB 2 represents the latest iteration in this format's evolution, with ongoing efforts to transition towards its implementation.

In the context of this study, SMHI utilizes GRIB files conforming to the GRIB 1 standard. Within each file, a number of meteorological parameters are encapsulated, offering a comprehensive view of atmospheric conditions. The GRIB format is flexible and enables the incorporation of optional parameters, leaving it up to the publisher to determine what parameter to include. Each parameter has a corresponding two-dimensional array, containing observations gathered across spatial coordinates. The flexibility of the data contained within GRIB files, while providing a rich source of information, poses significant challenges during the processes of unpacking and decoding the data. As a result, the extraction of parameters

Name	**Unit**	**Height measurement**	**Height (m)**
Pressure	Pa	Above sea	0
Temperature	K	Above ground	2
Wet bulb temperature	K	Above ground	2
Maximum temperature	K	Above ground	2
Minimum temperature	K	Above ground	2
Visibility	m	Above ground	2
Wind gust	m/s	Above ground	10
U-component of wind	m/s	Above ground	10
V-component of wind	m/s	Above ground	10
Relative humidity	Fraction	Above ground	2
Total cloud cover	Fraction	Above ground	0
Low cloud cover	Fraction	Above ground	0
Medium cloud cover	Fraction	Above ground	0
High cloud cover	Fraction	Above ground	0
Fraction of significant clouds	Fraction	Above ground	0
cloud base of significant clouds	m	Above sea	0
cloud base of significant clouds	m	Above ground	0
Cloud Top of significant clouds	m	Above ground	0
Frozen part of total precipitation	Fraction	Above ground	0
Type of precipitation	Categorical	Above ground	0
Sort of precipitation	Categorical	Above ground	0
12 hour precipitation	mm	Above ground	0
24 hour precipitation	mm	Above ground	0
1 hour precipitation	mm	Above ground	0
3 hour precipitation	mm	Above ground	0
12 hour snow	cm	Above ground	0
24 hour snow	cm	Above ground	0
1 hour snow	cm	Above ground	0
3 hour snow	cm	Above ground	0

Table 3.2: Parameter overview of the MESAN dataset.

becomes an intricate and time-consuming task, demanding significant optimizations and error-handling techniques to increase the decoding time and ensure accurate interpretation.

To elaborate further, the 2D arrays within GRIB files serve as a spatial grid, collecting observations at specific geographical points. This grid-based representation facilitates the organization of meteorological data across large areas, allowing for the correlation of various parameters across both time and space. Important to note is that while the grid is divided into tiles of $2.5km^2$, the number of sensors are fewer and the grids are largely constructed using interpolation techniques.

GRIB-files provided by SMHI employ a rotated grid network or Lambert projection for data storage. Rotated coordinates involve relocating the South Pole from its conventional position at lat/lon -90/0. This strategic rotation is motivated by the desire to minimize the distance disparity between grid points in the northern and southern regions of the model area when the equator is shifted to align with Sweden's latitude. The rotation is applied to both the coordinates as well as parameters represented as vectors[1]. Notably, the same rotation is consistently applied across all files. Consequently, the vectors can be preserved in a rotated format, as it is their relational orientation with respect to other parameters that holds significance.

Since the grid is rotated, a new challenge arises: the need to recalibrate the coordinates to regular latitude and longitude. This recalibration is vital for the accurate interpretation and utilization of meteorological data stored in GRIB files. Fortunately, all requirements for this process is provided as metadata within the GRIB file.

3.1.4 Pre-processing

LIGHT Dataset

Filtering: Due to hardware limitations, exclusion of samples was deemed necessary. A geographic filter was applied and focus was restricted to the vicinity of the Swedish city Linköping within a range of $\pm 300\ km$, resulting in a square area covering 90,000 km^2. Another temporal filter was put into place to restrict the dataset to a span of 8 years between June 2015 and June 2023. This refined spatial and temporal scope may have the benefit to also contribute to enhancing the model's predictive accuracy, as local terrain and environmental conditions play a pivotal role in weather patterns [28]. By constraining samples to a similar location and point in time, it is possible to mitigate the impact of environmental variations on the predictive model.

Balancing: For effective binary classification, it is desirable to work with a balanced dataset, ideally maintaining an even ratio between positive and negative samples [29]. However, the initial dataset was exclusively comprised of positive samples. To rectify this imbalance, the task of generating and incorporating negative samples was performed. This involved creating samples representing "non-lightning strikes" with randomized timestamps and coordinates, while adhering to the previously established temporal and spatial constraints. The generated samples, if not present in the original dataset, signify locations and times where no lightning strikes occurred, thus serving as valid negative samples.

The balancing process involved the execution of the following steps:

1. Generate a timestamp, longitude, and latitude randomly within the predefined temporal and geographic constraints.
2. Verify if the generated label already exists in the dataset. If it does, return to step 1.
3. Include the new sample in the dataset as a negative.
4. Repeat the process until an adequate number of negative samples have been added.

Binning: An inherent challenge arises from the tendency of lightning strikes to manifest in sequences and clusters, as suggested in the analysis of the lightning dataset (see section 4.1 on page 26). To address this issue, a binning process was employed. Lightning strikes often occur in close succession, leading to high similarity or even identical samples in the positive dataset. The binning process served the purpose of reducing redundancy and enhancing the diversity of the dataset. The process involved flooring (i.e. rounding down) the timestamp of each lightning strike to the nearest hour, while rounding the coordinates

[1]In the MESAN dataset, only the U and V wind components are represented as vectors.

to a specified number of decimals. Samples that shared identical timestamp, latitude, and longitude after the binning operation were considered duplicates and was subsequently discarded. This step was essential to mitigate the skew introduced by clusters of similar samples, ensuring that the models were exposed to a more representative and varied set of positive examples.

The decision to floor timestamps hourly aligns with the MESAN dataset, which is exclusively comprised of hourly samples. The spatial bin size (i.e the granularity or level of detail used to group coordinates together) have been rounded to 0 decimals, resulting in a clustering radius of approximately 555.6 km that covers the entire extent of the LIGHT dataset. Consequently, each hourly cluster represents a single lightning strike. It's worth noting that the chosen spatial bin size may be sub-optimal and was dictated by hardware limitations. A more refined dataset could be achieved by increasing the decimal count, for example, to 3[2].

Given the ease of generating negative samples, the discarding process prioritizes negative samples. Following the binning process, the dataset was re-balanced to ensure a neutral class ratio.

MESAN Dataset

Extraction: As the MESAN data was stored in an encoded GRIB format, the measurements had to be extracted to be compatible with the models. In this study, a trailing sequence of 73 weather observations were extracted for each lightning strike, with each observation representing an hourly point in time leading up to the corresponding lightning strike. Older data beyond this 3-day window were ignored due to hardware restrictions. To extract a single MESAN observation, the required GRIB file was loaded. All weather parameters were derived by computing the mean of values observed within a three-kilometer radius of the corresponding lightning strike. If no observations were present within this area, the radius was automatically extended up to six kilometers before labeling the value as "missing". The six-kilometer threshold was set in order to include the maximum number of observations while limiting their relative variance. As the coordinates included in the GRIB files were rotated (refer to section 3.1.3 on page 13) they had to be un-rotated to match the regular coordinate format[3].

Imputation: The resulting dataset exhibits numerous missing values. To preserve the continuity of the time sequences, missing values undergo imputation by replacing them with the closest known values – either from the subsequent closest value or from the preceding closest value (with priority given to the latter). It is noteworthy that imputation adheres to the group boundaries defined by each lightning indices. Thus, a missing value at the onset of the time sequence for lightning strike #2 cannot be substituted with the final value from lightning strike #1. All sequences with remaining NA values after imputation, are dropped entirely. The number of missing values were deemed low enough to not have a significant effect on the end results.

[2]A decimal number of 3 results in a radius of 55.6 km.

[3]The most common format used for global positioning is the WGS84 standard.

3.2 Model Selection

The models were to be provided with a timeseries sequence of meteorological parameters measured at a specific location and a specific point in time. Each sequence lead up to either a lightning strike or a non-lightning strike event. The objective of the models was to produce a percentage representing the likelihood of a lightning strike occurring at the end of the sequence, and conclude whether a lightning strike would or would not occur.

To achieve this four distinct models architectures were selected:

1. **DNN:** This model was chosen to serve as a baseline, representing a simpler and more traditional approach to neural networks without temporal data handling.
2. **SRNN:** This model was chosen because it represents the simplest form of recurrent neural networks, capable of processing sequences but with limitations in handling long-term dependencies.
3. **LSTM:** This model was chosen because it is specifically designed to overcome the limitations of SRNNs, such as the vanishing gradient problem, which allows it to capture long-term dependencies.
4. **GRU:** This model was chosen because it retains the advantages of LSTMs in handling long-term dependencies but with fewer parameters, leading to potentially faster training times and reduced computational cost.

3.2.1 Dense Neural Network

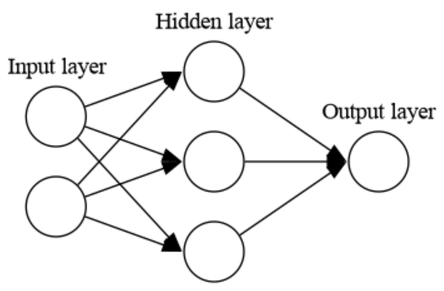
Dense Neural Networks (DNN) models, also known as fully connected neural networks, are the fundamental type of deep learning models. They consist of an input layer, an output layer, and one or more hidden layers in between. Each hidden layer contains multiple perceptrons, also called neurons, which have their own weights for each input signal, a bias term, and an activation function. The perceptrons in each layer receive input signals from all the perceptrons in the previous layer and transmit their output signals to all the perceptrons in the next layer. Figure 3.1a illustrates the structure of a dense neural network, while figure 3.1b depicts the functioning of an individual perceptron.

Although dense neural networks do not possess the recurrent nature of other models, they are still valuable in establishing a baseline performance and providing an initial representation of how neural networks perform on the data without considering the temporal aspect. It is important to note that the assumption of lower accuracy for dense neural networks is not universally true and can vary depending on the specific problem and dataset.

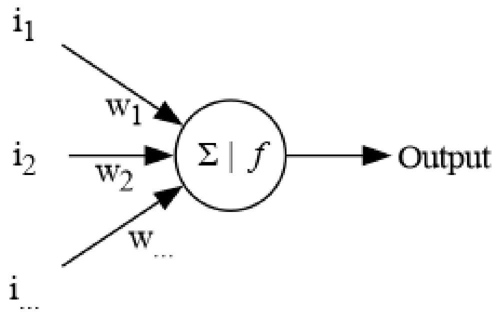
3.2.2 Simple Recurrent Neural Network

Simple Recurrent Neural Networks (SRNN) [30] are a type of neural network specifically designed to handle sequential data. Unlike traditional feedforward neural networks, SRNNs have the ability to capture temporal dependencies and extract patterns from time sequences of multiple features, adding another dimension to the input data. This temporal aspect is crucial in domains such as time series prediction where the changes in observed values over time are more important than the individual observations themselves [31].

The key feature of RNNs is the presence of recurrent connections, which allow information to be passed from one step in the sequence to the next. This enables the network to maintain an internal memory or state, which can be updated and used to influence future



(a) Overview of dense neural networks.



(b) Inner workings of an individual perceptron.

predictions. The recurrent connections create a feedback loop that allows the network to consider the entire history of the sequence when making predictions at each step.

Figure 3.2 provides an illustration of how recurrent neural networks work. It shows the flow of information through the network at each time step, with the recurrent connections allowing the network to incorporate information from previous steps.

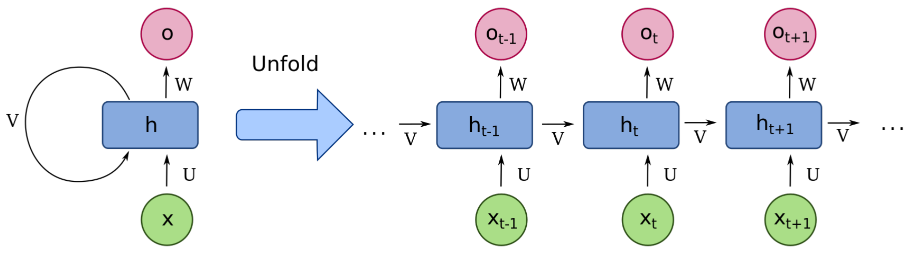


Figure 3.2: Overview of a simple RNN layer [32].

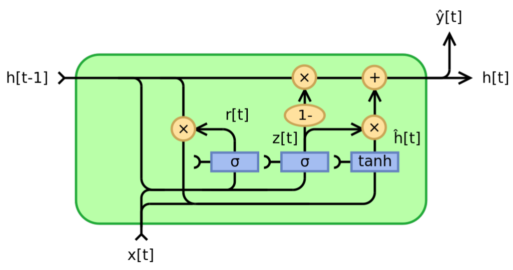
3.2.3 Long Short Term Memory

SRNNs face challenges in capturing long-term dependencies in sequences. As the sequence length increases, the ability of SRNNs to retain and utilize information from earlier time steps diminishes due to the vanishing gradient problem [4].

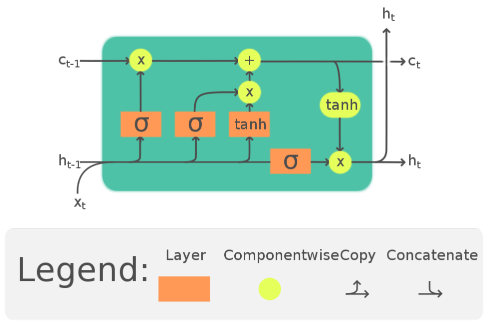
The LSTM model was designed to address the vanishing gradient problem more effectively. It consists of three key gates: the forget gate, input gate, and output gate. These gates enable LSTMs to selectively process and store information over sequential data, addressing the vanishing gradient problem. The forget gate determines which information from the previous cell state should be discarded, the input gate decides what new information to store in the cell state, and the output gate regulates what information to output as the hidden state. These gates enable LSTMs to capture long-term dependencies by controlling the flow of information through the memory cell, allowing the model to retain and utilize relevant information while discarding irrelevant portions, which is important for effective sequence learning. An overview of LSTM models is presented in figure 3.3b.

3.2.4 Gated Recurrent Unit

The GRU model is a newer variation of recurrent neural networks. GRUs are similar to LSTMs in that they use a gating mechanisms to selectively update and reset its internal state, allowing it to capture long-term dependencies more efficiently than SRNNs [33]. GRUs have shown to have a similar but slightly lower accuracy as LSTM layers, while being more computationally efficient [34]. See figure 3.3a for an overview of how GRU models work.



(a) Overview of a GRU layer [35].



(b) Overview of a LSTM layer [36].

3.3 Optimizing Input Parameters

In order to determine the optimal features for lightning strike prediction, a three-fold strategy was used. The first part involved a manual analysis of both the LIGHT and MESAN data to exclude eventual parameters with a high number of missing values. A visualization of a subset of the LIGHT dataset, serving as the foundation to the binning process, was developed as shown in figure 4.1. Secondly, a covariance matrix was constructed to analyze the correlations between parameters. The third part extended the use of the covariance matrix to conduct a Principal Component Analysis.

3.3.1 Covariance Matrix

The correlation analysis was performed using the built-in correlation functionality in R, which uses the Pearson correlation coefficient by default. However, the Pearson correlation coefficient assumes that the underlying data follows a normal distribution. To check for normality among the parameters, the Shapiro-Wilk test was used. The Shapiro-Wilk test is a statistical test that determines if a set of samples follows a normal distribution, and is calculated using equation 3.1 [37]. The results of the test indicate that none of the parameters are normally distributed, meaning an alternative correlation method must be used.

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{3.1}$$

where

- $x_{(i)}$ with parentheses enclosing the subscript index i is the ith order statistic, i.e., the ith-smallest number in the sample (not to be confused with x_i).
- $\overline{x} = (x_1 + \cdots + x_n)/n$ is the sample mean.

One alternative correlation method is the Spearman's rank correlation coefficient. Unlike the Pearson correlation coefficient, the Spearman's rank correlation coefficient is non-parametric, meaning it does not make any assumptions about the underlying distribution of the data. It is also robust against outliers and can be calculated efficiently. Before calculating the Spearman's rank correlation coefficient, all parameters were normalized.

When training deep neural networks, it is preferable to have fully orthogonal and independent parameters [18]. This makes the model more efficient as it reduces the amount of input data and calculations required. Therefore, parameters with high correlation may be dropped. However, it is important to note that the statistical tests are not definitive, and it is possible that the removed parameters may prove beneficial.

Based on the covariance matrices presented in section 4.2.2 on page 27, the following parameters were removed due to their high correlation with other, more orthogonal parameters:

- Wet-bulb temperature
- Cloud base of significant clouds above sea
- Total cloud cover
- Wind gust

3.3.2 Principle Component Analysis

In order to gain a deeper understanding of the MESAN data, a Principal Component Analysis (PCA) was conducted. Humans often struggle to visualize and comprehend data in dimensions higher than three. PCA is a commonly used technique that aims to reduce the dataset into its principal components, a set of orthogonal vectors representing the eigenvalues of the dataset matrix. The primary goal of PCA is to minimize the number of features while retaining the maximum amount of information. Additionally, PCA can be useful for data exploration, providing a more comprehensive understanding of the data. The first principal component (PC1) captures the largest variance in the data. By examining the parameters that have the greatest influence on PC1, it is possible to determine which specific parameters have the most impact on the dataset as a whole.

3.4 Hyperparameters Tuning

A neural network consists of two types of hyperparameters: internal and external. Internal hyperparameters, such as perceptron weights and biases, are automatically optimized during the training process using the training data. On the other hand, external hyperparameters, such as the number of layers, the type and size of each layer, activation functions, and the number of training epochs, need to be manually configured.

The optimization of external hyperparameters is important for achieving good performance in neural networks. Two commonly used methods for optimizing these hyperparameters are random search and grid search. In random search, models are constructed and evaluated with randomly selected hyperparameter values. In grid search, all possible combinations of hyperparameters are evaluated. However, these methods can be computationally inefficient, especially for large search spaces.

Genetic algorithms have shown promising results for hyperparameter tuning compared to random and grid search methods [19]. Due to the large number of parameters being tested, conducting grid search would be time-consuming, as such in this study a genetic algorithm was used for hyperparameter tuning. The genetic algorithm consists of the following steps:

1. An initial population of 100 candidates is generated, where each candidate represents a random combination of hyperparameters.
2. A fitness value is calculated for each candidate based on its measured accuracy, with an additional penalty for longer training times. The exact formula is presented in equation 3.2.
3. The top 10% best performing candidates survive to create the next generation.
4. The remaining new candidates are created by sampling two random sequences of the initial candidates, with the probability of selection proportional to their fitness. The candidates in the two sequences are paired and merged to create offspring. Each parameter of the offspring is copied from one of the parents at random, with a 5% chance of mutation where the parameter is changed to random value.
5. This process starts anew from step 1.

$$\text{fitness} = \text{accuracy} - \frac{\text{training time}}{24000} \tag{3.2}$$

The base architecture of the model used for hyperparameter tuning consists of an input layer followed by a number of recurrent layers, then a number of dense layers, with a final output layer. The purpose of the recurrent layers is to capture the temporal changes in the input data, while the dense layers abstract this knowledge to learn more complex patterns. Note that when evaluating the DNN model, the recurrent layers are replaced with dense layers.

The genetic algorithm was used to determine the optimal values for the following twelve parameters:

- Batch size: The number of samples processed in each training iteration.
- Optimizer: The algorithm used to update the model's weights during training.
- Activation function for the recurrent layers: The function applied to the output of the recurrent layers.
- Activation function for the dense layers: The function applied to the output of the dense layers.
- Dropout for the recurrent layers: The fraction of input units to drop during training to prevent overfitting.
- Dropout for the dense layers: The fraction of input units to drop during training to prevent overfitting.
- Number of recurrent layers: The number of recurrent layers in the model.
- Number of dense layers: The number of dense layers in the model.
- Number of units in the recurrent layers: The number of neurons in each recurrent layer.
- Number of units in the dense layers: The number of neurons in each dense layer.
- Normalization between recurrent layers: Whether to apply normalization between the recurrent layers.

- Normalization between dense layers: Whether to apply normalization between the dense layers.

Due to the computational resources required for training hundreds of models through hundreds of generations, only three generations were simulated in this project, making the method similar to a random search. The resulting hyperparameters were copied onto all model types in the final evaluation.

3.5 Model Evaluation

3.5.1 Stratified K-Fold Cross-Validation

When training machine learning models, it is desirable to minimize the impact of randomness and luck. Simply splitting the dataset into two parts, one for training and one for testing, can lead to highly variable results and may not accurately represent the model's performance.

Stratified k-fold cross-validation is a strategy that can be applied to mitigate the impact of randomness. In this study, a k value of $k = 10$ has been chosen. The method consists of the following steps:

1. Split the data into k different partitions, where each partition contains an equal number of positive and negative samples.
2. Iterate through the training and evaluation phase k times, with each iteration using a different partition as the test set and the remaining partitions as the training set.
3. Calculate the evaluation metric for each iteration and set the final evaluation metric as the average of all the test results.

All following metrics were determined using this method.

3.5.2 Training Time

The training time of a deep learning model is an important factor to consider for several reasons. Firstly, it provides an indication of the computational resources required by the model. This is particularly important when deploying the model in real-time applications where fast predictions are necessary.

Secondly, the training time impacts the development process of the model. During hyper-parameter tuning and model development, it is essential to have a manageable training time to iterate and experiment with different configurations effectively. A shorter training time allows for faster experimentation and quicker feedback on the model's performance.

Lastly, the need for retraining the model should also be taken into account. Environmental factors, such as changes in lightning patterns or new geographical locations, may require the model to be updated or retrained periodically. A shorter training time facilitates this process, enabling the model to adapt to new data and maintain its accuracy over time.

3.5.3 F1 Score

The F1 score is a metric commonly used in machine learning evaluation as it provides a more comprehensive metric than simple accuracy. It is a compound metric derived multiple independent metrics.

Accuracy: Accuracy is the most prevalent and simplest type of machine learning metric. It gives an indication of the frequency of correct predictions made by the model. It is calculated as the ratio of the number of correct predictions to the total number of predictions made by the model. The formula for accuracy is shown in equation 3.3.

$$\text{accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} \tag{3.3}$$

Binary Accuracy: A critical issue arises when using accuracy on binary classification models. The model outputs a probability that represents how likely a sample belongs to a class, rather than the actual class itself. Because of this, using simple accuracy would constantly result in 0% accuracy as very few of the predictions match the label exactly. To accommodate for this, binary accuracy rounds the output of the model to the closest class before evaluating, making it a practically feasible metric.

Precision & Recall: While binary accuracy is a commonly used metric for evaluating model performance, it may not provide a complete picture, especially when dealing with imbalanced testing data. To address this, two additional metrics, precision and recall, are introduced.

Precision measures the accuracy of the model's positive predictions. It answers the question: "Out of all samples predicted as positive, how many were actually positive?". A high precision indicates a low rate of false positives. The formula for calculating precision is shown in equation 3.4.

Recall, on the other hand, measures the model's ability to correctly identify all positive samples. It answers the question: "Out of all actual positive samples, how many were correctly predicted?". A high recall indicates that the model is effective at identifying positive samples, but it may have a higher rate of false negatives. The formula for calculating precision is shown in equation 3.5.

$$\frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{3.4}$$

$$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{3.5}$$

F1-Score: Finally, the F1-score is a metric used to measure the performance of a classification model. It is the harmonic mean of precision and recall. The F1-score combines these two metrics to provide a single value that represents the balance between precision and recall, and can thus be used for model comparison. It ranges from 0 to 1, with 1 being the best possible score. A higher F1-score indicates a better overall performance of the classification model. Equation 3.6 shows how the f1-score is calculated.

$$2\frac{precision \cdot recall}{precision + recall} \tag{3.6}$$

3.5.4 Mean Absolute Error

The mean absolute error (MAE) metric is used to measure the average level of confidence the model has in its predictions. A lower value indicates a higher confidence, as the individual predictions deviate from the true value by this average amount. While being an uncommon metric as it is the accuracy that count, it can be a useful metric when comparing models with very similar accuracy.

3.5.5 Wilson Score

The Wilson Score is a statistical method used to estimate the confidence interval for binary accuracy. It provides a range of values within which the true accuracy of the model is likely to fall, with a chosen confidence level of 95% used in this study. The Wilson Score helps assess the accuracy of the model.

3.6 Time frames

In order to assess the performance of the models across different time frames, two additional evaluation factors are introduced: *lookback* and *lookahead*.

The lookback refers to the size of the temporal dimension of each feature sequence. A higher lookback value increases the time window, while a lower value decreases it. For instance, a lookback of 24 implies that the model can utilize one day of data, with an hourly frequency, as input.

On the other hand, the lookahead signifies the size of the window within which the model is expected to make predictions. A lookahead of 24 means that the model should predict the occurrence of a lightning strike within a 24-hour period, while a lookahead of 1 means predicting if a lightning strike will happen within the next hour.

In order to comprehensively evaluate the models, various combinations of lookback and lookahead values are explored. This allows for analyzing the impact of different time frames on the performance of the DL models. By considering a range of lookback and lookahead values, it is possible to gain insights into the optimal time window for predicting lightning strikes. Additionally, by examining the effect of different time frames on the model's accuracy, it is possible to determine the most suitable time frame for practical applications. Due to the limited number of available observations however, the largest combination of lookback + lookahead that can be used cannot exceed 73 hours.

Furthermore, it is worth mentioning that the choice of lookback and lookahead values should be guided by the specific requirements of the application. For instance, if the goal is to provide real-time lightning strike predictions for immediate safety measures, a shorter lookahead period, such as 1 hour, may be more appropriate. On the other hand, if the objective is to forecast lightning strikes for longer-term planning, a larger lookahead period, such as 24 hours, could be more beneficial.

3.6.1 Data Structuring

Before training the model, the data needs to be restructured based on the lookback and lookahead parameters. Changing the lookback is straightforward, as it involves adjusting the number of observations in the lookback sequence leading up to the lightning strike. However, changing the lookahead is more complex.

When increasing the lookahead to a value greater than 1, a lookahead sequence needs to be created where the lightning strike can occur at any time point, not just the first time point. This requires shifting the lookahead sequence backwards by a random amount, which renders a portion of the lookback sequence unusable. Consider figure 3.4, where the lookback sequence X leading up to lightning strike Y has to be shifted to allow Y to occur at any point in the lookahead sequence.

```
_ _ x x x x | y
_ x x x x | _ y
x x x x | _ _ y
```

Figure 3.4: Illustration of lookahead shifting.

Furthermore, within the newly created lookahead sequence, another independent lightning strike may have occurred, either at the same location or a nearby one. Therefore, the binning method used previously (see Section 3.1.4 on page 15) needs to be reapplied to every other lightning strike in the LIGHT dataset. If another lightning strike has occurred within the lookahead sequence, the label is changed to positive, regardless of its previous class. Consider figure 3.5, where a negative samples becomes positive after lightning strike Z gets included in the lookahead.

```
x x x x | _
x x x x | _ _ z
```

Figure 3.5: An independent lightning strike occurring within the lookahead sequence.

When changing the distribution of labels, it is important to again consider class skewness. After restructuring the dataset to accommodate the lookahead parameter, it is likely that the dataset is biased towards a positive label. At this stage, it is not feasible to regenerate or change the dataset. Therefore, class weights are used as a last resort. The class weighting mechanism examines the restructured dataset and calculates the ratio between positive and negative samples. This ratio is then used to adjust the weight or importance of each label in that class during training.

Chapter 4

Results and Analysis

4.1 Dataset Analysis: LIGHT

As described in section 3.3 on page 19, a visualization of a LIGHT subsample from January 1st, 2020 was developed and is shown in figure 4.1. The visualization features a five-dimensional graph and depicts the geographical distribution of lightning strikes, with longitude and latitude represented by the x and y axes respectively. The z-axis indicates the number of lightning strikes that occurred at each location, while the color of the dots represents the time of occurrence.

From the graph, we can observe that lightning strikes tend to occur in sequences with close temporal proximity. This suggests that binning or grouping of the lightning strikes may prove beneficial.

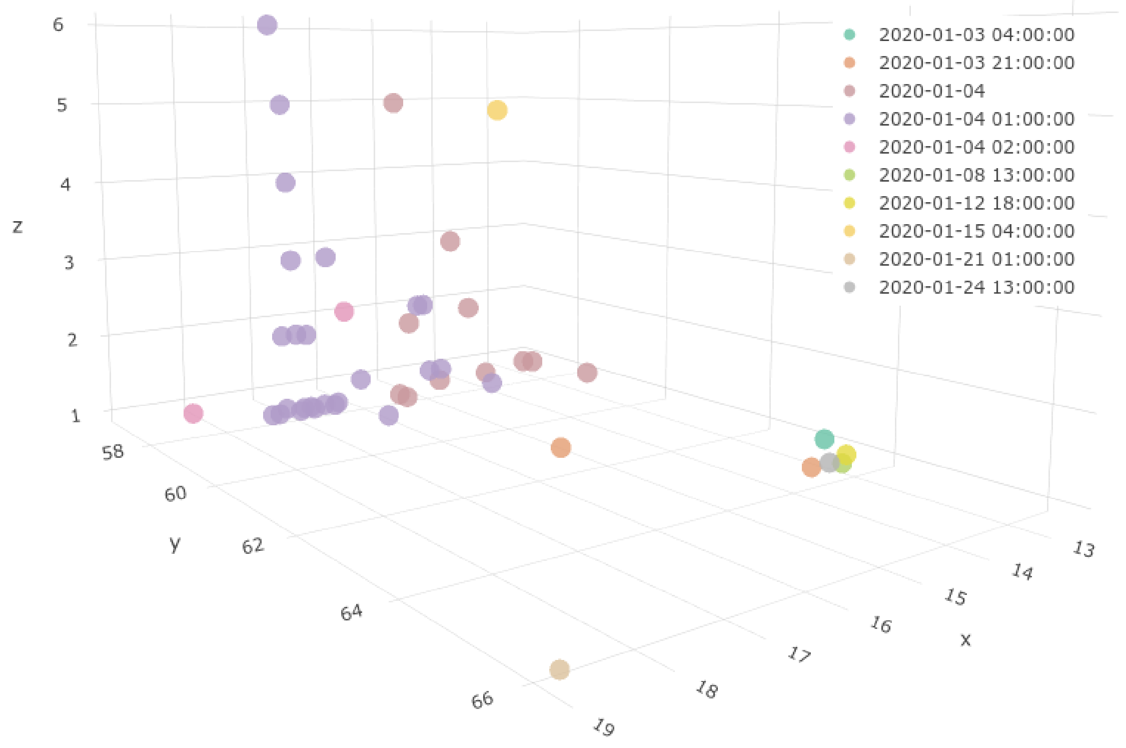


Figure 4.1: Visualization of the LIGHT dataset during January 1st, 2020.

4.2 Dataset Analysis: MESAN

4.2.1 Exploratory Analysis

Upon manual inspection, it is evident that the MESAN dataset contains numerous observations with missing values. Some variables, such as `24-hour precipitation` and `12-hour snow` have all values missing except for at the specific intervals. Additionally, the parameters `minimum temperature` and `maximum temperature` are completely missing, likely due to a mistake in SMHI's documentation. All empty or time-dependent fields were dropped from the dataset as they do not provide any additional information to the models.

Furthermore, it is assumed that certain fields are highly correlated, such as derived and related measurements. To verify this assumption, a covariance matrix was utilized (see section 4.2.2).

4.2.2 Correlation Analysis

The covariance matrix for the MESAN dataset provides information about the correlation between different parameters. Figure 4.2 shows the correlation matrix for a subset of the MESAN data, which consists of 458,328 observations. The `Snowfall` parameter was excluded from the analysis as it only contained zeroes.

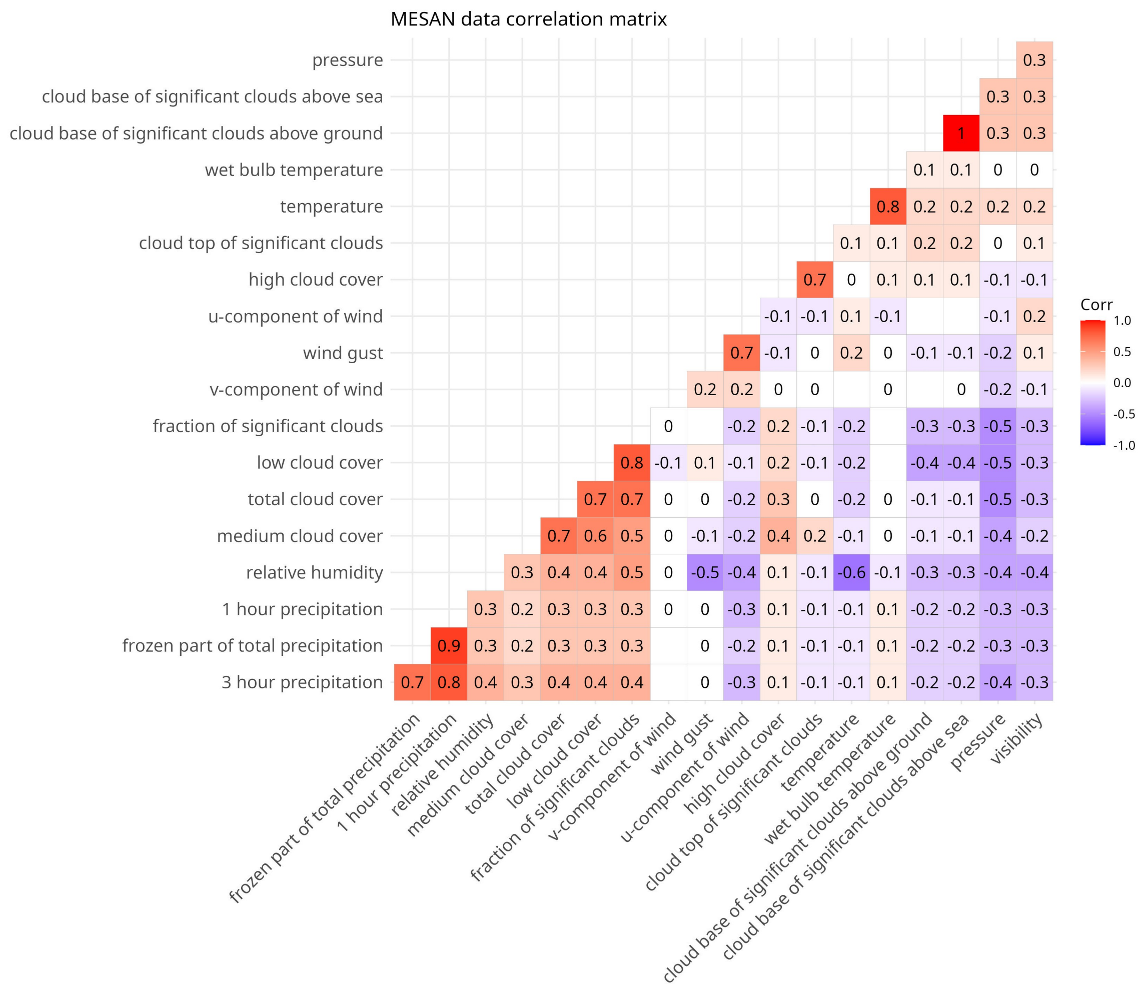


Figure 4.2: Covariance matrix of the MESAN dataset, before analysis.

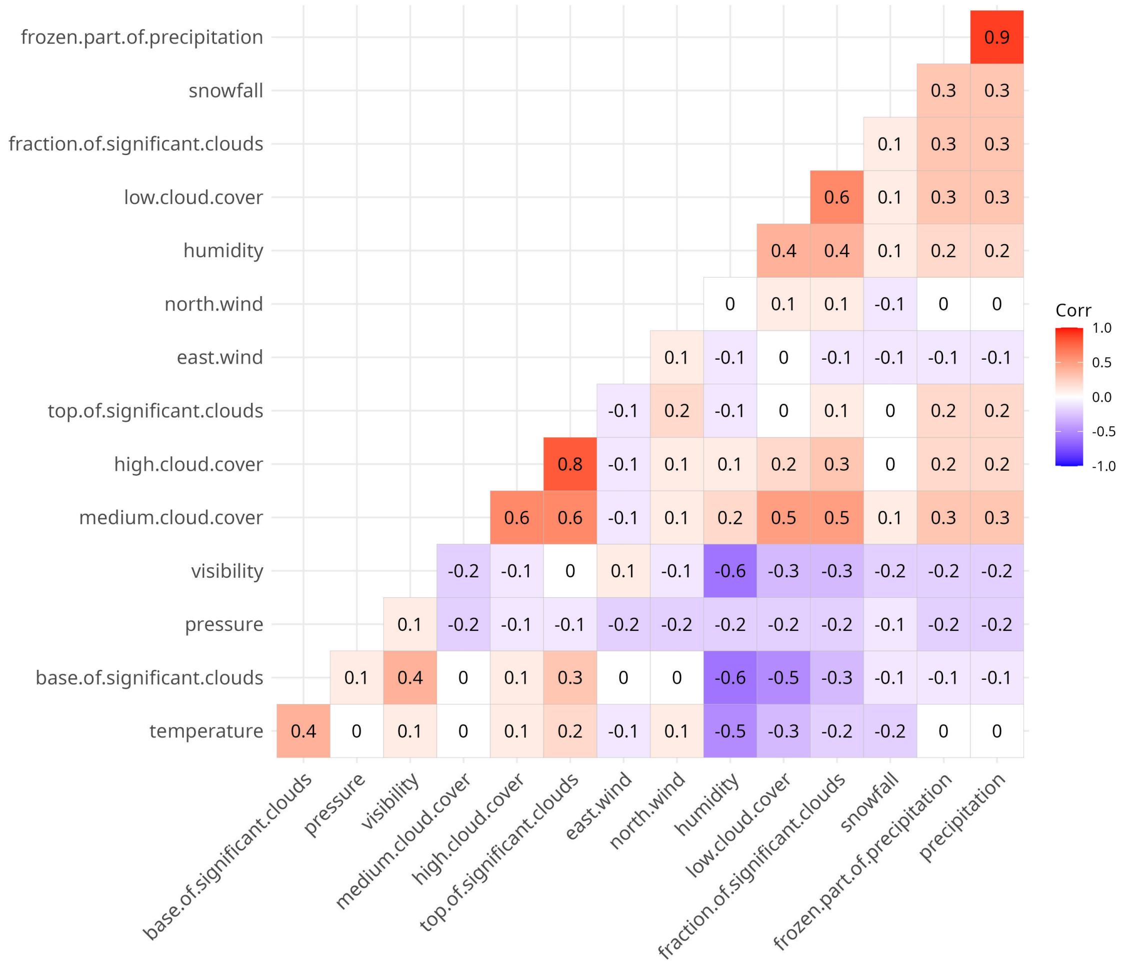


Figure 4.3: Covariance matrix of the MESAN dataset, after analysis.

The correlation analysis reveals that some parameters have a high correlation coefficient (0.8 or higher), as shown in table 4.1, while others have a medium correlation coefficient (between 0.6 and 0.8), as shown in table 4.2.

Parameter 1	**Parameter 2**	**Correlation**
1h precipitation	3h precipitation	0.8
1h precipitation	frozen part of total precipitation	0.9
Low cloud cover	fraction of significant clouds	0.8
Temperature	wet-bulb temperature	0.8
Cloud base of significant clouds above ground	cloud base of significant clouds above sea	1

Table 4.1: Parameter pairs with high correlation (0.8+).

Parameter 1	**Parameter 2**	**Correlation**
3h precipitation	Frozen part of total precipitation	0.7
Relative humidity	Temperature	-0.6
Medium cloud cover	Total cloud cover	0.7
Medium cloud cover	Low cloud cover	0.6
Total cloud cover	Low cloud cover	0.7
Total cloud cover	Fraction of significant clouds	0.7
Wind gust	U-component of wind	0.7
High cloud cover	cloud top of significant clouds	0.7

Table 4.2: Parameter pairs with medium correlation (between 0.6 and 0.8).

4.2.3 Principle Component Analysis

Table 4.3 presents a summary of the principal components, while Table 4.4 provides a detailed analysis of the loadings of the top three principal components, together comprised of over 50% of the variance in the MESAN dataset.

PC	Standard deviation	Proportion of variance	Cumulative proportion
1	1.92	0.25	0.25
2	1.56	0.16	0.40
3	1.18	0.09	0.50
4	1.13	0.09	0.59
5	1.02	0.07	0.66
6	0.96	0.06	0.72
7	0.92	0.06	0.77
8	0.85	0.05	0.82
9	0.81	0.04	0.87
10	0.76	0.04	0.90
11	0.60	0.02	0.93
12	0.59	0.02	0.95
13	0.56	0.02	0.97
14	0.49	0.02	0.99
15	0.43	0.01	1.00

Table 4.3: Summary of the principle component analysis.

Parameter	PC1	PC2	PC3
pressure	-0.21	-0.02	-0.02
temperature	-0.15	0.37	-0.10
visibility	-0.28	0.15	0.08
east.wind	-0.04	-0.08	0.18
north.wind	0.09	0.12	0.17
humidity	0.32	-0.28	0.07
low.cloud.cover	0.39	-0.15	0.21
medium.cloud.cover	0.37	0.25	0.22
high.cloud.cover	0.28	0.43	0.17
fraction.of.significant.clouds	0.37	-0.06	0.20
base.of.significant.clouds	-0.21	0.43	-0.05
top.of.significant.clouds	0.17	0.51	0.11
frozen.part.of.precipitation	0.30	0.08	-0.55
precipitation	0.25	0.09	-0.56
snowfall	0.13	-0.04	-0.35

Table 4.4: Loading of the first three PCs, together representing over 50% of the variation of the dataset.

4.3 Hyperparameter Tuning

The following list shows the resulting hyperparameters derived by the genetic algorithm (refer to section 3.4 on page 20):

- Batch size: 128
- Optimizer: Adam
- Activation function for the recurrent layers: tanh
- Activation function for the dense layers: relu
- Dropout for the recurrent layers: 0.2
- Dropout for the dense layers: 0.2
- Number of recurrent layers: 2
- Number of dense layers: 0
- Number of units in the recurrent layers: 256
- Number of units in the dense layers: 64
- Normalization between recurrent layers: True
- Normalization between dense layers: True

It is important to note that these hyperparameters might not be the most efficient in every scenario, and will likely have to be re-evaluated after making changes in the dataset or regarding model type and lookback/lookahead combination.

4.4 Model Evaluation

Table 4.5, 4.6, 4.7 and 4.8 presents the metrics gathered from the evaluation of DNN, SRNN, LSTM and GRU models respectively. All models were trained over eight epochs using the stratified 10-fold cross validation method. As such, each metric is an average over ten iterations, where the original dataset is split into ten parts, eight parts used for training, one for validation and one for testing in each fold.

Each table row shows the parameters lookback, lookahead, training time (TT), accuracy (AC), F1-Score (F1), Mean Absolute Error (ME) and Wilson Score (WS) measured from both the training and testing phases of each model. Lookback and lookahead are specified in hours, training time in seconds and the remaining four metrics as percentages. As the feature sequence have to be randomly shifted for every lookahead value, some combinations of lookback and lookahead (e.g. 72 and 3) exceeds the available time points, and the metrics are labeled as missing (NA). As the dense model is unable to consider timeseries (see 3.2.1 on page 17), the only lookback available is `1`.

Lookback	**Lookahead**	**TT**	**AC**	**F1**	**ME**	**WS**
1	1	8.77	88.57	89.08	14.83	12.27
1	3	8.80	88.84	89.37	14.61	12.00
1	6	8.75	88.98	89.57	14.37	11.85
1	12	8.79	89.00	89.75	14.44	11.83
1	24	8.79	88.85	89.86	14.54	11.99

Table 4.5: Metrics from model: DENSE

Lookback	**Lookahead**	**TT**	**AC**	**F1**	**ME**	**WS**
1	1	132.04	89.33	89.77	13.79	11.49
1	3	130.83	89.37	89.91	13.72	11.45
1	6	130.02	89.60	90.14	13.46	11.21
1	12	130.85	89.19	89.93	13.96	11.63
1	24	131.12	88.73	89.84	14.24	12.11
3	1	130.19	89.14	89.62	14.01	11.68
3	3	129.05	89.08	89.46	14.67	11.74
3	6	129.22	89.54	90.09	13.53	11.28
3	12	130.54	89.49	90.17	13.66	11.33
3	24	130.01	89.36	90.31	13.72	11.46
6	1	130.29	89.18	89.66	14.03	11.64
6	3	130.78	89.32	89.81	13.70	11.50
6	6	130.64	89.00	89.57	14.05	11.83
6	12	131.10	89.57	90.27	13.58	11.24
6	24	130.55	89.46	90.45	13.69	11.36
12	1	129.52	88.78	89.26	14.60	12.06
12	3	131.15	88.81	89.33	14.41	12.03
12	6	128.89	89.54	90.09	13.46	11.27
12	12	130.35	89.71	90.43	13.32	11.10
12	24	129.01	88.98	89.93	14.41	11.85
24	1	129.58	88.57	89.13	14.62	12.27
24	3	130.07	89.33	89.83	13.69	11.49
24	6	129.76	89.71	90.16	13.53	11.10
24	12	131.16	89.23	89.91	14.22	11.59
24	24	129.80	89.24	90.16	13.70	11.58
48	1	128.39	89.39	89.84	13.77	11.43
48	3	129.28	89.35	89.81	13.84	11.47
48	6	128.58	89.28	89.82	14.04	11.54
48	12	129.65	89.36	90.11	13.84	11.46
48	24	129.65	88.76	89.85	14.53	12.08
72	1	130.95	89.15	89.61	14.12	11.67
72	3	NA	NA	NA	NA	NA
72	6	NA	NA	NA	NA	NA
72	12	NA	NA	NA	NA	NA
72	24	NA	NA	NA	NA	NA

Table 4.6: Metrics from model: Simple RNN

Lookback	**Lookahead**	**TT**	**AC**	**F1**	**ME**	**WS**
1	1	57.92	91.77	92.05	10.37	8.96
1	3	57.76	91.96	92.25	10.09	8.76
1	6	57.66	91.42	91.69	10.93	9.32
1	12	57.56	87.93	87.94	16.02	12.93
1	24	57.36	85.09	85.01	19.93	15.85
3	1	57.59	79.25	29.77	24.36	21.83
3	3	57.45	81.06	38.78	23.83	19.98
3	6	57.56	80.86	47.96	24.85	20.19
3	12	57.54	80.82	58.46	24.29	20.23
3	24	57.40	80.09	66.76	25.39	20.97
6	1	57.62	79.83	22.31	24.06	21.24
6	3	57.68	81.37	30.83	22.95	19.67
6	6	57.54	81.39	37.86	22.77	19.65
6	12	57.46	81.52	48.71	23.81	19.51
6	24	57.39	82.21	60.31	23.83	18.81
12	1	57.65	81.07	19.56	25.31	19.97
12	3	57.65	83.54	26.70	23.51	17.45
12	6	57.62	81.66	33.38	24.57	19.37
12	12	57.62	79.83	43.01	25.73	21.24
12	24	57.54	77.86	53.83	27.49	23.24
24	1	57.62	77.53	17.36	27.76	23.58
24	3	57.57	80.35	24.17	25.78	20.71
24	6	57.44	78.96	29.66	27.04	22.12
24	12	57.51	81.93	40.14	25.21	19.10
24	24	57.43	75.02	49.11	29.05	26.13
48	1	57.68	70.53	12.59	31.48	30.69
48	3	57.59	69.44	17.71	33.07	31.79
48	6	57.58	73.47	24.20	30.68	27.71
48	12	57.49	75.90	34.89	29.41	25.24
48	24	57.51	79.12	46.83	27.14	21.96
72	1	57.69	85.46	11.07	27.43	15.47
72	3	NA	NA	NA	NA	NA
72	6	NA	NA	NA	NA	NA
72	12	NA	NA	NA	NA	NA
72	24	NA	NA	NA	NA	NA

Table 4.7: Metrics from model: LSTM

Lookback	**Lookahead**	**TT**	**AC**	**F1**	**ME**	**WS**
1	1	48.67	91.94	92.20	9.73	8.78
1	3	48.51	92.19	92.47	9.45	8.52
1	6	48.42	91.60	91.87	10.36	9.14
1	12	48.46	88.24	88.24	15.34	12.61
1	24	48.22	85.26	85.03	19.19	15.68
3	1	48.53	80.75	31.06	23.11	20.30
3	3	48.45	81.90	39.71	23.15	19.13
3	6	48.40	82.07	48.82	23.18	18.95
3	12	48.29	81.16	59.21	23.76	19.87
3	24	48.25	79.99	66.83	24.34	21.07
6	1	48.37	80.81	23.56	22.64	20.24
6	3	48.42	83.27	32.52	20.89	17.73
6	6	48.34	81.88	39.46	21.36	19.14
6	12	48.32	82.73	50.62	21.46	18.27
6	24	48.19	83.11	62.34	21.97	17.89
12	1	48.42	82.74	23.04	21.29	18.27
12	3	48.38	83.50	30.65	20.65	17.48
12	6	48.35	83.09	37.26	21.35	17.91
12	12	48.32	81.34	47.03	23.42	19.69
12	24	48.07	80.11	57.42	24.31	20.95
24	1	48.24	82.18	19.98	23.20	18.84
24	3	48.37	79.52	24.41	25.90	21.55
24	6	48.30	81.71	32.17	23.94	19.32
24	12	48.18	82.46	43.36	22.99	18.55
24	24	48.19	79.07	53.86	25.27	22.01
48	1	48.39	72.47	14.02	29.93	28.72
48	3	48.45	72.74	19.61	30.28	28.45
48	6	48.29	75.95	26.60	27.97	25.19
48	12	48.26	79.12	38.82	26.09	21.96
48	24	48.10	81.33	51.76	23.78	19.70
72	1	48.36	82.63	14.92	24.52	18.38
72	3	NA	NA	NA	NA	NA
72	6	NA	NA	NA	NA	NA
72	12	NA	NA	NA	NA	NA
72	24	NA	NA	NA	NA	NA

Table 4.8: Metrics from model: GRU

The training time appears to be consistent across different combinations of lookback and lookahead values, while varying across models with the SRNN being the slowest by a significant margin. The LSTM and GRU demonstrates similar training times, with GRU being slightly more performant. The DNN model is the fastest by a large margin.

For the DNN and SRNN model, the accuracy and F1 score is very consistent, and does not exhibit much variation with respect to lookback and lookahead. This applies to the confidence of the model as well.

The accuracy and F1 score values for the LSTM and GRU models vary greatly depending on the lookback and lookahead values. Generally, higher lookback values (e.g. 24 and 48) result in lower accuracy and F1 score values, while lower lookback values (e.g. 1 and 3) result in higher accuracy and F1 score values. This might indicate that considering a smaller number of previous time steps for prediction leads to better performance. Increasing the lookahead from 1 to 3 or 6 improves the accuracy and F1 score slightly, but the mean error also increases. This suggests that the model struggles to accurately predict lightning strikes further into the future.

The Mean Absolute Error and Wilson Score for the LSTM and GRU models values also follow a similar trend as accuracy and F1 score. Smaller lookback values result in lower mean absolute error, indicating better prediction accuracy. Increasing the lookahead seems to decrease the confidence of the model, but does not have as drastic effect as lookback.

Chapter 5

Discussion

5.1 Model Performance Across Time frames

The four different models has been evaluated and all demonstrates promising results. Each model exhibited high accuracy, with short-term lookback/lookahead combinations yielding an accuracy of 85-90% or higher, similar to the results concluded by Bao, Zhang, Ma, *et al.* [16]. It is worth noting that while all the models performed well overall, certain models excelled at specific combinations. This indicates that different models may be more suitable for different forecasting tasks, depending on the specific requirements and characteristics of the purpose.

Interestingly, it can be observed that the training time remained consistent across different combinations of lookback and lookahead values. Contrary to initial expectation, higher lookback values did not result in longer training times. This may be attributed to underlying optimizations in the underlying Keras library or the way batch training is implemented, alternatively that the difference in number of timesteps were insufficient enough to be observable.

Additionally, the lookahead value does not have a significant impact on the training time, as the amount of training data supplied to the models remained the same regardless of the lookahead value. The consistent training time across different combinations of lookback and lookahead values also suggests that the models can be trained efficiently on larger time frames without sacrificing computational performance.

The accuracy and F1 score values for the LSTM and GRU models vary greatly depending on the lookback and lookahead values. Generally, higher lookback values (e.g. 24 and 48) result in lower accuracy and F1 score values, while lower lookback values (e.g. 1 and 3) result in higher accuracy and F1 score values. This suggests that considering a smaller number of previous time steps for prediction leads to better performance, while a higher lookback confuses the model. Increasing the lookahead from 1 to 3 or 6 improves the accuracy and F1 score slightly, but the mean error also increases. This suggests that the model struggles to accurately predict lightning strikes further into the future. This is the expected behavior as weather data is inherently unstable. As time progresses, more variations will affect the weather parameters which will cause the result to deviate from the prediction. Increasing the lookahead is therefore expected to show an exponential decrease in accuracy.

The Mean Absolute Error and Wilson Score for the LSTM and GRU models values also follow a similar trend as accuracy and F1 score and remains relatively consistent, similar to the findings of Essa, Ajoodha, and Hunt [15]. Smaller lookback values result in lower mean absolute error, indicating better prediction accuracy. Increasing the lookahead seems to decrease the confidence of the model, but does not have as drastic effect as lookback.

For both the DNN and SRNN models, it is observed that the accuracy and F1 score remained consistent across different lookback and lookahead values. This consistency also extended to the confidence of the models.

This finding aligns with the previous observations that a lower lookback window and predicting based on short-term observations tend to result in better performance for the DNN model. It is important to note that the DNN model does not take time series into account and effectively uses a constant lookback of 1.

Similarly, the consistent performance of the SRNN model can be attributed to the well-known issue of the vanishing gradient problem in recurrent neural networks [4]. The vanishing gradient problem arises due to the nature of RNNs, where the factors associated with older time points become exponentially smaller during backpropagation, making older time points in the feature sequence less relevant. While the vanishing gradient problem is typically considered a drawback for SRNN models, in this case, it might be beneficial as longer lookback values tend to confuse the models. The SRNN model may place higher emphasis on the earliest values, resulting in more stable performance and higher confidence.

Overall, the findings suggest that using shorter lookback values and considering shorter feature sequences can lead to more effective models. Additionally, it can be observed that higher lookahead values negatively impacted the performance of LSTM and GRU models, while the performance of the DNN and SRNN models remained consistent. It is possible that the latter models may continue to exhibit similar accuracy even with higher lookahead values.

5.1.1 Lookback Deterioration

The obtained results are highly unexpected and would benefit from further investigation. Surprisingly, the LSTM and GRU models, which are specifically designed for time series prediction, exhibit better performance when not considering the time series aspect, compared to the DNN and SRNN models. Furthermore, the recurrent models are performing worse when actually taking time series into account. Multiple factors could contribute to this unexpected outcome:

Nature of the problem: A possible explanation is that the patterns indicative of a lightning strike may only emerge shortly before the actual occurrence of the strike. Incorporating older data might introduce extraneous patterns, thereby confusing the models.

Model complexity: The LSTM and GRU models are more complex and have a larger number of internal hyperparameters compared to the DNN and SRNN models. This increased complexity may allow the LSTM and GRU models to capture more intricate patterns and dependencies in the data, even without explicitly considering the time series nature of the data.

Data quality: The quality of the meteorological data used in the models may be insufficient. It is possible that the data lacks important features or contains noise, making it difficult for the models to extract meaningful information over time. This could lead to confusion and sub-optimal performance when the models attempt to incorporate the time series aspect.

Model architecture: The architecture of the LSTM and GRU models may not be well-suited for the specific characteristics of long-term meteorological data. It is possible that different hyperparameters or a different architecture combination, such as a convolutional neural network (CNN) or a transformer model, could better capture the spatial and temporal patterns present in the data.

5.2 Comparison of model types

Each model exhibits distinct characteristics and traits, and selecting the most suitable model depends on the specific use case and purpose.

When it comes to model confidence and prediction stability across time frames, the DNN and SRNN models are potential candidates. Both show high and consistent accuracy and F1 score regardless of lookahead value, and in the case of SRNN regardless of lookback. They do have some significant differences however, which may decide which one to use. The most prominent factor is the training time and computational requirements.

The DNN and SRNN model exhibits a mean training time of 8.76 and 130.01 seconds respectively, a 14.84x faster training time of the DNN model. However, the SRNN model do demonstrate a slightly higher accuracy and confidence, suggesting that if raw training performance is of high importance, the SRNN model may be more suitable at the cost of computation time.

The LSTM and GRU models shows a clear increase in performance for short lookback and lookahead values when compared to the DNN and SRNN models, reaching an unexpectedly high accuracy and F1 score of approximately of 91-93%. This performance suffers a sharp dropoff for longer lookback/lookahead values. The threshold for this dropoff is estimated to be somewhere around a lookback of 1+ and a lookahead of somewhere between 6-12+ hours. The performance of LSTM and GRU after this point is at best on par with the DNN and SRNN models, most likely performing worse.

Between the LSTM and GRU model, the GRU model is shown to be the most effective model, outperforming the LSTM model in all metrics. The GRU model shows both lower computational requirements, increased classification performance and increased confidence in its predictions.

Summarized, the selection of the model type depends on the specific use case. For real-time applications, the DNN model is preferred, while the SRNN model is better suitable for higher accuracy and confidence. The LSTM and GRU models excel in short lookback and lookahead scenarios, with the GRU model being the most effective of the two.

5.3 Optimal Features

The optimal features are likely to vary depending on the source of the data. This study was limited by the datasets publicly released by SMHI, meaning only a handful of weather parameters were able to be compared and tested. The PC analysis (refer section 4.2.3 on page 31) shows which parameters contributes with the highest variance, and thereby importance.

The proportion of variance between the PCs indicates that the parameters are relatively orthogonal and independent from each other, as 12 out of 15 PCs are required before reaching a 95% comprehensiveness. The PCs in table 4.4 reveals which parameters has the greatest impact on the overall variability of the dataset. PC1, while having the highest influence, is also dispersely influenced by its parameter loadings. It is primarily dependent on cloud-related factors such as *Low cloud cover*, *Medium cloud cover*, and *Fraction of significant* *clouds*, indicating that these parameters are of importance for predicting lightning strikes. PC2 exhibits similar behavior, with a focus on the base and cover of significant clouds, as well as high cloud cover. On the other hand, PC3 is more influenced by precipitation-related parameters such as *Precipitation* and *Frozen part of precipitation*. Amongst the less important parameters seems to be *snowfall*, *temperature* and wind-related parameters, meaning these should be dropped if model performance is of priority in sacrifice of accuracy.

5.4 Optimal Hyperparameters

The hyperparameters obtained from the genetic algorithm are presented in section 4.3. However, due to the limited number of iterations, the results are likely to be suboptimal and can be significantly improved. It is recommended to further investigate this topic in future research.

Additionally, these hyperparameters may not be the most efficient in every scenario. They should be re-evaluated if changes are made to the dataset or if different model types or lookback/lookahead combinations are used. The genetic algorithm evaluated LSTM models only. As such it is possible that the hyperparameters are sub-optimal for the other model types.

One unexpected finding however was the absence of dense layers. The genetic algorithm determined that the recurrent layers alone were sufficient to capture the temporal changes in the input data and learn complex patterns. This suggests that the dense layers may not provide significant additional information in this particular context.

The choice of activation functions also played an important role in achieving good performance. The tanh activation function was found to be effective for the recurrent layers, while the relu activation function was found to be effective for the dense layers.

The use of dropout regularization for both the recurrent and dense layers seems to have helped prevent overfitting. A dropout rate of 0.2 was found to be optimal, indicating that dropping out 20% of the inputs during training strikes a balance between preventing overfitting and retaining important information.

The number of recurrent layers and the number of units in these layers were also important factors in achieving good performance. The genetic algorithm determined that two recurrent layers with 256 units each were optimal for the given dataset. This suggests that having multiple recurrent layers and a sufficient number of units in each layer allows for better capturing of the temporal changes in the input data.

5.5 Ethical Considerations & Sustainability

The primary ethical consideration of this study is the transparency and interpretability of the models. Deep learning models are often criticized for being "black boxes", making it difficult to understand how they arrive at their predictions. This lack of transparency can hinder trust and acceptance among stakeholders, including meteorologists, emergency responders, and the general public. Efforts should be made to develop methods for interpreting and explaining the model's predictions, thereby enhancing transparency and accountability.

From an environmental perspective, the computational resources required for training and deploying deep learning models can be substantial. Training deep learning models, particularly those with complex architectures like LSTM and GRU, can consume significant amounts of energy, contributing to carbon emissions. To address this, it is important to optimize the models for computational efficiency and explore the use of energy-efficient hardware and cloud computing resources powered by renewable energy sources.

On the societal front, accurate lightning strike prediction can significantly contribute to disaster preparedness and risk mitigation, thereby enhancing community resilience. Early warnings of lightning strikes can save lives, reduce property damage, and minimize disruptions to critical infrastructure. This aligns with the broader goals of sustainable development by promoting safety, well-being, and economic stability.

Balancing these factors and compare the models to current methods for lightning strike prediction is the key to achieve an ethical, sustainable and proper method of lightning prediction.

5.6 Broader Interpretation

The findings of this study may have important implications for the development and application of DL models for lightning forecasting. The high accuracy and performance of the models indicate that deep learning has the potential to be a valuable tool in this domain. However, it is important to consider the limitations and challenges associated with these models.

One of the key insights from this study is the importance of selecting the appropriate lookback and lookahead values for the models. The results show that shorter lookback values and smaller feature sequences tend to result in better performance. This suggests that using a smaller number of previous time steps for prediction leads to more accurate results, and perhaps even shorter than hourly frequency may prove advantageous. Additionally, increasing the lookahead value beyond a certain threshold negatively impacts the performance of the models. This is likely due to the inherent uncertainty and variability of weather data, which might make it difficult to predict lightning strikes far into the future.

Another important finding is the variation in performance across different model types. The LSTM and GRU models, which are specifically designed for time series prediction, exhibit better performance when not considering the time series aspect. One reason for this unexpected result is that the complexity of these models allows them to capture more complex patterns and dependencies in the data, even without explicitly considering the time series nature. On the other hand, the DNN model which does not explicitly consider time series, exhibit consistent performance across different lookback and lookahead values. This suggests that this model may be more suitable for real-time applications or situations where computational efficiency is a priority.

The findings also highlight the importance of data quality and availability. The performance of the models is highly dependent on the quality and quantity of the meteorological data used for training. Insufficient or noisy data can lead to sub-optimal performance and inaccurate predictions. Therefore, it is of importance to ensure that the data used for training is of high quality and covers a broad range of geographic locations and time periods. This will enhance the generalization and transferability of the models, allowing them to make accurate predictions in new and unseen environments.

Lastly, the challenge of interpretability in deep learning models is an important consideration. The complex nature of neural networks makes it difficult to understand the factors that contribute to their predictions. Addressing this challenge is an active area of research and will allow their adoption into practical applications.

Chapter 6

Conclusions and Future Work

6.1 Overview

This study has achieved its goal of addressing the challenge of accurately and timely predicting lightning strikes, an important task with the potential to save lives and prevent significant economic losses worldwide.

The motivation for this research stems from the advancements in machine learning (ML) and deep learning (DL) models. Recent developments in neural networks have demonstrated that complex tasks can be accomplished using limited hardware, given the availability of sufficient data. In comparison to the extensive scale and complexity of numerical weather simulations, DL offers a more efficient approach to forecasting. Additionally, DL models are lightweight, portable, faster, and potentially more accurate.

The application of DL models in this context has the potential to solve various problems. Apart from contributing to the field of artificial intelligence and machine learning through further research, these models can solve challenges such as:

- improved timely warning systems for lightning strikes
- aiding with the development of strategies for resource allocation
- mitigation of environmental impacts caused by storms and lightning strikes

The project was started with the objective of exploring the latest advancements in artificial intelligence and determining how to construct effective deep learning models for predicting lightning strikes across different time frames. To achieve this, data provided by the Swedish Meteorological and Hydrological Institute (SMHI) was collected and utilized. The Lightning Archive dataset, which contains comprehensive information on all lightning strikes in Sweden since January 2, 2012, was used, along with the MESAN (AROME) dataset, which serves as a repository of the meteorological data required for predictive modeling. Both datasets underwent extensive pre-processing, including methods such as filtering, balancing, binning, extraction, and imputation.

Four deep learning models were evaluated in this study: Dense Neural Networks (DNN), Simple Recurrent Neural Networks (SRNN), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) models. The performance of these models was assessed using stratified k-fold cross-validation, a method in which data classes are equally split into a number of train/test partitions and measured across iterations. Various metrics, such as Training Time, F1 Score, Mean Absolute Error and Wilson Score were collected for each model.

These metrics were evaluated across combinations of two additional evaluation factors: lookback and lookahead. The lookback refers to the size of the temporal dimension of each input (i.e. the length of each feature sequence) while the lookahead represents the size of the window within which the model is expected to make predictions.

The hyperparameters for the models were determined using a genetic or evolutionary algorithm. A population of randomly generated combinations of hyperparameters from a predetermined set was created, and each combination was assigned a fitness score based on its efficiency. The next generation of candidates was then created based on the previous generation, with a higher emphasis on those with higher fitness scores. After three generations, the optimal parameters were found to be an architecture consisting of two recurrent layers with a size of 256 units.

Continuing with the data analysis, an exploratory analysis was conducted on the lightning dataset. This analysis played an important role in determining the pre-processing techniques for the lightning data. One important finding was that lightning strikes tend to occur in close proximity both spatially and temporally, highlighting the significance of the binning process.

Similarly, an analysis was performed on the MESAN dataset. It was observed that the dataset contained multiple missing values, with some parameters completely missing. These missing values were considered invalid and were either imputed using the closest known value or dropped entirely.

To identify the optimal features for predicting lightning strikes, two methods were employed. First, a covariance matrix was used to illustrate the correlation between the available parameters. This matrix provided a simple and easily interpretable representation of the relationships between the parameters. Additionally, a Principal Component Analysis (PCA) was conducted to identify the parameters with the highest variance. Parameters with higher variance were preferred as they ensured the input data remained as independent as possible. This was expected to not only increased the accuracy of the models but also improved computational efficiency.

The covariance matrix revealed a high correlation between certain parameters, such as `wet-bulb temperature`, `cloud base of significant clouds above sea`, `total cloud cover`, and `wind gust`. As a result, these parameters were dropped in favor of more independent features. Furthermore, the PCA analysis indicated that cloud-related parameters, such as cloud cover at various heights and the fraction of significant clouds, were the most useful for predicting lightning strikes. Precipitation-related parameters also showed significant importance. On the other hand, parameters related to temperature and wind were found to be less influential.

The results of the study are promising, showing that DL models are capable of differentiating instances with high likelihood of lightning strike. While all model types demonstrated an overall accuracy of above 75%, some models excelled at specific combinations of lookback and lookahead.

In general, two types of behavior were witnessed. The performance of the LSTM and GRU models varied greatly based on the lookback and lookahead combination, with decreasing performance as either value increased. Compared to the DNN and SRNN models they showed higher accuracy when considering shorter time frames, but experienced a sharp dropoff once the lookback is greater than 1 hour and lookahead is in between or above 6-12 hours. Between the LSTM and GRU models, the GRU model demonstrates higher performance across all metrics.

The second type of behavior were shown by the DNN and SRNN models. These two models measured a close to constant, extremely stable performance regardless of the lookback/lookahead combination used, suggesting that they may be able to score high when predicting even further into the future. Compared to each other, the SRNN models demonstrated a higher performance in general by a small margin, at the expense of significantly longer training time.

The training times were shown to differentiate highly between the model types, with the DNN model being the fastest one by far. This was followed by the GRU and LSTM models, of which the GRU models were slightly more performant. Lastly, the SRNN model was the slowest, being close to 15 times slower then the DNN model.

6.2 Challenges and Limitations Encountered

6.2.1 Generalization and Transferability

Producing a model that can generalize and work effectively across various environments is an important aspect to consider in the prediction of lightning strikes. The ability to have a single model that can accurately predict lightning strikes regardless of the environment, geographic location, and time can greatly streamline the distribution of predictions by having a centralized, unified model.

Traditionally, larger and more complex algorithms and systems tend to be less efficient at specific tasks. However, deep learning models have shown the opposite behavior, as they are able to increase their specific abilities and improve performance by generalizing [38]. This means that deep learning models might have the potential to perform well across different environments.

To ensure that the models are effectively generalizing, it is important to train them on larger datasets that cover different geographic locations. Using diverse data from various regions, the models can learn to capture the common patterns and characteristics of lightning strikes, regardless of the specific location. This will enhance the transferability of the models and enable them to make accurate predictions in new and unseen environments.

6.2.2 Data Availability and Processing

One of the primary challenges in developing DL models for lightning strike prediction is the availability and processing of data. Although the computational requirements pail in comparison to numeric models currently in use, obtaining sufficient and high-quality data is vital for training accurate models. However, the types and number of data parameters to include, as well as the standardization of units and error corrections, may vary depending on the geographic location and the source of the acquired datasets.

The availability and standardization of data can be limited, especially in certain regions or for specific time periods. This scarcity of data can pose challenges in training models for variety and generalization effectively. To address this issue, future works could explore alternative methods, such as creating a larger number of smaller, local and more specialized models. Each model could be trained to predict lightning strikes in specific geographic locations using the data available, leveraging the available data for those regions. However, this approach may introduce difficulties in examining the effectiveness of different parameters, as the datasets used for training would vary.

6.2.3 Interpretability and Explainability

Deep learning models are often referred to as "black boxes" due to their complex and meticulous nature. They are known for being difficult to interpret and understand through simple inspection. This lack of interpretability and explainability can be a limitation when it comes to gaining insights into the inner workings of the models and understanding the factors that contribute to their predictions.

Interpretability and explainability are important aspects, especially in critical applications. Stakeholders, including meteorologists and emergency response teams, need to have a clear understanding of how the models arrive at their predictions to help build trust in the models and enable better decision-making based on the predictions.

Addressing the challenge of interpretation in deep learning models is an active area of research. Various techniques, such as feature visualization, attribution methods, and model-agnostic approaches, are currently being explored. By developing methods to interpret and explain the predictions of DL models, it is possible to enhance their transparency and facilitate their adoption in real-world applications.

6.3 Improvements and Future Work

6.3.1 Dataset Selection and Pre-processing

The size of the dataset should be further explored. Experimenting with different sizes of the data can provide insights into the optimal amount of data required for accurate predictions. It is important to strike a balance between having enough data to capture the underlying patterns and avoiding overfitting the model. Additionally, the choice of features is critical. Exploring different combinations of features can help identify the most relevant features for the prediction task. It is also worth considering whether to include derived or compound parameters, as they may provide additional information that can improve the classification accuracy and model confidence.

Furthermore, reformatting the parameters can be beneficial. For example, transforming parameter such as temperature or pressure to percentual changes can help capture the relative variations in the data, which may be more informative for the prediction models.

6.3.2 Evaluating Data Across Different Environments and Topologies

Lightning strikes can vary significantly depending on the geographic location and the surrounding environment. Factors such as temperature, humidity, elevation, and topography can all influence the occurrence and characteristics of lightning strikes. Therefore, including data from diverse environments ensures that the models can effectively capture the underlying patterns and make accurate predictions in different settings. This can help improve the models' ability to generalize and make accurate predictions in new and unseen locations.

Furthermore, including the topological data from the region of focus also provide valuable insights. Different topologies, such as coastal areas, mountainous regions, or urban environments, can have unique characteristics that affect the occurrence and behavior of lightning strikes. By including environment data as features, the models can learn to adapt to these specific conditions and make more accurate predictions in similar settings.

6.3.3 Model Architecture and Hyperparameter Tuning

The choice of model architecture and hyperparameter tuning are important aspects to consider in improving the lightning prediction models. Different deep learning architectures, such as convolutional neural networks or recurrent neural networks can be explored to capture the spatial and temporal dependencies in the data. It would be beneficial to compare the performance of different architectures and select the one that yields the best results.

Hyperparameter tuning is another crucial step in model development. Fine-tuning the hyperparameters, such as learning rate, batch size, and regularization techniques, can significantly impact the model's performance. The genetic process used in this study could run for longer periods, or alternative approaches may be explored such as Bayesian or grid search.

Moreover, ensembling techniques can be employed to combine multiple models and improve the prediction accuracy. Techniques like bagging or boosting can be explored to leverage the diversity of multiple models and reduce the variance in predictions.

6.3.4 Balancing the Dataset with Similar Negative Samples

The current dataset used for training the storm and lightning prediction models consists of 50% positive samples representing lightning strikes and 50% randomly generated negative samples. However, this approach may not accurately reflect the real-world distribution of lightning occurrences, which are typically negatively skewed. In reality, there are far more samples where no lightning strike occurs compared to samples where it does.

The imbalance in the dataset likely pose challenges for the models. Randomly sampling negative samples might have resulted in negative examples that are significantly different from the positive samples. As a result, the models can easily differentiate between the positive and negative samples, leading to high accuracy but limited generalization ability in environments where lightning strike prediction is practically used. In such environments, the positive feature sequences may closely resemble the negative sequences, making it difficult for the models to accurately predict lightning strikes.

To address this issue, it is important to include more specific negative samples that are similar to the positive samples. By incorporating negative samples that closely resemble the positive samples, the models can learn to differentiate between the two more effectively. This can help improve the models' ability to generalize and make accurate predictions in environments where the feature sequences for positive and negative samples are highly similar.

In conclusion, this study has successfully addressed the challenge of predicting lightning strikes using deep learning models. The results demonstrate that deep learning models can accurately identify samples of lightning strikes based on meteorological data with overall accuracies surpassing 75%. The performance of the models varies based on the combination of lookback and lookahead used, with the LSTM and GRU models being more efficient when considering shorter time frames, and the DNN and SRNN models being more performant and stable overall.

This study may contribute to the field of artificial intelligence and machine learning by demonstrating the effectiveness of deep learning models for predicting lightning strikes, as well as providing a baseline for input features and hyperparameters. Future work can focus on fine-tuning the models, incorporating more data, and evaluating their performance in real-world scenarios.

References

[1] P. Bauer, A. Thorpe, and G. Brunet, "The quiet revolution of numerical weather prediction," 2015. DOI: 10.1175/WAF-D-17-0010.1.

[2] X. Ren, X. Li, K. Ren, *et al.*, "Deep learning-based weather prediction: A survey," 2020. DOI: 10.1016/j.bdr.2020.100178.

[3] P. Hewage, A. Behera, M. Trovati, *et al.*, "Temporal convolutional neural (tcn) network for an effective weather forecasting using time-series data from the local weather station," 2020. DOI: 10.1007/s00500-020-04954-0.

[4] Hochreiter and Sepp, "Untersuchungen zu dynamischen neuronalen netzen," 1991.

[5] M. H. Sulaiman, A. I. Mohamed, and Z. Mustaffa, "An application of deep learning for lightning prediction in east coast malaysia," 2023. DOI: 10.1016/j.prime.2023.100340.

[6] Y. Essa, H. G. Hunt, and R. Ajoodha, "Short-term prediction of lightning in southern africa using autoreggressive machine learning techniques," 2021 IEEE International IOT Electronics and Mechatronics Conference, 2021. DOI: 10.1109/IEMTRONICS52119.2021.9422493.

[7] M. G. Schultz, C. Betancourt, B. Gong, *et al.*, "Can deep learning beat numerical weather prediction?," 2021. DOI: 10.1098/rsta.2020.0097.

[8] X. Wang, K. Hu, Y. Wu, and W. Zhou, "A survey of deep learning-based lightning prediction," 2023. DOI: 10.3390/atmos14111698.

[9] L. Y, H. P, B. L, and B. Y, "Object recognition with gradient-based learning," 1999. DOI: 10.1007/3-540-46805-6_19.

[10] H. S and S. J, "Long short-term memory," 1997. DOI: 10.1162/neco.1997.9.8.1735.

[11] G. FA and S. J, "Recurrent nets that time and count," Joint Conference on Neural Networks, 2000. DOI: 10.1109/IJCNN.2000.861302.

[12] C. J, G. Ç, C. K, and B. Y, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014. [Online]. Available: http://arxiv.org/abs/1412.3555.

[13] K. DP and W. M, "Auto-encoding variational bayes," 2014. [Online]. Available: http://arxiv.org/abs/1312.6114.

[14] G. I, P.-A. J, M. M, *et al.*, "Generative adversarial nets," 2014.

[15] Y. Essa, R. Ajoodha, and H. G. Hunt, "A lstm recurrent neural network for lightning flash prediction within southern africa using historical time-series data," 2021. DOI: 10.1109/CSDE50874.2020.9411544.

[16] R. Bao, Y. Zhang, B. J. Ma, Z. Zhang, and Z. He, "An artificial neural network for lightning prediction based on atmospheric electric field observations," 2022. DOI: 10.3390/rs14174131.

[17] A. Verikas and M. Bacauskiene, "Feature selection with neural networks," 2002. DOI: 10.1016/S0167-8655(02)00081-8.

[18] T. Howley, M. G. Madden, M.-L. O'Connell, and A. G. Ryder, "The effect of principal component analysis on machine learning accuracy with high dimensional spectral data," 2006. DOI: 10.1007/1-84628-224-1_16.

[19] H. Alibrahim and S. A. Ludwig, "Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization," 2021 IEEE Congress on

Evolutionary Computation (CEC), Jul. 21, 2021. DOI: 10.1109/CEC45853.2021.9504761.

[20] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," 2020. DOI: 10.48550/arXiv.2003.05689.

[21] N. Mehdiyeva, D. Enkec, P. Fettkea, and P. Loosa, "Evaluating forecasting methods by considering different accuracy measures," Complex Adaptive Systems, Publication 6, 2016.

[22] "About smhi." (Sep. 10, 2023), [Online]. Available: https://www.smhi.se/en/about-smhi.

[23] C. Renggli, L. Rimanic, N. M. Gurel, *et al.*, "A data quality-driven view of mlops," 2021. DOI: 10.48550/arXiv.2102.07750.

[24] "Smhi lightning archive." (Sep. 10, 2023), [Online]. Available: https://www.smhi.se/data/utforskaren-oppna-data/blixtdata-historiska-arkivdata.

[25] K. Pearson, "X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling," 1900. DOI: 10.1080/14786440009463897.

[26] "Smhi historical meteorological data." (Sep. 11, 2023), [Online]. Available: https://www.smhi.se/data/utforskaren-oppna-data/meteorologisk-analysmodell-mesan-arome-historiska-analysdata.

[27] "Smhi grib format." (Sep. 11, 2023), [Online]. Available: https://www.smhi.se/data/oppna-data/information-om-oppna-data/grib-format-1.30761.

[28] T. T. Warner, R. A. Peterson, and R. E. Treadon, "A tutorial on lateral boundary conditions as a basic and potentially serious limitation to regional numerical weather prediction," 1997-02-10.

[29] W. Lee and K. Seo, "Downsampling for binary classification with a highly imbalanced dataset using active learning," 2022. DOI: 10.1016/j.bdr.2022.100314.

[30] L. Medsker and L. Jain, *Recurrent Neural Networks: Design and Applications*. CRC Press, 2001.

[31] A. Tealab, "Time series forecasting using artificial neural networks methodologies: A systematic review," 2018. DOI: 10.1016/j.fcij.2018.10.003.

[32] fdeloche. "Recurrent neural network unfold." Accessed 2024-07-10. (Jun. 19, 2017), [Online]. Available: https://en.wikipedia.org/wiki/File:Recurrent_neural_network_unfold.svg.

[33] K. Cho, B. van Merrienboer, C. Gulcehre, *et al.*, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," Sep. 3, 2014. DOI: 10.48550/arXiv.1406.1078.

[34] Y. Su and C. J. Kuo, "On extended long short-term memory and dependent bidirectional recurrent neural network," Apr. 25, 2019. DOI: 10.1016/j.neucom.2019.04.044.

[35] fdeloche. "Gated recurrent unit." Accessed 2024-07-10. (Jun. 28, 2017), [Online]. Available: https://en.wikipedia.org/wiki/File:Gated_Recurrent_Unit.svg.

[36] fdeloche. "Long short-term memory." Accessed 2024-07-10. (Jun. 20, 2017), [Online]. Available: https://en.wikipedia.org/wiki/File:Long_Short-Term_Memory.svg.

[37] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," 1965. DOI: 10.1093/biomet/52.3-4.591.

[38] K. Wiggers. "Openai open-sources whisper, a multilingual speech recognition system." (Feb. 12, 2023), [Online]. Available: https://techcrunch.com/2022/09/21/openai-open-sources-whisper-a-multilingual-speech-recognition-system/.

Glossary

AI: Artificial Intelligence

ARIMA: AutoRegressive Integrated Moving Average

CNN: Convolutional Neural Network

DL: Deep Learning

DNN: Dense Neural Network

GRIB: Gridded Binary

GRU: Gated Recurrent Unit

LSTM: Long Short Term Memory

LIGHT: Lightning Archive Dataset

MAE: Mean Absolute Error

MAPE: Mean Absolute Percentage Error

MESAN: Meteorological Analysis Model Data Dataset

MIMO: Multiple-Input Multiple-Output

MISO: Multiple-Input Single-Output

ML: Machine Learning

PC: Principal Component

PCA: Principal Component Analysis

RNN: Recurrent Neural Network

SMHI: Swedish Meteorological and Hydrological Institute

SRNN: Simple Recurrent Neural Network

TCN: Temporal Convolutional Neural

UALF: Universal ASCII Lightning Format

VAE: Variational Autoencoder

WS: Wilson Score

Faculty of Computing. Blekinge Institute of Technology. 371 79 Karlskrona, Sweden.