A DEEP LEARNING MODEL TO PREDICT THUNDERSTORMS WITHIN $400km^2$ SOUTH TEXAS DOMAINS

A Thesis

by

HAMID KAMANGIR

MS, University of Isfahan, 2016

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

December 2019

December 2019

A DEEP LEARNING MODEL TO PREDICT THUNDERSTORMS WITHIN $400 km^2$ SOUTH TEXAS DOMAINS

A Thesis

by

HAMID KAMANGIR

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Dr. Scott King, PhD
Chair

Dr. Philippe Tissot, PhD
Co-Chair/Committee Member

Dr. Longzhuang Li, PhD
Committee Member

December 2019

ABSTRACT

High resolution predictions, both temporally and spatially, remain a challenge for the prediction of thunderstorms and related impacts such as lightning strikes. The goal of this work is to improve and extend a machine learning method to predict thunderstorms at a 3*km* resolution with lead times of up to 15 hours. A deep learning neural networks (DLNN) was developed to post process deterministic High-Resolution Rapid Refresh (HRRR) numerical weather prediction (NWP) model output to develop DLNN thunderstorm prediction models (categorical and/or probabilistic output) with performance exceeding that of the HRRR and other models currently available to National Weather Service (NWS) operational forecasters. Notwithstanding the discovery that shallow neural network models can approximate any continuous function (provided the number of hidden layer neurons is sufficient), studies have demonstrated that DLNN models based on representation learning can perform superior to shallow models with respect to weather and air quality predictions. In particular, we use the method known as stacked autoencoder representation learning, yet more specifically, greedy layer-wise unsupervised pretraining. The training domain is slightly large specific area in Corpus Christi, Texas (CRP). The domain is separated into a grid of 13×22 equidistant points with a grid spacing of 20 km. These points serve as boundaries/centres for 286 20×20 km (400 km2) square regions. The strategy is that DLNN model train on whole boxes and test on three most important boxes to evaluate the model. The target refers to the existence, or non-existence, of thunderstorms (categorical). Cloud to Ground (CG) lightning was chosen as the proxy for thunderstorm occurrence. Logistic regression was then applied to SDAE output to train the predictive model. An iterative technique was used to determine the optimal SDAE architecture. The performance of the optimized DLNN classifiers exceeded that of the corresponding shallow neural network models developed by Collins and Tissot [12], a classifier via a combination of principal component analysis and logistic regression, and operational weather forecasters, based on the same dataset.

DEDICATION

This thesis is dedicated to my parents for their love, endless support and encouragement.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Scott King for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis and throughout my graduation.

Besides my advisor, I would like to thank of Dr. Tissot, for his encouragement, insightful comments, and suggestions. You have set an example of excellence as a researcher, mentor, instructor, and role model.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES PAGE

LIST OF TABLES

TABLES PAGE

CHAPTER I: INTRODUCTION

1.1 Motivation

A thunderstorm or convective storm is a cumulonimbus cloud that produces the electric discharge known as lightning (which produces thunder) and typically generates heavy rainfall, gusty surface wind, and possibly hail [11]. The terms thunderstorm, convective storm, and convection are used interchangeably in this chapter to refer to thunderstorms. Thunderstorms adversely affect humans and infrastructure. An estimated 24,000 deaths and 240,000 injuries worldwide are attributable to lightning [33, 34]. In the USA, lightning is the third leading cause of storm-related deaths based on averages during the period 1985-2014 [11]. Additional hazards that can occur include large hail, flash flooding associated with heavy rainfall, and damage from wind from tornadoes and/or non-rotational (straight-line) wind. During the period 1980 – 2014, 70 severe thunderstorm events each totaling $\geq$ 1 billion US dollar damage occurred in the USA, which totaled to 156.3 billion US dollars (adjusted for inflation to 2014 dollars) [11]. Further, convection exacts an economic cost on aviation in terms of delays [101]. Given the adverse socioeconomic impact associated with thunderstorms, there is motivation to predict thunderstorm occurrence and location to inform the public with sufficient lead time. However, the complexity of thunderstorm generation (hereafter convective initiation, or CI), given the myriad of processes (operating on different scales) that influence the vertical thermodynamic structure of the atmosphere (that directly influences the thunderstorm development) and the CI itself [89], the characteristic Eulerian (with respect to a fixed point at the surface) time and linear space scales of individual thunderstorms [68], and the inherent predictability limitations of atmospheric phenomena on the scale of individual thunderstorms [56], renders the skillful prediction of thunderstorm occurrence, timing, and location very difficult.

1.2 Related Work

There are four main techniques for thunderstorm prediction: (1) numerical weather prediction (NWP) models, (2) post-processing of NWP model ensemble output, (3) the post-processing of single deterministic NWP model output via statistical and artificial intelligence (AI)/machine learning (ML), and (4) classical statistical, AI/ML techniques.

Predicting thunderstorm by NWP models are using the atmospheric variables/parameters which directly influence on thunderstorm development. Such models are based on determinism concept means future events are influenced by past condition, means predicting thunderstorm is based on initial condition of atmosphere. In fact, limitations of NWP models include limitations of predictability of a chaotic atmosphere and this inherent model error growth [11]. NWP models based on ensembles techniques attempt to consider the uncertainty in the NWP model initial condition which is the main reason of predictability limitations in NWP models [53]. The post-processing of NWP model ensembles can generate useful probabilistic thunderstorm output. However, there is more computational cost for ensemble NWP models compared to single NWP model.

On other hand, machine learning approaches involves a much lower computational cost relative to model ensembles. Our objective is to more accurately and skillfully predict the occurrence, time, and location of thunderstorms up to 15 hours in advance, on the horizontal space scale of a single convective storm post-processing of single deterministic NWP model output via deep learning models [11].

1.3 Purpose and Contribution

Our strategy involves the post-processing of deterministic NWP model output to develop deep learning neural network (DLNN) models to predict thunderstorms. We seek to improve accuracy and skill further via DLNN models. The method is known as representation learning, yet more specifically, greedy layer-wise unsupervised pretraining, whereby unsupervised learning occurs across ≥ 2 hidden layers (one layer at a time) as pre-training step (to create an increasingly higher

order representation of the input data) followed by the hidden layer with the highest order representation serving as input into a supervised learning algorithm [25]. Notwithstanding the discovery that shallow neural network models can approximate any continuous function (provided the number of hidden layer neurons is sufficient), studies have demonstrated that DLNN models based on representation learning can perform superior to shallow models with respect to weather and air quality predictions [28, 35, 54].

In summary, this work makes the following contributions:

1. Develop a appropriate deep learning model as a post-process of single deterministic NWP model for thunderstorm prediction.

2. Compare the deep learning model with a shallow neural network to assess the influence of deep layers and greedy-layer wise learning on the performance.

3. Compare the stacked autoencoder model as nonlinear feature reduction with principal component analysis as a linear feature reduction technique to show the ability of deep learning models to model the nonlineatrity between predictors and prediction.

4. Analyse the latent features generated by SDAE model and PCA to show the difference between the features and why SDAE is better.

1.4 Outline

The following chapter 2 will address the theoretical background concerning this work, focusing on different model used for thunderstorm lightning prediction. After describing the taken methodology in chapter 3, the empirical data and the respective results of the experiments will be shown in chapter 4. Chapter 5 will discuss the results, critically review the taken approaches and methods as well as examine the validity and reliability of presented results. Finally, chapter 6 will summarize the work and give an outlook for possible future research and expansion on this topic.

CHAPTER II: Theoretical Background: Thunderstorm prediction methods

In this section, different techniques used for thunderstorm prediction by researchers and the advantageous and disadvantageous of them will be considered.

2.1 Numerical Weather Prediction (NWP) models

NWP models are based on the concept of determinism which explains that future states of a system influenced by earlier states relied on physical laws [11]. The atmospheric motion is described by a set of nonlinear equations include of partial differential conservation, the continuity equation, the equation of state and etc by meteorologist [43, 87] which can be solved mainly numerically. As discussed before, the NWP model prediction has an initial value problem. Data assimilation process is used to provide the requisite initial value for NWP model. In practice, data assimilation technique uses the balanced short-term output from an earlier NWP model run in combination of meteorological observations to serve as an initial condition for the NWP model [87]. The primary output variables include temperature, wind, pressure/height, mixing ratio, and precipitation are calculated in next step. A post-processing step is performed which includes the calculation of secondary variables/parameters (CAPE, relative humidity, etc.) and the development of techniques to remove model biases[43, 87].

Despite the utility of NWP models, there exist fundamental limitations. The NWP is dependent on the system initial condition of atmosphere and clearly the atmosphere is chaotic and hard to predict[56]. Thus, minor errors between the initial atmospheric state and the NWP model representation of the initial state can result in a big mistake on future NWP prediction. Unfortunately, by using the state-of-the-art NWP models is hard to have a a true, exact, and complete representation of the initial state of the atmosphere. Even with having the perfect reforestation of the initial atmospheric state by the NWP model, still the errors associated with imperfections inherent in model formulation and time integration is problematic. Also, accurately predicting the exact time

and location of thunderstorm in high-resolution ($3-6hr$) by using the NWP model is difficult[98].

2.2 Post processing of NWP model ensembles

Another strategy to predict lightning is based on ensemble concept, which is a Monte Carlo approximation to stochastic dynamical forecasting [95]. The modeler then conducts an NWP model run on each member of the ensemble, hence the term ensemble forecasting [95]. In practice, each member of the ensemble producing a unique combination of model initial condition, dynamics, and/or physics [85]. The advantage of using ensemble forecasting is assessing the level of forecast uncertainty over prediction with single deterministic NWP output. The ensemble model can be used as post-processing for probability prediction. The NWS Environmental Modeling Cen- ter developed a ensemble model to thunderstorm forecasting as the short-range ensemble forecast (SREF), a collection of selected output from an ensemble of 21 mesoscale (16-km) NWP model runs. One limitation of NWP ensembles to support operational forecasters is the tremendous computational cost necessary to run since each ensemble member is a separate NWP model run. Another limitation is the realization that the true PD of the initial condition uncertainty is unknown and changes daily [95] .

2.3 Post-processing of NWP model output using statistical and machine learning techniques

Statistical and machine learning methods can be utilized to post-process NWP output to compensate the uncertainty of NWP model biases and to quantify the level of uncertainty in single NWP deterministic output. Statistical post-processing methods include model output statistics (MOS) [24] and logistic regression [95]. MOS involves the development of data-driven models to predict the future state of a target based on a data set of past NWP output (features/predictors) and the corresponding target/predictand.

Specific AI/ML techniques involving the post-processing of NWP output include expert systems [64], adaptive boosting [71], artificial neural networks [60, 12], and random forests [63]. A random forest [6] a classifier resulting from an ensemble/forest of tree-structured classifiers,

whereby each tree is developed from independent random subsamples of the data set (including a random selection of features from which the optimum individual tree predictors are selected) drawn from the original training set.

2.4 Classical statistical and artificial intelligence/machine learning techniques

Statistical and machine learning models not using the NWP model output that have been used with previous categories.

In one study, the utility of both a graphical method and multiple regression was tested to predict thunderstorms [73]. The multiple regression technique involved the stepwise screening of 274 potential predictors to 9 remaining. Both prediction models provided thunderstorm predictions in probabilistic terms. Objective techniques were applied to convert probabilistic predictions to binary predictions for the purpose of verification. An expert system was developed using the decision tree method to forecast the development of thunderstorms and severe thunderstorms [13]; the tree was based on physical reasoning using the observation of meteorological parameters considered essential for convective development. An expert system named Thunderstorm Intelligence Prediction System (TIPS) was developed to predict thunderstorm occurrence [52].

The artificial neural network (ANN) has been used to predict thunderstorms. Thermody- namic data from Udine Italy rawinsondes, surface observations, and lightning data were used to train/optimize an ANN model to predict thunderstorms 6 h in advance over a 5000-km2 domain in the Friuli Venezia Giulia region [59]. An ANN model was developed (also using thermodynamic data) to predict severe thunderstorms over Kolkata India during the pre-monsoon season (April-May) [10]. Logistic regression was used to develop a binary classifier to predict thunderstorms 6-12 h in advance within a 6825 km2 region in Spain [78]. A limitation of classical statistics to weather prediction is that utility is bi-modal in time;confined to very short time periods (less than a few hours) or long time periods (10 days) [95].

We seek a different strategy/paradigm to account for chaos in atmospheric prediction and errors in NWP output. Rather than post-process a set of unique NWP ensemble runs to predict the

future atmospheric state and/or atmospheric phenomena, we post-process NWP output from single deterministic runs by training and optimizing a model using ML, a strategy similar to that used by Collins and Tissot [11] wherein select NWP model output served as features to train and op- timize shallow neural network models to predict thunderstorms. Unlike Collins and Tissot ([11, 12]), we use the DLNN as the ML technique instead of shallow neural networks. Pathak et al, [70] demonstrated the utility of predicting the future state of chaotic dynamical systems by combining a dynamical model that describes the system with ML. Further, given that an ensemble of NWP runs is obviously computationally expensive relative to single deterministic runs, a prediction of thunderstorm occurrence by post-processing secondary output parameters from a single deterministic NWP model integration with an accuracy/skill comparable to that of a model ensemble system would clearly possess high utility. A motivation for using deep learning to predict thunderstorms is the apparent performance enhancement of deep learning models relative to the shallow variety with respect to weather and air quality predictions ([28, 35, 54, 88]), These deep learning models are based on representation learning [25], whereby unsupervised learning occurs across $\geq$ 2 hidden layers as pre-training step in order to generate an increasingly higher order representation of the input features, with the final hidden layer (with the highest order representation) serving as input into a supervised learning algorithm.

Thanks to the increasing availability of large data sets and affordable computational power deep learning algorithms can now model complex non linear relationships in the earth sciences([28, 22, 49, 80, 74]). The introduction of DLNN's in 2006 [29] has led to large changes in Artificial Intelligence (AI) and Machine Learning (ML) in many areas of research. Deep learning algorithms aim at learning high-level feature representations to solve complex problems while establishing relationships between problem predictors and predictands. DLNN has shown the ability to generate higher performance models as compared to traditional machine learning. The specific representation learning based algorithm used in this study is the Stacked Denoising Auto-Encoder (SDAE) which involves an unsupervised greedy layer-wise pre-training process followed by the training of a predictive model [25].

CHAPTER III: Methodology

This section first describes the input data and target and then the DLNN model, the chosen features, the target, a detailed explanation of the SDAE method, and the description of the reference methods used to compare with the DLNN model developed in this study.

3.1 Dataset

The domain is slightly larger than the County Warning and Forecast Area (CWFA) of the U. S. National Weather Service (NWS) Weather Forecast Office (WFO) in Corpus Christi, Texas (CRP). The latitude/longitude pair of the SW, NW, NE and SE corners of the domain are 26.91000° N/ 100.58000° W, 29.25613 N/100.58000 W, 29.17955 N/96.05539 W, and 26.86122 N/96.15182 W, respectively. For this study, the domain is represented as a grid of 13×22 equidistant points with a horizontal grid spacing of 20 km. To create the foregoing domain, the position of each grid point was determined via the Inverse and Forward computer software programs provided by the US National Geodetic Survey. These points serve as boundaries/centres for 286 20 km $\times$ 20 km (400 km2) square regions. The amount of target data (discussed below) and the desire to reflect the diversity of thunderstorm triggering mechanisms was based on the proximity to Terminal Aerodrome Forecast (TAF) locations. The first box chosen was located in the northeastern sector of the domain (Box 238), a region with a relatively high frequency of cloud-to-ground (CG) lightning strikes, in order to maximize the amount of target data. Another box chosen was located in the far southwest sector adjacent to the Rio Grande River that borders Mexico (Box 73), which climatologically has a much smaller frequency of CG lightning than the corresponding frequency over the northeastern section of the TANN domain; thus, the utility of the model with limited target data could be assessed. The third box chosen was located near the Gulf of Mexico just west of Corpus Christi (Box 103). Boxes 103 and 238 were located in the Western Gulf Coastal Plain region of Texas. Much of the thunderstorm activity initiated within the Coastal Plain is likely to be triggered

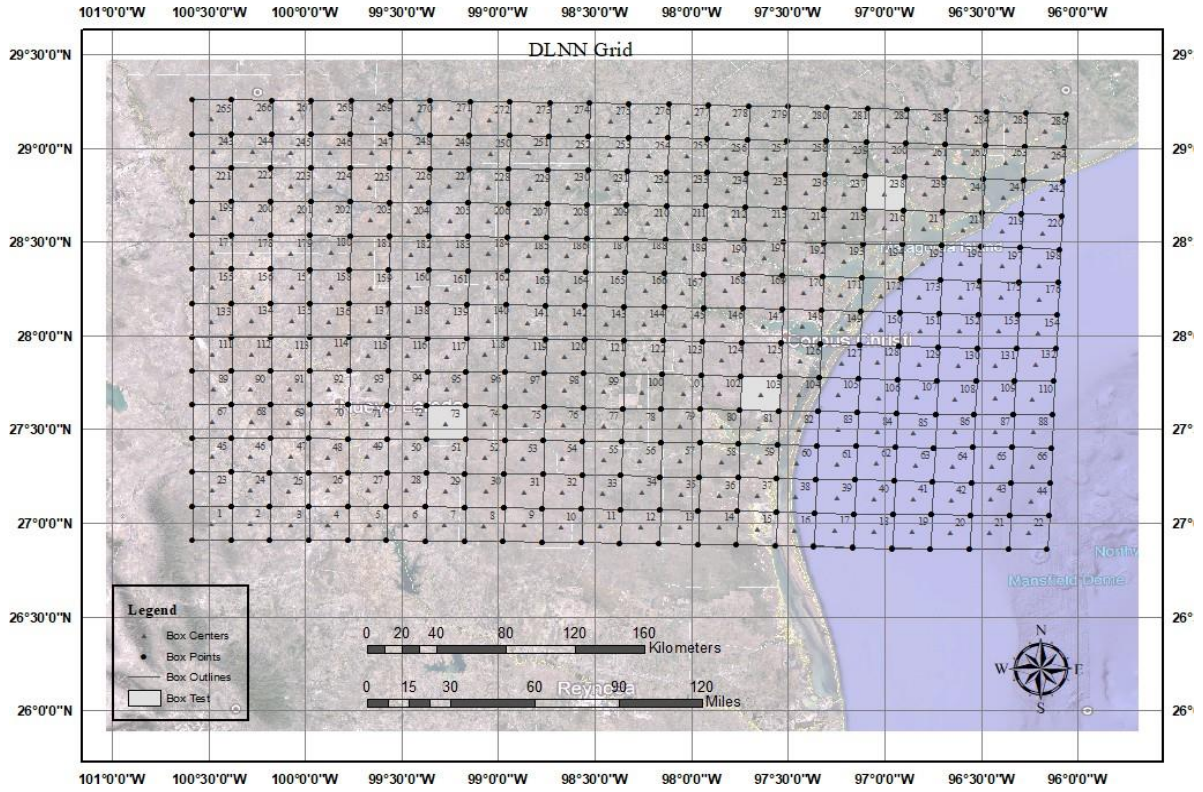


Figure 3.1: Grid domain. This grid is defined by 286 (20 km × 20 km square) regions (boxes).

by coastal processes (e.g. sea breezes). A fraction of thunderstorms near Box 73 develops over the Sierra Madre Oriental mountain chain in Mexico and propagate into the box. Further, South Texas experiences thunderstorm development in association with synoptic scale features (fronts, upper level disturbances). However, on several occasions, even during synoptically favorable dy- namics, thunderstorm development will occur over Box 238, yet not over Boxes 73 and 103, due to the development of an elevated stable layer, possibly in response to the advection of warmer air (possibly due to compressional heating) from approximately southwest to northeast downwind from the Sierra Madre Oriental. Thus, the box regions chosen reflect the diversity with respect to thunderstorm mechanism and frequency.

In this study, the target refers to the existence, or non-existence, of thunderstorms (categorical). CG lightning was chosen as the proxy for thunderstorm occurrence. The CG source data originated

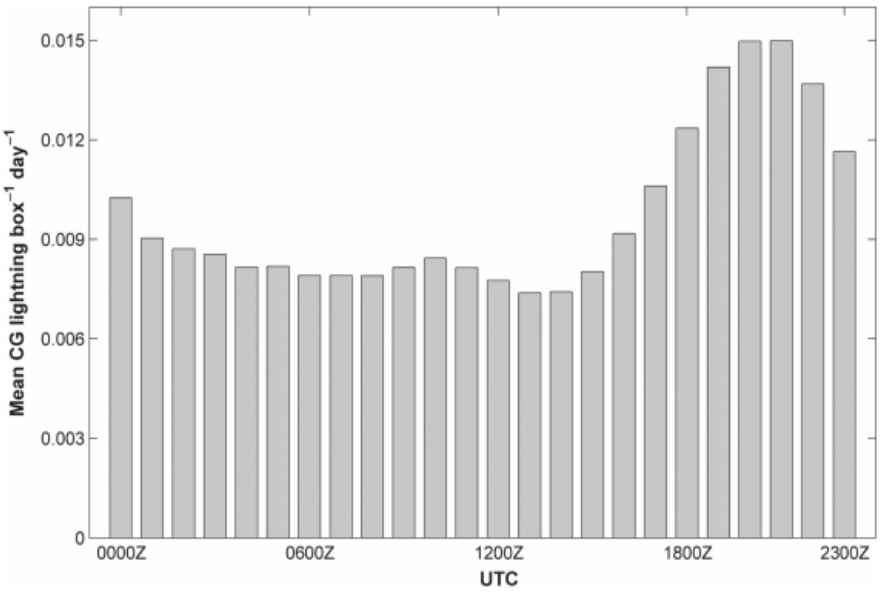


Figure 3.2: Histogram (mean CG Lightning box-1 day-1 versus hour in UTC) of CG lightning strikes for the TANN domain during the period 1 January 2003 through 31 December 2011. Note that the greatest frequency of CG strikes occurs during the afternoon and evening hours [11]
.

from the National Lightning Detection Network (NLDN). The source data are discrete (number of CG lightning strikes per unit time and location) and are transformed to binary.

Figure 3.2 depicts the histogram (mean CG lightning box-1 day-1 versus hour in UTC) of CG lightning for the entire TANN domain (Figure 3.1) for the period 1 January 2003 through 31 December 2011. Note that the majority of thunderstorm activity for the domain typically occurs during the afternoon and early evening hours. These results influenced the decision to predict thunderstorms only for the 2100 − 0300 UTC (±2 h) period.

Table 3.1 depicts the quantity of data utilized in this project. The total number of cases over the study period was 1,148,576 with 939,510 cases used for model calibration and 209,066 cases contained in the 2007-2008 testing set. The strategy is training for whole dataset and test on three mentioned boxes.

Table 3.1: Quantity of data available to train models.

Prediction hour	training sample	Positive target data	Percent positive target
9	663,519	22,139	3.3
12	646,073	16,904	2.6
15	659,801	12,682	1.9

Features

The features chosen originated from the North American Mesoscale Forecast System (NAM)[67] developed by the National Weather Service, National Centers for Environmental Prediction, Environmental Modeling Center (NWS/NCEP/EMC.) The NAM is a placeholder for the operational mesoscale model run on the North American domain. The training and testing sets were derived from the 2004 – 2012 period of the NAM, which includes the hydrostatic Eta [75] (1 March 2004 to 19 June 2006), WRF-NMM (Weather Research and Forecasting Non-hydrostatic Mesoscale Model) [39] (29 June 2006 to 30 September 2011), and NEMS-NMMB (NOAA Environmental Modeling System Non-hydrostatic Multiscale Model on the Arakawa B-grid) (1 October 2011 to 31 December 2012) model output.

The features chosen were the NWP output variables/parameters in [12] and [11]. These predictors were retained based on the selection procedure used by [83] to identify those features predictive of thunderstorms, and those identified through a high random forest-based variable importance to thunderstorm development [62]. These NWP-based features chosen include the familiar atmospheric moisture, instability, and lift factors known to contribute to thunderstorm development, and factors that preclude convection. Tables 3.2 and 3.3 in both [12, 11] depict the specific features used to develop this study's deep learning model and a brief discussion of the rationale for their selection. This information is reproduced in Table 1 and 2. Although model performance was assessed with regard to three (3) separate 400 km^2 box regions, the model was trained/optimized over all 286 boxes. Non-meteorological features, location (lat/lon) and a seasonal function based on the Julian day (Table 3.4) was added the predictors. All 38 predictors and features were used as input features for the DLNN model.

Table 3.2: Description NAM predictor variables/parameters used in DLNN.

Abbreviation	Description (units)	Justification as thunderstorm predictor
PWAT	Total predictable water (mm)	Atmospheric moisture proxy
MR_{850}	Mixing ratio at 850 hPa(gkg^{-1})	Lower-level moisture necessary for convective cell to reach a horizontal scale of $\geq 4km$ [46]
RH_{850}	Relative humidity at 850 hPa (%)	When combined with CAPE, predictor of subsequent thunderstorm location independent of synoptic pattern [18]
CAPE	Surface-based convective available potential energy (Jkg^{-1})	Instability proxy; the quantity $(2CAPE)^{0.5}$ is the theoretical limit of thunderstorm updraft velocity [89]
CIN	Convective inhibition (Jkg^{-1})	Surface-based convective updraft magnitude must exceed $(CIN)^{0.5}$ for parcels to reach level of free convection [89]
LI	Lifted index (K)	Atmospheric instability proxy [26]
U_{LEVEL},V_{LEVEL}	U,V wind components at surface, 850hPa [LEVEL=surface, 850hPa] (ms^{-1})	Strong wind can modulate surface heterogeneity-induced mesoscale circulations ([15, 92])
VV_{LEVEL}	Vertical velocity at 925, 700, 500hPa [LEVEL=925, 700, 500hPa] (Pas^{-1})	Account for mesoscale and synoptic scale thunderstorm triggering mechanisms (sea breezes, fronts, upper level disturbances) that are resolved by the NAM Atmospheric instability proxy; highly sensitive to CI [14]
$DROPOFF_{PROXY}$	Potential temperature dropoff proxy (K)	
LCL	Lifted condensation level (m)	Proxy for cloud base height; positive correlation between cloud base height and CAPE to convective updraft conversion efficiency [97]
T_{LCL}	Temperature at the LCL (K)	$T_{LCL} \geq -10^{\circ}C$ essential for presence of supercooled water in convective cloud essential for lightning via graupel-ice crys- tal collisional mechanism [79]
CP	Convective precipitation ($kg\ m^{-2}$)	Byproduct of the Betts-Miller-Janjic convective parameterization scheme [39], when triggered
VSHEARS8	Vertical wind shear: 10m to 800hPa layer ($10^3 s^{-1}$)	The combination of horizontal vorticity (associated with ambient 0–2km vertical shear), and density current generated horizontal vorticity (associated with 0–2km vertical shear) can trigger new convection [76]
VSHEAR86	Vertical wind shear: 800-600hPa layer ($10^3 s^{-1}$)	Convective updraft must exceed vertical shear immediately above the boundary layer ([14, 13])

Table 3.3: Description NAM initialization predictor variables/parameters used in DLNN.

Abbreviation	Description (units)	Justification as thunderstorm predictor
U_{LEVEL},V_{LEVEL}	U, V wind at the surface, 900, 800, 700, 600, 500 hPa levels [LEVEL=surface, 900, 800, 700, 600, 500] (ms^{-1})	Thunderstorm profile modification owing to veering of wind (warming) or backing of wind (cooling); backing (veering) of wind in the lowest 300hPa can suppress (enhance) convective development [20]
HI_{LOW}	Humidity index (C)	Both a constraint on afternoon convection and an atmospheric control on the interaction between soil moisture and convection [20]
CTP_{proxy}	Proxy for convective triggering potential (dimensionless)	Both a constraint on afternoon convection and an atmospheric control on the interaction between soil moisture and convection [20]
VSHEARS7	Vertical wind shear: surface to 700hPa layer ($10^3 s^{-1}$)	Both a Strong vertical shear in the lowest 300hPa can suppress convective development [20]
VSHEAR75	Vertical wind shear: 700-500hPa layer ($10^3 s^{-1}$)	Convective updraft must exceed vertical shear immediately above the boundary layer for successful thunderstorm development ([14, 13])

Table 3.4: Miscellaneous variables/parameters used in DLNN.

	Abbreviation	Description	Justification
JD	Julian day (day)		Periodic function providing information to the DLNN regarding thunderstorm occurrence as a function of season
Location	latitude and longitude		Providing information to the DLNN regarding thunderstorm occurrence as a function of location

Target

Cloud-to-ground (CG) lightning was used as the proxy for thunderstorm occurrence and was obtained from the terrestrial-based National Lightning Data Network (NLDN) [69]. A target vector was created which contained the number of CG lightning strikes per date/hour/box for the study

duration (2004 − 2012.) The target is defined as: t

date and hour (in UTC), respectively. The value L represents the quantity of CG lightning strikes per hour within a given 400 km^2 box region.

3.2 Methodology

This section describes the DLNN model, the chosen features, the target, a detailed explanation of the SDAE method, and the description of the reference methods used to compare with the DLNN model developed in this study.

Feature Extraction

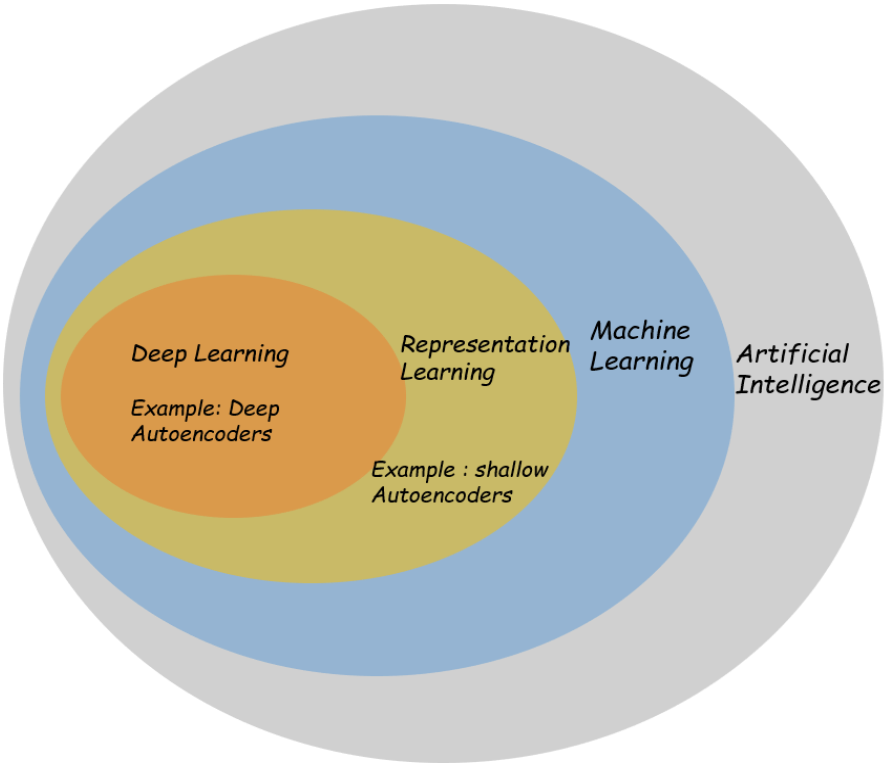


Figure 3.3: A Venn diagram showing how deep learning is a kind of representation learning, and subset of machine learning and AI

While most of the machine learning models are working with raw input features, their performance are influenced by the number of features and variables. Sometimes the performance is degraded by increasing the number of features. This problem known as the course of dimensionality [9]. To solve this problem, manually engineering of features started which was doing by an expert. This technique was time-consuming and error-prone. Automated feature selection method were soon available to reduce the dimension of input space, selecting the best subset of features [91]. But this methodology considers the importance of each feature independently to select or eliminate them, which may in combination with other features, provide useful information. Feature extraction or feature construction [55] is the best alternative to reduce the dimension of input feature space. There are several feature extraction techniques which mainly trying to find a better representation of input features by extracting linear combination of the original ones (known as linear dimensionality reduction techniques like Principal Component Analysis (PCA) [40] or linear discriminant analysis [82]) or nonlinear combination of original ones (known as nonlinear dimensionality reduction techniques such as Kernel PCA, AutoEncoders [91]).

Today, **Artificial Intelligence (AI)** is a succeed field with massive practical application research topics such as intelligent automation of understanding speech or images, medicines and disease detection and many other basic scientific research [8, 1, 72, 45]. AI trying to tackle and solve problems that are intellectually difficult for human beings but understandable for machines by using list of formula and mathematical rules. AI is a function to learn from experience and understand the pattern of data without human involvement in hierarchy manner. The hierarchy manner of learning enables the machines to learn complicated concepts in automated way. AI system need to learn their own knowledge from data automatically, by extracting patterns from raw data, this capability is known as **Machine Learning (ML)**. Figure 3.3. shows a Venn diagram which illustrates deep learning is a subset of representation algorithms and in higher level a subset of machine learning and AI.

So, the performance of ML algorithms depends on the **representation** of the raw data. Each piece of information included in the representation of the data is known as **features** [25]. This

dependency of ML on representation is the main phenomenon to deal with and the choice of representation has an enormous impact on the performance of machine learning models. For many ML algorithms, tasks can be solved by providing the right set of features to extract. But, unfortunately, it is difficult to providing the right features manually and also takes too much time with a lot error sources [91, 25]. One of the best solution to understand features and extract them automatically is representation learning, which not only learn the mapping from representation to output but also learn the representation itself. Automatic learning representation often results in much better performance than hand-crafting design for discovering the representation. In fact, the representation learning algorithm can discover a good set of features related to task which is so much easier and productive to manually designing features. As a matter of fact, extracting high level of representation of features from raw data is difficult because of the existence various nonlinearity and complexity in raw data which sometimes it is head even for ML models. **Deep Learning (DL)** solves this central problem by using more sophisticated representation algorithm. Deep learning as a subset of machine learning is able to learn by experience and acquire skills in unsupervised or supervised way without human involvement from a large amounts of data.

One of the best example of a representation algorithm is the **AutoEncoder (AE)** model [91, 25]. An AE is the combination of an **encoder** function and **decoder** function. Encoder converts the input data into a different level of representation, and a decoder turn the representation back to the original data. AEs are trained into different ways depends on the goal of task, either to preserve highest level of information from input data through encoder or to make the new representation with enough and nice features for the task. In continue, two main models for feature extraction will be explained, PCA as linear model and stacked AEs as nonlinear deep learning model.

PCA

PCA is one of the oldest and the best known of technique for reducing the dimension of feature space which preserving as much variability as possible [100]. Based on this assumption that there are a large number of interrelated variables between features, PCA is trying to reduce the dimen-

sionality of a data while retaining as much as possible variation in features. The central strategy of PCA is transforming the variables to a new set of variables which are not correlated. The new set of variables based on their variability are ordered so that the first few retain variable present the most variation in all of the original variables [41]. Computation of PCA is straightforward and based on eigenvalue-eigenvector concepts. So, PCA for given dataset find a linear subspace of dimension than original data which attempt to maintain most of the variability of the data.

In summary, to implement the PCA analysis we need to do following steps:

1. calculating the mean value for predictors and subtract them from the mean. In fact, by subtracting the mean the coordinate system is transferred to the location of the mean.

2. calculate the covariance matrix, since the non-diagonal elements in this covariance matrix are positive, there is high correlation between predictors.

3. calculating the eigenvalues and eigenvectors of the covariance matrix. these values are so important because they propose very important information about the pattern of in the data. The first eigenvector shows the most important pattern in the data, the second the less important and so on. By extracting the eigenvectors of the eigenvalues of the covariance matrix, the PAC model is able to extract lines that characterise the data.

4. Choosing components and forming a feature vector. In this step, the dimensionality reduction can come in by choosing the eigenvector with the highest eigenvalues as the principal component of the data.

So, basically by using the PCA the data is transformed to eigenvectors lines, by choosing the highest term of pattern between data or highest component of data, where the patterns are the lines to describe the relationship between the data.

AutoEncoders Model

Basic structure of AEs

Generally, there are four different major deep learning models: Unsupervised Pretrained Networks (UPNs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) and Recursive Neural Networks. Diagram in Figure 3.4. depict all different deep learning models with short summary about them. AEs are the one of the best up-supervised learning model act as nonlinear dimensionality reduction technique due to take advantages of greedy-layer wise learning strategy. In this work, particularly the **stacked autoencoder (SAE)** model is backbone of model for training the raw input and predict the thunderstorm lightning.

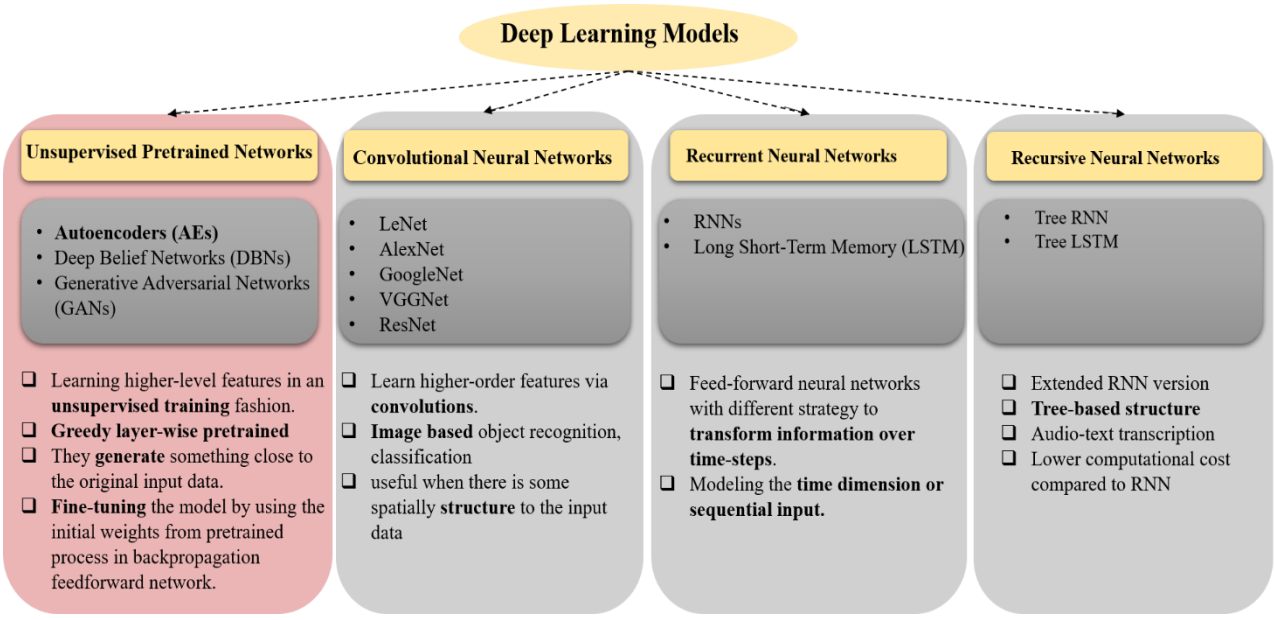


Figure 3.4: There are four major group of deep learning algorithms. Unsupervised pretrained models are trying generate the raw input data as much as accurate in unsupervised manner. They are powerful models for understanding the most important futures from raw input in lower dimension. Convolutional based models mainly used for data with spatial structure like image due to using convolution for learning features. Recurrent and recursive neural networks are using for sequential data with two different strategy for training the model

.

An AutoEncoder (AE) network is a specific type of Feed-Forward Neural Network with sym-

metric structure, which attempts to resemble the inputs as closely as possible to its output. An

AE is constructed based on two main modules, an encoder function to map the input data and a decoder function to reconstruct the original data [2, 91]. Internally, an AE has a hidden layer known as bottleneck to represent the input used by encoder function. The basic structure of an AE, as shown in Figure 3.5, includes an input x, and encoder function (function f) responsible to map the input onto the encoding y (encoding section of AE for learning the latent features) and r the reconstructed features computed using decoder function g. Both x and r must have the same dimension. The dimension of the encoding y is selected based on the properties desired, it can be less than the input dimension known as undercomplete AE, or higher than the the input dimension known as overcomplete AE. The simplest AE consists of just one hidden layer, and is defined by two weight matrices and two bias vectors [9]:

$$y = f(x) = S_1(W_1 x + b_1) \tag{3.1}$$

$$r = g(x) = S_2(W_2 x + b_2) \tag{3.2}$$

where, W_1 and W_2 are weight matrices, b is bias, x is the input data and S_1 and S_2 denote the activation functions.

Traditionally, AEs have been used as dimensionality reduction method or feature extraction technique [2, 91]. To reduce the dimension of input feature by AEs, needs to constrain the dimension of bottleneck (layer y in Figure 3.5) to have a smaller dimension than input features (layer x in Figure 3.5) known as undercomplete AEs. Undercomplete representation strategy forces the AEs to learn the most latent features of the input data to minimize a loss function.

$$\Gamma(w, b; S) = \sum_{x \in S} L(x, g(f(x))) \tag{3.3}$$

Where for weights (w) and bias (b), L is a loss function to minimize the difference between the original input (x) form the set of input S and its reconstruction $g(f(x))$, such as the mean squared error [93] (L_{MSE} , Equation 4) or cross entropy [3] (L_{CE} , Equation 5). Back-propagation is using to update weights and biases for minimizing the reconstruction error[30]. With linear activation

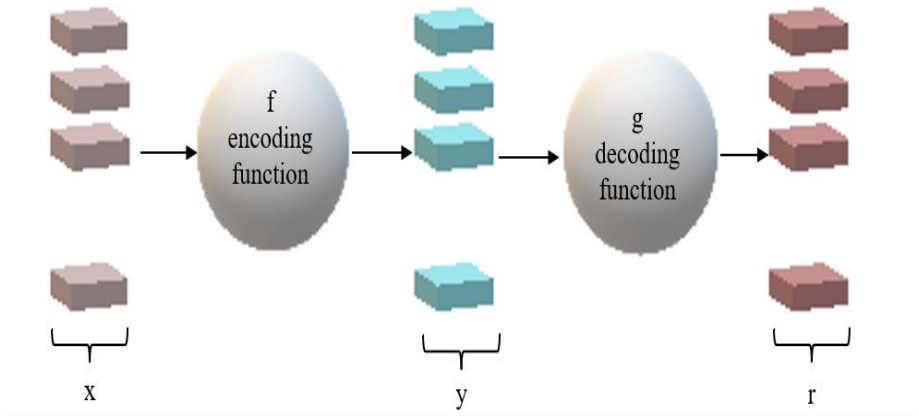


Figure 3.5: The basic structure of an AE with one variable encoding layer, includes an input x which is mapped onto the encoding y via an encoder, represented as function f. This encoding is in turn mapped to the reconstruction r by means of a decoder, represented as function g.

function for encoder and the mean squared error loss function, by extracting the layer *y* from an undercomplete AE, each node can now be treated as a variable in the same way variables are mapped by PCA [40]. But, an undercomplete AE with nonlinear activation function for encoder and decoder functions can learn the most salient features of data distribution compared to PCA (such AEs model describes a nonlinear generalization of PCA, which known as nonlinear PCA) [48].

$$L_{MSE}(r, x) = \|r - x\|^2 \qquad (3.4)$$

$$L_{CE}(r,x) = -\sum_{k=1}^{d} r_k log(x_k) + (1 - r_k) log(1 - x_k) \qquad (3.5)$$

Where r is reconstructed output, x is input, d is set of samples, k is iteration.

Common activation function in use for AEs

In AEs architecture for each hidden layer there is a unit known as activation function, which receives input from the previous layer to compute the ”weighted sum” of these inputs, adds a bias

and then decides whether this neurons should fired (activated) or not (Formula 3.6.).

$$Y = \sum(weight * input) + bias \tag{3.6}$$

The value of Y can be anything from $-inf$ to $+inf$, and the activation function decides to transform it to the specific range and make a decision to activate it or not. The behaviour of the model is rely on choosing the activation function, which can be nonlinear behaviour by selecting nonlinear activation function or linear behaviour. Generally, there are six different most popular activation functions include of **Linear activation function**, **Binary or Boolean** , **Rectified Linear Units (ReLU)**, **Logistic**, **Tanh** and **Scaled Exponential Linear Units (SELU)**. Figure 3.6 present different activation functions.

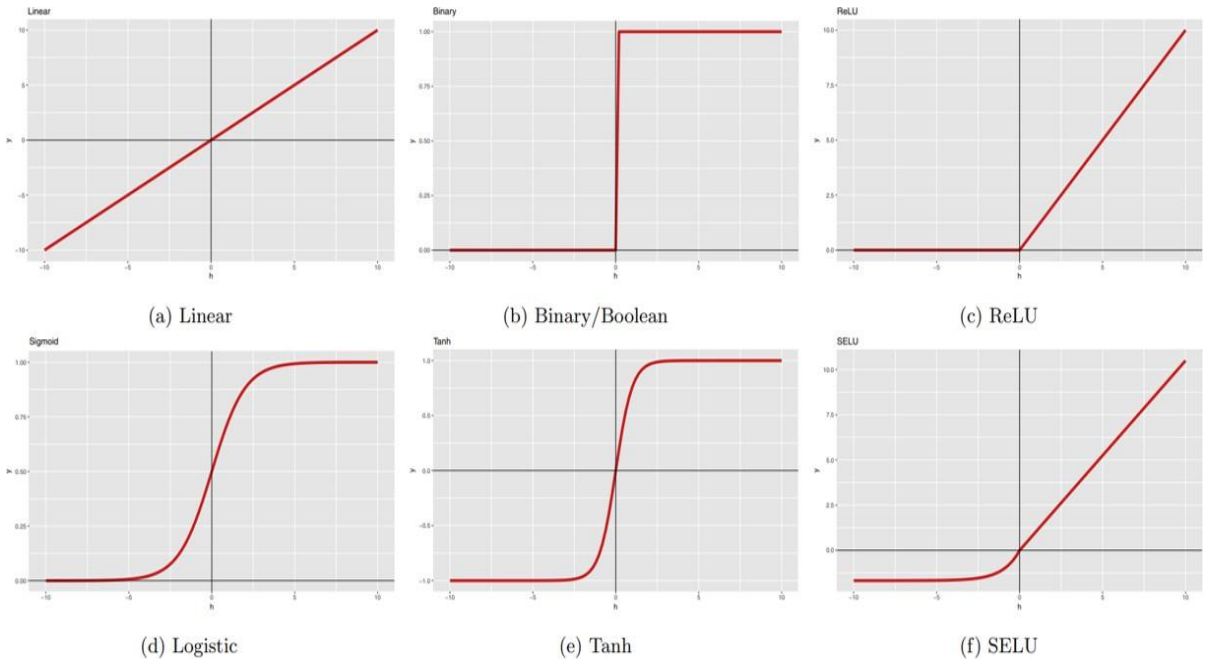


Figure 3.6: Common activation functions [9].

- **Linear activation function**: This activation function is a straight line function ($A = cX$, where A is activation function, X is input and c is constant value), which the output of activation function is proportional to input, the weighted sum of inputs. Obviously, the deriva-

- tive of linear activation function is a constant value (c), means the gradient for optimization

process would has no relationship with input data, so no matter how many layer there is in model. So, this activation function would not be a good choice for AEs model since it rarely provide useful representation.

- **Binary or Boolean activation function** : There are several Boolean functions (NOT, OR, AND, XOR), which are mostly utilized for educational uses, also they have some application in data hashing.

- **ReLU**: The ReLU function is one of the most popular activation function for deep learning models. This activation function gives an output if input is positive and 0 otherwise. The function is $A(x) = max(0, x)$, where x is input. This function tends to degrade the AEs performance since it always output 0 for all negative inputs, although it is one of the best nonlinear activation function. This is a downside of ReLU function known as "dying ReLU", because the slope for negative section of ReLU function is zero, once a neuron get negative value it will died and never can recover it. As we learned, the main goal of AEs is reconstruction the input as much as close to output. So, by removing the negative inputs by ReLU function the performance of reconstruction process.

- **Logistic** : Sigmoid activation function is the most appropriate activation for AEs. It is a nonlinear function ($A = \frac{1}{1+e^{-x}}$, for input x), and combination of them would be nonlinear as well. It is a smooth gradient which its derivative provides enough strong gradients. Unlike linear activation function, the output of sigmoid function is in range of 0 and 1. But, the gradient in end of the sigmoid function tend to be small and has less response to change in input (x), known as vanishing problem. But, still this activation function is very popular for classification and for AEs model.

- **Tanh**: This function is very similar to sigmoid function, known as scaled sigmoid function ($tanh(x) = \frac{2}{1+e^{-2x}} - 1 = 2sigmoid(2x) - 1$). It is similarly has nonlinear characteristic. one point to mention is that tanh activation function has a stronger gradient than sigmoid, so the derivatives are steeper.

- **SELU** : SELU activation function is an improved ReLU function, by adding some slope for negative values. In fact, it designed to combine the good parts of ReLU and allowing to some negative values to participate in learning process, and solving the dying problem of ReLU function.

AE network structures

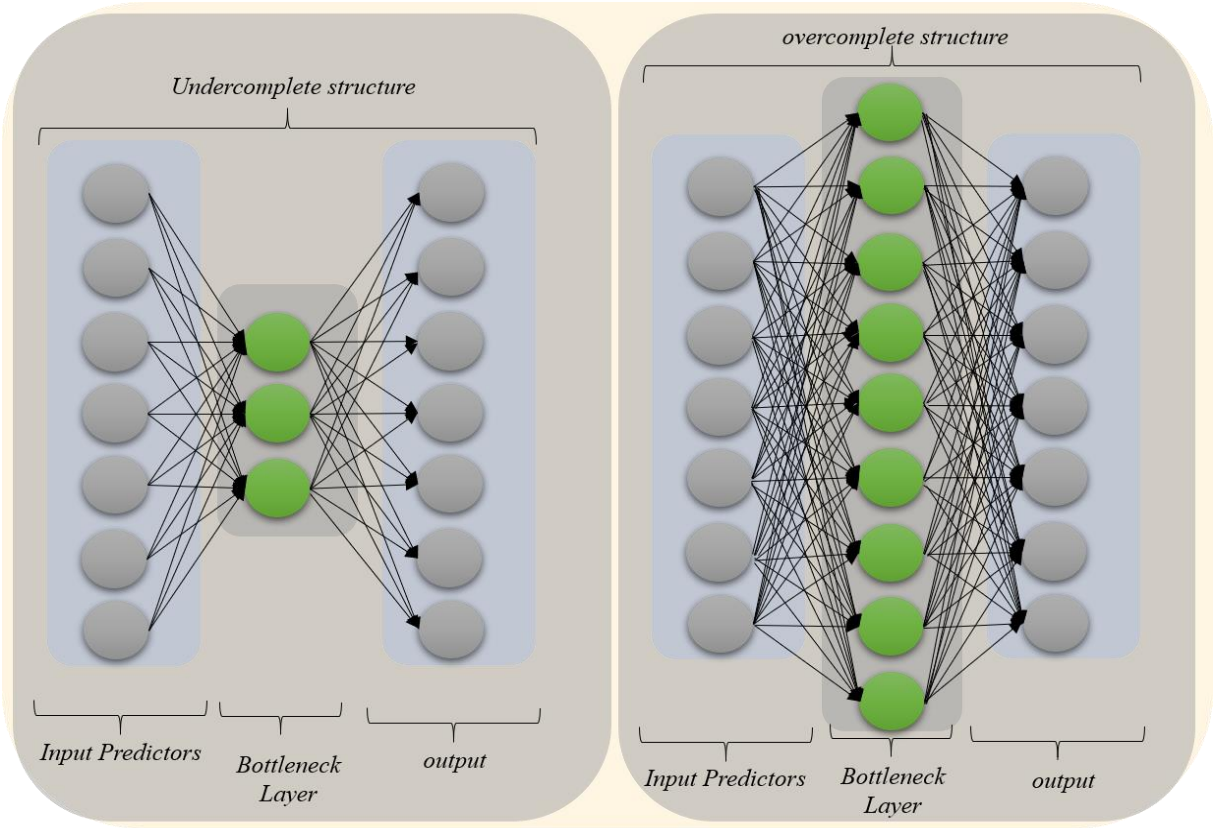
As we learned, AEs are a symmetrical structure, which the encoder and decoder have the same number of layers and same number of neurons for each layers in both side. Depending on the dimensionality of bottleneck layer, AEs are categorized into two main structure:

- **undercomplete**: if the encoding layer at bottleneck has a lower dimension than input features, it is undercomplete structure of AEs (Figure 3.7.a.). This strategy force the network to learn higher level of representation.
- **overcomplete**: when encoding layer at bottleneck has higher dimension than input features, the structure is overcomplete structure (Figure 3.7.b.). This architecture simply trying to learn the identity function, same features of input as output. In combination of overcomplete structure some limitation as regularization process is needed to avoid falling into overfitting problem.

Regularized AEs

like many other machine learning models, AEs are prone to overfitting of the training dataset resulting in poor out-of-sample performance [84]. To avoid the overfitting issue, in addition to limit the model capacity by choosing the undercomplete structure for the model, some regularization terms can be added to the loss function to encourage the model to learn other properties of input data. Regularization means control the capacity of encoder and decoder based on the complexity of the input data distribution. Regularization can be achieved by adding a penalization like weight

decay or sparsity in representation. Weight decay enhance the generalization of training by ampli-



a) Undercomplete structure. b) Overcomplete structure.

Figure 3.7: AEs model based on it structure.

fying smaller weights that produce good reconstruction [30]. In equation 6, the weight decay term is added that λ attempts to determine the magnitude of the decay and limit the weight growth, the resulting loss function could be:

$$\Gamma(w, b; S) = \sum_{x \in S} L(x, g(f(x))) + \lambda \sum_{i} w^2 \qquad (3.7)$$

where w_i are transverses all the weights in W. Also, the regularization can be sparsity known as sparse autoencoder [102]. In sparse autoencoder units in hidden layer usually do not fire to satisfy the low values in average for activation of the encoding layer [102]. Another strategy to force the AEs to learn the best latent features is Denoising AEs (DAEs) [91]. A DAE learns to generate robust features from input by reconstruction from potentially noisy instances. The structure for DAEs is same as AEs and the parameter will be the same as well, just for DAEs during the training

stochastic corruption of the input data will applied. Based on the concept behind the denoising

technique, the obtained representations and latent features are more robust and informative and in turn more useful for reconstruction.

Stacked AutoEncoder

As with any neural network there is the flexibility to construct the AE with different number of hidden layers and number of nodes. Stacked AE (SAE) is an autoencoder with more than one hidden layer. A stacked autoencoder takes advantage of all the benefits of any deep network with higher expressive power and computes features based on the greedy layer-wise training method ([30, 3]). In this training method, the first layer of neurons, which ingests the raw input, is trained to obtain weight and biases that allow for a good reconstruction of the input layer. The output of the first layer is then used by the second layer to obtain its own set of weights and biases with the target to reproduce the output of the first layer. The process is repeated for potential additional layers using the output of each layer as input for the next and computing sets of weights and biases to reproduce the output that previous layer [3]. Based on this strategy, the parameters of each layer are trained individually. Unlike supervised learning that tends to train models directly by gradient descent starting from randomly-initialized parameters, the greedy layer-wise based model is using unsupervised learning to pre-train each layer and leads to progressively higher level representations based on the lower-level representation output of the previous layer [3].

Hinton and Salakhutdinov pointed out that ”Gradient descent can be used for fine-tuning the weights in such AE networks, but this works well only if the initial weights are close to a good solution. They describe an effective way of initializing the weights that allows deep autoencoder networks to learn low-dimensional codes (Greedy-layer wise approach) that work much better than principal components analysis as a tool to reduce the dimensionality of data” [30]. Regarding randomly initializing the weights, optimizing the weights in nonlinear autoencoders is difficult because with large initial weights AE is not able to find local minima, and with small initial weights there is potential for the gradient vanishing problem. The Greedy-layer wise technique or pretraining approach was the solution to solve the training of AE proposed by Hinton and Salakhutdinov

[30]. There is a global fine-tuning stage to replace stochastic by deterministic, real-valued probabilities by using the backpropagation through the whole AE to update the weights for optimal reconstruction. Each layer in an AE extracts high-order correlation between features in the two layers. For a wide variety of data AE is able to reveal a low-dimensional nonlinear structure of the original data.

3.3 Experiment and Results

DLNN Model Setup

DLNN models were developed to predict thunderstorm occurrence within three 400 km^2 box domains (box numbers 73, 103, and 238 in Figure 1) for three prediction hours (9, 12 and 15hr $\pm 2hr$). The data for the periods of $2004-2006$ and $2009-2013$ were used for training the model. The data for the period of $2007-2008$ were used for testing. For each prediction hour (9hr, 12hr, and 15hr), DLNN models were trained over all 286 400-km^2 continuous domains. This approach increases the number of thunderstorm cases ($t_{d,h} = 1$; Section 2.12) and total instances sufficient to justify the use of deep learning. For 9hr prediction, there are 663519 instances in the training sample, 22139 (3.34%) of which are positive target data ($t_{d,h} = 1$). The corresponding values for 12hr prediction are 646073 instances with 16904 (2.62%) positive target. For 15hr, there are 659802 instances in the training sample, 12682 (1.92%) of which are positive target data ($t_{d,h} = 1$).

The development of the DLNN model began with the determination of the SDAE architecture. As discussed in the previous section, the number of predictor variables (X) (the input layer) for the SDAE model is 38. SDAE with different undercomplete and overcomplete architectures were tested. The output of the SDAE is fed into a logistic classifier consisting of 2 neurons and resulting in a binary classification, zero for non-lightning and one for lightning. The SDAE is trainedbased on stochastic gradient descend and experiments are repeated 50 times to assess the variability of the process. The trained model is finally evaluated based on the independent test dataset. In this experiment, we vary the number of neurons in the hidden layers from $1-100$ to determine the

optimum number while also varying the number of hidden layers from 1 to 3. Table 3.5 depicts the selected topology, including the number of layers, number of neurons for each layer, optimization technique, etc.

Table 3.5: Hyper-parameters of SDAE model.

Architecture	**Hyper-parameter**	**Value**
Stacked Denoising AE	*Number of hidden layer*	2 – 3
	Number of neurons	1 – 100
	Activation functions	sigmoid
	Loss function	Cross entropy
	Optimization technique	SGD
	Noise mask	15%; 0; 0
	Regularization parameter λ	0.1 – 0.001
	Learning rate	0.0001
	Training epochs	500

For determining the optimum number of neurons we utilized a 95% confidence interval based on the PSS metric and the standard error estimated based on the 50 iterations [11]. Figure 3.8. depicts the results for the bottleneck layer which it is more important one as this layer determines the number of latent features which will be the input data for the supervised classifier. The optimum number of hidden layer neurons is selected on the maximum PSS while also avoiding an overlap in the standard errors with a solution for a smaller number of hidden neurons. A bottleneck layer with 3 neurons provided the maximum PSS with no overlap in standard error with solutions with 1 or 2 hidden neurons. Based on the same strategy 30 neurons were selected for the first layer. Hence, the result is an undercomplete architecture shown in Figure 3.9.

Conceptually, for the purpose of binary classification or prediction, fine-tuning using back-propagation can be applied by tuning the parameters for all layers and it is common to discard the ”decoding” layers of a stacked autoencoder and link the last hidden layer (bottleneck) to the softmax classifier [91, 23, 77]. Then, the gradients from the softmax classification or prediction error will be back-propagated into the encoding layers [91]. As a softmax classifier for fine-tuning the whole network, logistic regression (logistic function) is applied on top of the network. To avoid overfitting in our model (model $38 - 30 - 3 - 2$), 15% noise is added only for first layer to force

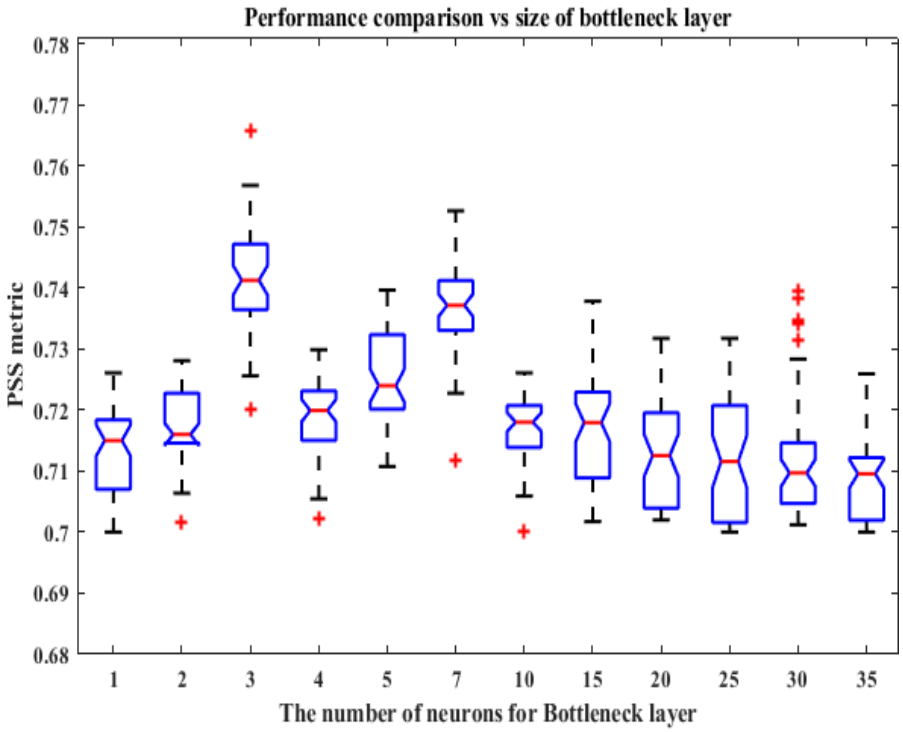


Figure 3.8: Determination of the optimum number of hidden layer neurons for the Box 238 12 hr prediction SDAE model. The box plot shows the median, and interquartile ranges estimated based on the 50 iterations. This graph is for the bottleneck layer. The number of hidden layer neurons, 3, is selected based on the maximum PSS while insuring that there is not overlap with solutions including less hidden neurons.

the SDAE model to better understand latent features. Also a regularization term is added to further decrease the likelihood of overfitting based on formula 5. By trial and error λ is set to 0.001.

The Receiver Operating Characteristics (ROC) curve [5] is a technique for visualizing the skill of binary classifiers. To determine the model that optimizes performance, we created a ROC curve by adjusting the decision threshold at an iteration of SDAE output range and calculating the POD and F at each iteration. Based on the best performance for PSS metric, the threshold is chosen. Figure 3.10 depicts ROC curve associated with the development of the SDAE model for the 12*hr* thunderstorm prediction in box 238.

Feature Reduction Comparison and Model Performance Assessment

blueFigure 3.11 compares the importance of the first ten PCA variables sorted by their ranking. The

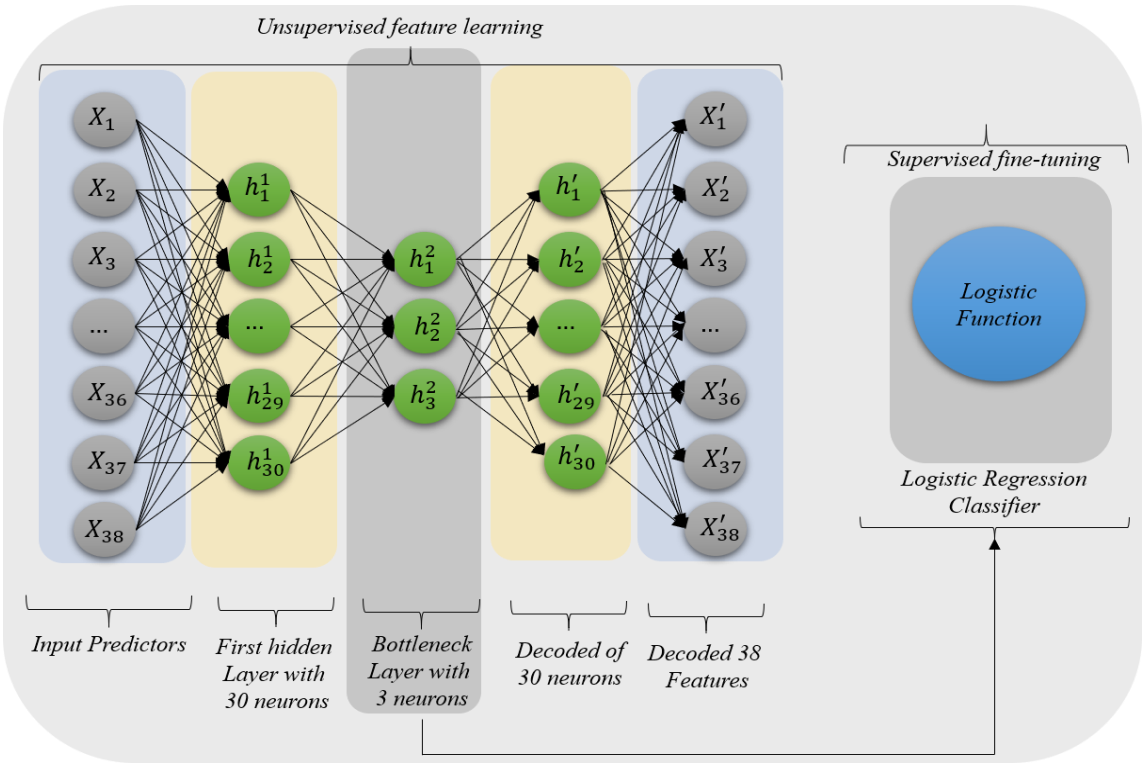


Figure 3.9: SDAE architecture applied for thunderstorm prediction.

first ten variables explain approximately %90 of the variability, but the first three PCA components already provide more than 70% of the variability of the original data. Inputs consisting of both the first three and the first ten PCA components were tested to predict lightning using a logistic regression classifier. Results were very close to each other indicating that using only a simpler input with three PCA components gives a good representation of the performance of this method. Using three PCA components also allows to compare more directly with the SDAE methods which also uses three latent features.

As applied here, the SDAE is learning low-dimensional representations of the data through dimensionality reduction controlled by the number of hidden neurons in the bottleneck layer. The information from the bottleneck is then used as input to the classifier resulting in improved performance by focusing the model on the most relevant information in the input features [**R33**]. SDAE can detect repetitive and redundant structures, consolidate them into lower dimensionality latent features resulting in more distinguishable and more informative features. To better understand how

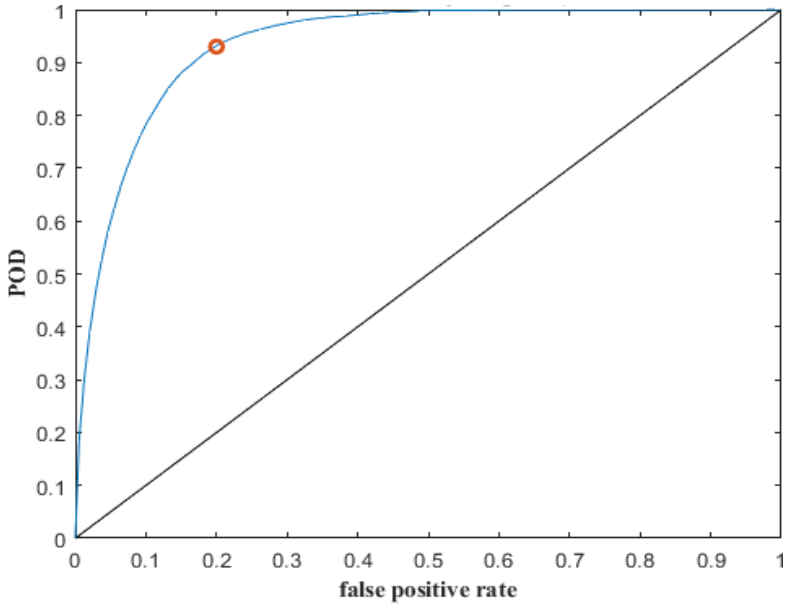


Figure 3.10: ROC curve generated over the training set to select the logistic classifier threshold based on maximizing PSS (0.73).

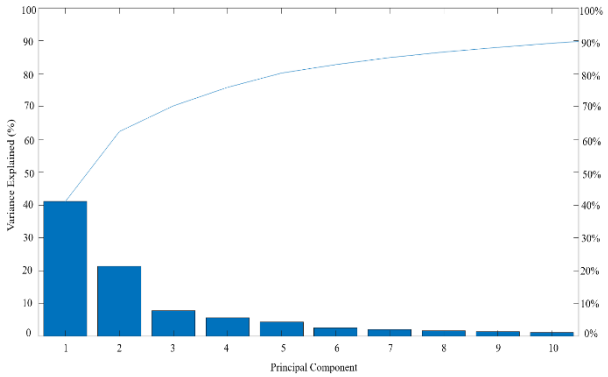
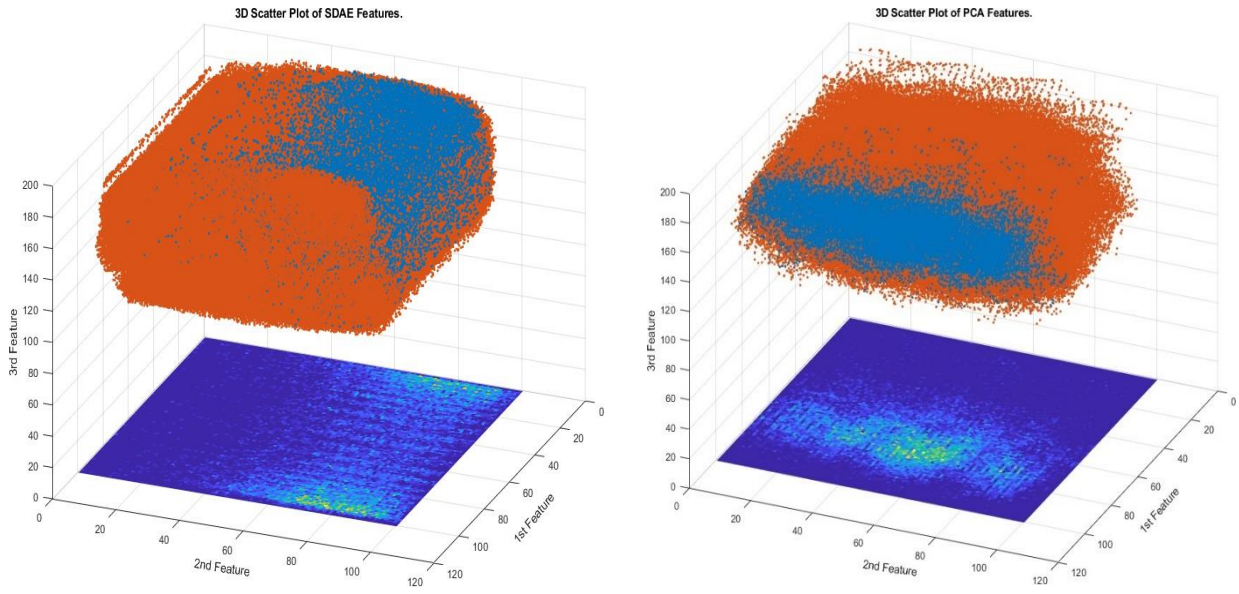


Figure 3.11: Feature importance of PCA model for the training dataset.

dimension reduction may lead to better performance the SDAE output for three latent features is compared with the results of dimension reduction using the linear PCA technique. The comparisons are presented in blueFigure 3.12. The point cloud illustrated in blueFigure 3.12.a. shows the result for the SDAE model, and (blueFigure 3.12.b) presents the result for PCA. Lightning cases are displayed in blue while non lightning cases are displayed in orange for the 3D visualization. Also, density maps for the lightning cases are shown at the bottom of each figure. For the 3D analysis of the SDAE point cloud, one can see that the lightning cases are located along the outermost layer of the feature space forming a wedge around the non lightning cases. Similarly for the



a) Visualization of SDAE latent features.

b) Visualization of PCA latent features.

Figure 3.12: Representation of latent features generated by SDAE and PCA models. 3D scatter of latent features for SDAE and PCA at top and 2D density map only for lightning cases at the bottom.

related 2D density analysis, the lightning cases are mapped in the corner and edges. Such segmentation is more distinguishable than the one illustrated for PCA in (blueFigure 3.12.b) [**R100**]. For both the 3D and the 2D cases of the PCA dimension reduction, the lightning cases are restricted to a portion of the feature space but they are surrounded and mixed in with non lightning cases. Another advantage of the SDAE dimension reduction method is the better use of the feature space as the PCA point cloud does not fill the same cube as thoroughly. The comparisons between the 2D and 3D SDAE dimension reduction also illustrate the need for at least three latent features as the lightning cases are well clustered in a wedge like shape area for the 3D case while being more spread throughout the feature space for the 2D case.

DLNN Model Evaluation

To evaluate the performance of the DLNN model, the model is applied to the independent dataset ($2007 - 2008$.) Based on the confusion matrix for binary classes, 8 different metrics were calculated. The formulation of the confusion matrix and performance metrics are shown in Tables 3.6

Table 3.6: Confusion matrix for calculating scalar performance metrics.

Forecast	**Yes**	**No**	**Total**
Yes	a(hit)	b(false alarm)	a+b
No	c(miss)	d(correct rejection)	c+d
Total	a+c	b+d	a+b+c+d =n

Table 3.7: Evaluation Metrics.

Performance metric	**Symbol**	**Equation**
Probability of detection [0, 1]	**POD**	a/(a+c)
False alarm rate [0, 1]	**F**	b/(b+d)
False alarm ratio [0, 1]	**FAR**	b/(a+b)
Critical Success Index [0, 1]	**CSI**	a/(a+b+c)
Peirce Skill Score [-1, 1]	**PSS**	(ad-bc)/(b+d)(a+c)
Heidke Skill Score [-1, 1]	**HSS**	2(ad-bc)/ [(a+c)(c+d)+ (a+b)(b+d)]
Odds-ratio skill score [-1, 1]	**ORSS**	(ad-bc)/(ad+bc)
Clayton Skill Score[-1, 1]	**CSS**	(ad-bc)/(a+b)(c+d)

and 3.7, respectively. The metrics include PSS, critical success index (CSI), Heidke skill score (HSS), odds-ratio skill score (ORSS) and so on. See ([95, 32]) for more detailed information about the utility of these metrics. Table 3.8-3.10 depict the performance results of the DLNN, shallow neural network, and PCA-based classifiers, and the corresponding performance of the operational forecasters, for boxes 73, 103 and 238.

With respect to the 9 hour prediction Table 3.8, there is a significant improvement in the value of the selected performance metrics for the DLNN models over the CT2016 model [11]. For box 73, the CSI metric increased from 0.19 (CT2016) to 0.55 (DLNN), and there is a large improvement of the DLNN models over the CT2016 models for the HSS (0.25 to 0.54), the CSS (0.19 to 0.58), and for the ORSS, POD and PSS. The F and FAR metrics are substantially improved; F decreases from 0.22 (CT2016) to 0.10 (DLNN), and FAR decreased from 0.81 to 0.44. For box 103, there is improvement of the DLNN models over the CT2016 models for the metrics CSI (0.13 to 0.53), PSS (0.62 to 0.75), HSS (0.15 to 0.52) and CSS (0.12 to 0.51). Also, F and FAR have been decreased from 0.31 (CT2016) to 0.08 (DLNN) and 0.87 to 0.43, respectively. For box 238, the superior performance of the DLNN over CT2016 is similar to the performance improvements at

Table 3.8: Performance results of the deep learning neural network (SDAE) and principal component analysis (PCA) based classifiers developed in this study, the shallow neural network of [11] (CT2016), and the corresponding performance of the operational forecasters (NDFD) for **9hr** prediction.

	POD	F	FAR	CSI	PSS	HSS	ORSS	CSS
Box 73								
SDAE	**0.91**	**0.10**	**0.44**	**0.55**	**0.75**	**0.54**	**0.96**	**0.58**
CT2016	0.94	0.22	0.81	0.19	0.71	0.25	0.96	0.19
NDFD	0.91	0.26	0.83	0.16	0.65	0.21	0.93	0.16
PCA	0.89	0.36	0.94	0.05	0.50	0.06	0.86	0.05
Box 103								
SDAE	**0.93**	**0.08**	**0.43**	**0.53**	**0.75**	**0.52**	**0.96**	**0.51**
CT2016	0.93	0.31	0.87	0.13	0.62	0.15	0.93	0.12
NDFD	1.00	0.31	0.85	0.15	0.69	0.19	1.00	0.15
PCA	0.85	0.40	0.96	0.03	0.45	0.04	0.79	0.03
Box 238								
SDAE	**0.89**	**0.09**	**0.39**	**0.55**	**0.73**	**0.57**	**0.96**	**0.53**
CT2016	0.94	0.30	0.80	0.20	0.63	0.24	0.94	0.19
NDFD	0.94	0.35	0.81	0.19	0.59	0.21	0.93	0.18
PCA	0.83	0.37	0.94	0.05	0.45	0.06	0.78	0.05

the other locations. In general, the DLNN models outperformed the PCA-based models and the operational weather forecasters.

Table 3.9 summarizes performance comparisons for 12*hr* predictions. For box 103, there is an approximately 89%, 88% and 91% improvement of the DLNN models over the CT2016 models for CSI, HSS and CSS metrics, respectively. Also, for the DLNN model, the number of false predictions and the false prediction alarm rate decrease significantly. For box 238, there is approximately 87% , 84% and 89% improvements of the DLNN models over the CT2016 models for the CSI, HSS and CSS metrics, and the F metric decreased from 0.28 (CT2016) to 0.07 (DLNN), and FAR decreased from 0.93 to 0.37. DLNN improvements over the PCA-based models and operational forecasters continue.

Table 3.10 summarizes the 15*hr* prediction performance. The performance improvement of the DLNN over the CT2016 models continues; the F metric for boxes 73, 103 and 238 decreased 68%, 66% and 65%, respectively. Also, FAR decreased by 56%, 60% and 62% for boxes 73, 103 and

Table 3.9: Performance results of the deep learning neural network (SDAE) and principal component analysis (PCA) based classifiers developed in this study, the shallow neural network of [11] (CT2016), and the corresponding performance of the operational forecasters (NDFD) for **12hr** prediction.

	POD	F	FAR	CSI	PSS	HSS	ORSS	CSS
Box 73								
SDAE	**0.82**	**0.07**	**0.36**	**0.55**	**0.74**	**0.66**	**0.96**	**0.60**
CT2016	0.86	0.29	0.90	0.10	0.57	0.12	0.88	0.09
NDFD	0.91	0.23	0.86	0.14	0.68	0.19	0.94	0.14
PCA	0.89	0.42	0.94	0.05	0.48	0.05	0.84	0.05
Box 103								
SDAE	**0.80**	**0.06**	**0.34**	**0.49**	**0.76**	**0.62**	**0.97**	**0.60**
CT2016	0.80	0.23	0.95	0.05	0.58	0.07	0.87	0.05
NDFD	0.80	0.24	0.94	0.06	0.56	0.08	0.86	0.06
PCA	0.85	0.36	0.96	0.03	0.48	0.04	0.82	0.04
Box 238								
SDAE	**0.81**	**0.07**	**0.37**	**0.55**	**0.74**	**0.59**	**0.96**	**0.59**
CT2016	0.81	0.28	0.93	0.07	0.53	0.09	0.83	0.06
NDFD	0.67	0.28	0.92	0.07	0.39	0.08	0.67	0.06
PCA	0.83	0.33	0.93	0.06	0.49	0.07	0.81	0.05

Table 3.10: Performance results of the deep learning neural network (SDAE) and principal component analysis (PCA) based classifiers developed in this study, the shallow neural network of [11] (CT2016), and the corresponding performance of the operational forecasters (NDFD) for **15hr** prediction.

	POD	F	FAR	CSI	PSS	HSS	ORSS	CSS
Box 73								
SDAE	**0.86**	**0.07**	**0.40**	**0.57**	**0.73**	**0.54**	**0.95**	**0.56**
CT2016	0.92	0.25	0.93	0.07	0.68	0.10	0.95	0.07
NDFD	0.86	0.24	0.93	0.07	0.61	0.01	0.89	0.07
PCA	0.82	0.34	0.94	0.05	0.47	0.07	0.79	0.05
Box 103								
SDAE	**0.91**	**0.07**	**0.38**	**0.56**	**0.78**	**0.58**	**0.97**	**0.58**
CT2016	0.83	0.21	0.96	0.04	0.62	0.05	0.90	0.03
NDFD	1.00	0.19	0.96	0.04	0.81	0.07	1.00	0.04
PCA	0.75	0.25	0.95	0.04	0.49	0.06	0.79	0.04
Box 238								
SDAE	**0.84**	**0.08**	**0.36**	**0.59**	**0.72**	**0.57**	**0.95**	**0.60**
CT2016	0.64	0.23	0.95	0.05	0.41	0.06	0.71	0.03
NDFD	0.92	0.23	0.92	0.08	0.69	0.11	0.95	0.07
PCA	0.70	0.20	0.91	0.08	0.50	0.11	0.80	0.07

238, respectively. For CSI, HSS and CSS there are 87%, 81% and 87% improvement for box 73. The improvement of CSI, HSS and CSS continue for boxes 103 and 238. Continued improvement of the DLNN models over that of the PCA-based variety and the operational forecasters was noted.

As mentioned earlier, and depicted in Tables 3.8-3.10, the DLNN classifiers provided greater performance than the PCA-based models (with respect to all performance metrics). Another dimension reduction approach was attempted in CT2016, minimum Redundancy, Maximum Relevance [4], to select a few predictors. This dimension reduction method did not lead to improvement in performance as compared to the results presented Tables 3.8-3.10 either. This is true for all prediction times and all test boxes. These results demonstrate that latent features generated by SDAE are more informative and distinguishable than those provided by PCA and another dimension reduction method. These results show that SDAE (by using nonlinear activation functions and greedy layer-wise learning) is able to model the complexity and nonlinearity of original predictors in order to achieve better performance. While PCA is limited to learning the linear relationship between the original predictors and the latent features.

CHAPTER IV: FUTURE WORK and CONCLUSION

DLNN models were developed to predict thunderstorms $\leq 15hr$ in advance within 400 km^2 square regions in a south Texas domain (United States.) The DLNN models were constructed via features originating from NWP model output variables/parameters that influence and preclude convective development and from location (latitude/longitude) and Julian day variables in order to train the models to predict thunderstorms as a function of location and season. Cloud-to-ground lightning served as the thunderstorm proxy and as the target. The particular deep learning technique used was SDAE, a type of representation learning, whereby unsupervised learning occurs across a multitude of hidden layers in order to create a higher order representation of the original features as a pre-training step. The highest order representation of the output served as features used to train predictive models via logistic regression. DLNN model performance exceeded substantially that of corresponding shallow neural network models developed by [11]. 2016 developed shallow feed-forward multilayer perceptron (MLP) models using a second-order learning algorithm, and using an iterative process to determine the number of hidden layer neurons that optimize performance.

We speculate that the superior performance of the DLNN classifiers over the shallow neural network classifiers is due to the ability of SDAE to identify the non-linear combination of the initial features that optimizes the performance of the subsequent predictive model. The DLNN (SDAE/logistic regression) predictive models were also compared with predictive models developed using Principal Component Analysis (PCA) as the pre-training step (PCA/logistic regression.) The SDAE based models performed superior to that of the PCA based models. The low optimal dimensionality of the resulting latent features (three) associated with SDAE allowed for visual comparison between the results of the SDAE non-linear dimension reduction and latent variables generated by the linear PCA. This comparison illustrates the better clustering of the two categories (lightning and non-lightning) by the non-linear method and provides an explanation for the superior performance of SDAE over PCA.

With respect to the skill-based performance metrics HSS and PSS, the performance of the DLNN models in this study generally exceeded substantially the corresponding performance of operational forecasters (NDFD in Tables $11-13$ in [12], and reproduced in Tables $7-9$ in this study.) This superior thunderstorm predictive performance of the DLNN models developed in this study demonstrates the predictive power of representation learning (SDAE combined with logistic regression) and suggests future improvement in operational thunderstorm forecasting (small sample size notwithstanding.) Such forecast improvements would benefit society greatly given the adverse socio-economic impacts of thunderstorms to specific industries such as aviation [101]; [17], and to human life [33], [34], [66]. We plan to conduct an additional assessment of DLNN model performance in a future study by comparing to the well-known ensemble approach used in operational meteorology, mentioned in the Introduction section.

REFERENCES

[1] Dario Amodei et al. “Deep speech 2: End-to-end speech recognition in english and mandarin”. In: *International conference on machine learning*. 2016, pp. 173–182.

[2] Pierre Baldi. “Autoencoders, unsupervised learning, and deep architectures”. In: *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012, pp. 37–49.

[3] Yoshua Bengio et al. “Greedy layer-wise training of deep networks”. In: *Advances in neural information processing systems*. 2007, pp. 153–160.

[4] Vilhelm Bjerknes. “Das Problem der Wettervorhers-age, betrachtet vom Standpunkte der Mechanik und der Physik”. In: *Meteor. Z.* 21 (1904), pp. 1–7.

[5] Andrew P Bradley. “The use of the area under the ROC curve in the evaluation of machine learning algorithms”. In: *Pattern recognition* 30.7 (1997), pp. 1145–1159.

[6] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[7] George H Bryan, John C Wyngaard, and J Michael Fritsch. “Resolution requirements for the simulation of deep moist convection”. In: *Monthly Weather Review* 131.10 (2003), pp. 2394–2416.

[8] Tsung-Han Chan et al. “PCANet: A simple deep learning baseline for image classification?” In: *IEEE transactions on image processing* 24.12 (2015), pp. 5017–5032.

[9] David Charte et al. “A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines”. In: *Information Fusion* 44 (2018), pp. 78–96.

[10] Sutapa Chaudhuri. “Convective energies in forecasting severe thunderstorms with one hidden layer neural net and variable learning rate back propagation algorithm”. In: *Asia-Pacific Journal of Atmospheric Sciences* 46.2 (2010), pp. 173–183.

[11] Waylon G Collins and Philippe Tissot. “Thunderstorm Predictions Using Artificial Neural Networks”. In: *Artificial Neural Networks-Models and Applications*. IntechOpen, 2016.

[12] Waylon Collins and Philippe Tissot. “An artificial neural network model to predict thunderstorms within 400 km2 South Texas domains”. In: *Meteorological Applications* 22.3 (2015), pp. 650–665.

[13] JR Colquhoun. “A decision tree method of forecasting thunderstorms, severe thunderstorms and tornadoes”. In: *Weather and forecasting* 2.4 (1987), pp. 337–345.

[14] N Andrew Crook. “Sensitivity of moist convection forced by boundary layer processes to low-level thermodynamic fields”. In: *Monthly Weather Review* 124.8 (1996), pp. 1767–1785.

[15] GA Dalu et al. “Heat and momentum fluxes induced by thermal inhomogeneities with and without large-scale flow”. In: *Journal of the atmospheric sciences* 53.22 (1996), pp. 3286–3302.

[16] Clarence W De Silva. *Intelligent machines: myths and realities*. CRC press, 2000.

[17] Wenzhe Ding and Jasenka Rakas. “Economic Impact of a Lightning Strike–Induced Outage of Air Traffic Control Tower: Case Study of Baltimore–Washington International Airport”. In: *Transportation research record* 2501.1 (2015), pp. 76–84.

[18] Véronique Ducrocq, Diane Tzanos, and Stéphane Sénési. “Diagnostic tools using a mesoscale NWP model for the early warning of convection”. In: *Meteorological Applications* 5.4 (1998), pp. 329–349.

[19] Kimberly L Elmore, David J Stensrud, and Kenneth C Crawford. “Explicit cloud-scale models for operational forecasts: A note of caution”. In: *Weather and forecasting* 17.4 (2002), pp. 873–884.

[20] Kirsten L Findell and Elfatih AB Eltahir. “Atmospheric controls on soil moisture-boundary layer interactions: Three-dimensional wind effects”. In: *Journal of Geophysical Research: Atmospheres* 108.D8 (2003).

[21] Michael A Fowle and Paul J Roebber. “Short-range (0–48 h) numerical prediction of convective occurrence, mode, and location”. In: *Weather and Forecasting* 18.5 (2003), pp. 782–794.

[22] David John Gagne et al. “Interpretable Deep Learning for Spatial Analysis of Severe Hailstorms”. In: *Monthly Weather Review* 2019 (2019).

[23] Jonas Gehring et al. “Extracting deep bottleneck features using stacked auto-encoders”. In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 3377–3381.

[24] Harry R Glahn and Dale A Lowry. “The use of model output statistics (MOS) in objective weather forecasting”. In: *Journal of applied meteorology* 11.8 (1972), pp. 1203–1211.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[26] Alwin J Haklander and Aarnout Van Delden. “Thunderstorm predictors and their forecast skill for the Netherlands”. In: *Atmospheric Research* 67 (2003), pp. 273–299.

[27] James A Hanley and Barbara J McNeil. “The meaning and use of the area under a receiver operating characteristic (ROC) curve.” In: *Radiology* 143.1 (1982), pp. 29–36.

[28] Emilcy Hernández et al. “Rainfall prediction: A deep learning approach”. In: *International Conference on Hybrid Artificial Intelligence Systems*. Springer. 2016, pp. 151–162.

[29] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. “A fast learning algorithm for deep belief nets”. In: *Neural computation* 18.7 (2006), pp. 1527–1554.

[30] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *science* 313.5786 (2006), pp. 504–507.

[31] Stephen Hodanish, Ronald L Holle, and Daniel T Lindsey. “A small updraft producing a fatal lightning flash”. In: *Weather and forecasting* 19.3 (2004), pp. 627–632.

[32] Robin J Hogan et al. “Equitability revisited: Why the “equitable threat score” is not equitable”. In: *Weather and Forecasting* 25.2 (2010), pp. 710–726.

[33] Ronald L Holle. “Annual rates of lightning fatalities by country”. In: *20th International lightning detection conference*. Vol. 2425. 2008.

[34] Ronald L Holle and Raul E Lopez. “A comparison of current lightning death rates in the US with other locations and times”. In: *International Conference on Lightning and Static Electricity*. 2003, pp. 16–18.

[35] Moinul Hossain et al. “Forecasting the weather of Nevada: A deep learning approach”. In: *2015 international joint conference on neural networks (IJCNN)*. IEEE. 2015, pp. 1–6.

[36] Random House. *Random House Webster’s college dictionary*. Random House, 1998.

[37] William W Hsieh and Benyang Tang. “Applying neural network models to prediction and data analysis in meteorology and oceanography”. In: *Bulletin of the American Meteorological Society* 79.9 (1998), pp. 1855–1870.

[38] Zaviša I Janjić. “The step-mountain eta coordinate model: Further developments of the convection, viscous sublayer, and turbulence closure schemes”. In: *Monthly Weather Review* 122.5 (1994), pp. 927–945.

[39] Zavisa I Janjic, JP Gerrity Jr, and S Nickovic. “An alternative approach to nonhydrostatic modeling”. In: *Monthly Weather Review* 129.5 (2001), pp. 1164–1178.

[40] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[41] Ian T Jolliffe and Jorge Cadima. “Principal component analysis: a review and recent developments”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.

[42] John S Kain et al. “A feasibility study for probabilistic convection initiation forecasts based on explicit numerical guidance”. In: *Bulletin of the American Meteorological Society* 94.8 (2013), pp. 1213–1225.

[43] Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

[44] Bekir Karlik and A Vehbi Olgac. “Performance analysis of various activation functions in generalized MLP architectures of neural networks”. In: *International Journal of Artificial Intelligence and Expert Systems* 1.4 (2011), pp. 111–122.

[45] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *Advances in neural information processing systems*. 2017, pp. 5574–5584.

[46] Marat Khairoutdinov and David Randall. “High-resolution simulation of shallow-to-deep convection transition over land”. In: *Journal of the atmospheric sciences* 63.12 (2006), pp. 3421–3436.

[47] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[48] Mark A Kramer. “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE journal* 37.2 (1991), pp. 233–243.

[49] Ryan Lagerquist, Amy McGovern, and David John Gagne. “Deep Learning for Spatially Explicit Prediction of Synoptic-scale Fronts”. In: *Weather and Forecasting* 2019 (2019).

[50] Quoc V Le et al. “ICA with reconstruction cost for efficient overcomplete feature learning”. In: *Advances in neural information processing systems*. 2011, pp. 1017–1025.

[51] Quoc V Le et al. “On optimization methods for deep learning”. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress. 2011, pp. 265–272.

[52] Robert R Lee and Jeffrey E Passner. “The development and verification of TIPS: An expert system to forecast thunderstorm occurrence”. In: *Weather and forecasting* 8.2 (1993), pp. 271–280.

[53] CE Leith. “Theoretical skill of Monte Carlo forecasts”. In: *Monthly Weather Review* 102.6 (1974), pp. 409–418.

[54] Xiang Li et al. “Deep learning architecture for air quality predictions”. In: *Environmental Science and Pollution Research* 23.22 (2016), pp. 22408–22417.

[55] Huan Liu and Hiroshi Motoda. *Feature extraction, construction and selection: A data mining perspective*. Vol. 453. Springer Science & Business Media, 1998.

[56] Edward N Lorenz. “The predictability of a flow which possesses many scales of motion”. In: *Tellus* 21.3 (1969), pp. 289–307.

[57] EN Lorenz. “Deterministic nonperiodic flow, J. of Atmos”. In: *Sci. 20 (1963) 130* 141 (1963).

[58] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[59] Agostino Manzato. “The use of sounding-derived indices for a neural network short-term thunderstorm forecast”. In: *Weather and forecasting* 20.6 (2005), pp. 896–917.

[60] Donald W McCann. “A neural network short-term forecast of significant thunderstorms”. In: *Weather and Forecasting* 7.3 (1992), pp. 525–534.

[61] Richard P McNulty. “A statistical approach to short-term thunderstorm outlooks”. In: *Journal of Applied Meteorology* 20.7 (1981), pp. 765–771.

[62] John R Mecikalski et al. “Improved convective initiation forecasting in the Gulf of Mexico region through enhanced uses of NASA satellite assets and artificial intelligence”. In: *15th Conference on Aviation, Range, and Aerospace Meteorology*. 2011.

[63] John R Mecikalski et al. “Probabilistic 0–1-h convective initiation nowcasts that com- bine geostationary satellite observations and numerical weather prediction model data”. In: *Journal of Applied Meteorology and Climatology* 54.5 (2015), pp. 1039–1059.

[64] GA Mills and JR Colquhoun. “Objective prediction of severe thunderstorm environments: Preliminary results linking a decision tree with an operational regional NWP model”. In: *Weather and Forecasting* 13.4 (1998), pp. 1078–1092.

[65] Tom Michael Mitchell. *The discipline of machine learning*. Vol. 9. Carnegie Mellon University, School of Computer Science, Machine Learning …, 2006.

[66] NWS. *Natural Hazards Statistics National Weather Service Office of Climate, Water, and Weather Services* . http://www.nws.noaa.gov/om/hazstats.shtml.. 2015.

[67] NWS/EMC. *The North American Mesoscale Forecast System.* https://www.emc.ncep.noaa.gov/index.php?branch=NAM. 2019.

[68] Isidoro Orlanski. "A rational subdivision of scales for atmospheric processes". In: *Bulletin of the American Meteorological Society* (1975), pp. 527–530.

[69] Richard E Orville. "Development of the national lightning detection network". In: *Bulletin of the American Meteorological Society* 89.2 (2008), pp. 180–190.

[70] Jaideep Pathak et al. "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.4 (2018), p. 041101.

[71] Donat Perler and Oliver Marchand. "A study in weather model output postprocessing: using the boosting method for thunderstorm detection". In: *Weather and Forecasting* 24.1 (2009), pp. 211–222.

[72] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[73] N Ravi et al. "Forecasting of thunderstorms in the pre-monsoon season at Delhi". In: *Meteorological Applications* 6.1 (1999), pp. 29–38.

[74] Markus Reichstein et al. "Deep learning and process understanding for data-driven Earth system science". In: *Nature* 566.7743 (2019), p. 195.

[75] Eric Rogers et al. "Changes to the operational "early" Eta analysis/forecast system at the National Centers for Environmental Prediction". In: *Weather and Forecasting* 11.3 (1996), pp. 391–413.

[76] Richard Rotunno, Joseph B Klemp, and Morris L Weisman. “A theory for strong, long-lived squall lines”. In: *Journal of the Atmospheric Sciences* 45.3 (1988), pp. 463–485.

[77] Tara N Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. “Auto-encoder bottleneck features using deep belief networks”. In: *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2012, pp. 4153–4156.

[78] José Luis Sánchez, Eduardo Garcıa Ortega, and José Luis Marcos. “Construction and assessment of a logistic regression model applied to short-term forecasting of thunderstorms in Leon (Spain)”. In: *Atmospheric research* 56.1-4 (2001), pp. 57–71.

[79] CPR Saunders. “A review of thunderstorm electrification processes”. In: *Journal of Applied Meteorology* 32.4 (1993), pp. 642–655.

[80] Sebastian Scher. “Toward Data-Driven Weather and Climate Forecasting: Approximating a Simple General Circulation Model With Deep Learning”. In: *Geophysical Research Letters* 45.22 (2018), pp. 12–616.

[81] Maurice J Schmeits, Kees J Kok, and Daan HP Vogelezang. “Probabilistic forecasting of (severe) thunderstorms in the Netherlands using model output statistics”. In: *Weather and forecasting* 20.2 (2005), pp. 134–148.

[82] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. “Nonlinear component analysis as a kernel eigenvalue problem”. In: *Neural computation* 10.5 (1998), pp. 1299–1319.

[83] Thorsten Simon et al. “Probabilistic forecasting of thunderstorms in the Eastern Alps”. In: *Monthly Weather Review* 146.9 (2018), pp. 2999–3009.

[84] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[85] David J Stensrud. *Parameterization schemes: keys to understanding numerical weather prediction models*. Cambridge University Press, 2009.

[86] Storm-Prediction-Center. *Short Range Ensemble Forecast (SREF) Products*. http://www.spc.noaa.gov/exper/sref. 2019.

[87] Roland Stull. *Meteorology for scientists and engineers*. Brooks/Cole, 2000.

[88] Yumeng Tao et al. "A deep neural network modeling framework to reduce bias in satellite precipitation products". In: *Journal of Hydrometeorology* 17.3 (2016), pp. 931–945.

[89] SB Trier. "CONVECTIVE STORMS— Convective Initiation". In: (2003).

[90] Michael A VandenBerg, Michael C Coniglio, and Adam J Clark. "Comparison of next-day convection-allowing forecasts of storm motion on 1-and 4-km grids". In: *Weather and Forecasting* 29.4 (2014), pp. 878–893.

[91] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion". In: *Journal of machine learning research* 11.Dec (2010), pp. 3371–3408.

[92] Jingfeng Wang, Rafael L Bars, and Elfatih AB Eltahir. "A stochastic linear theory of mesoscale circulation induced by the thermal heterogeneity of the land surface". In: *Journal of the Atmospheric Sciences* 53.22 (1996), pp. 3349–3366.

[93] Zhou Wang and Alan C Bovik. "Mean squared error: Love it or leave it? A new look at signal fidelity measures". In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117.

[94] Morris L Weisman et al. "Experiences with 0–36-h explicit convective forecasts with the WRF-ARW model". In: *Weather and Forecasting* 23.3 (2008), pp. 407–437.

[95] Daniel S Wilks. *Statistical methods in the atmospheric sciences*. Vol. 100. Academic press, 2011.

[96] Daniel S Wilks. *Statistical Methods in the Atmospheric Sciences: An Introduction, electronic version*. 2006.

[97] E Williams et al. “Thermodynamic conditions favorable to superlative thunderstorm updraft, mixed phase microphysics and lightning flash rate”. In: *Atmospheric Research* 76.1-4 (2005), pp. 288–306.

[98] James W Wilson et al. “Nowcasting thunderstorms: A status report”. In: *Bulletin of the American Meteorological Society* 79.10 (1998), pp. 2079–2100.

[99] N Wolchover. *Machine Learning’s ‘Amazing’ Ability to Predict Chaos*. https://www.quantamagazine.org/machine-learnings-amazing-ability-to-predict-chaos-20180418/.. 2018.

[100] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[101] Marilyn M Wolfson and David A Clark. “Advanced aviation weather forecasts”. In: *Lin- coln Laboratory Journal* 16.1 (2006), p. 31.

[102] Jun Xu et al. “Stacked sparse autoencoder (SSAE) for nuclei detection on breast cancer histopathology images”. In: *IEEE transactions on medical imaging* 35.1 (2015), pp. 119– 130.

[103] Yunji Zhang et al. “Practical predictability of the 20 May 2013 tornadic thunderstorm event in Oklahoma: Sensitivity to synoptic timing and topographical influence”. In: *Monthly Weather Review* 143.8 (2015), pp. 2973–2997.