THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND COMPUTER ENGINEERING

MACHINE LEARNING APPLICATIONS FOR SEVERE STORM DETECTION AND PREDICTION

KYLE BRADLEY
SPRING 2020

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Computational Data Sciences and Applied Mathematics
with honors in Data Sciences

Reviewed and approved* by the following:

James Z. Wang
Professor of Information Sciences and Technology
Thesis Supervisor

John Hannan
Professor of Computer Science
Honors Adviser

* Electronic approvals are on file.

ABSTRACT

Weather prediction using machine learning attempts to use historical weather data to make predictions about the future. Especially with recent advancements in computer vision algorithms, many visual forms of weather data can be examined under this machine learning framework. Most importantly, increasing the “lead time” of accurate weather predictions is vital in the space of weather forecasting. In this work, the effectiveness of making predictions on simulated data is examined. This thesis shows that there is promise to detection of storms in this simulated setting. The model was able to detect storms accurately in 87% of simulated images. This result shows that making predictions on simulated weather data is an avenue that can be explored further to increase prediction lead time. Given these results and the ability to use a more refined model, it could be possible that accurate predictions can be made using a similar process more than 48 hours in advance.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I'd like to thank Dr. Wang for his advice, both related to my thesis and related to my career. Also I appreciate the opportunity given to me to work on this project. I'd also like to thank Rachel Xinye Zheng for her weekly support in meetings which provided excellent insight and direction to my project, while still allowing me to come up with solutions myself. To Dr. Hannan, I'd say thank you for always having answers to questions related to both career planning and any administrative difficulties I experienced. Finally, I'd like to thank my family for their continued support throughout my college experience and for encouraging me to pursue my interests.

Chapter 1

Introduction

The task of predicting the weather is one that has been important to society since the dawn of time. In this thesis, the effectiveness of computational methods in predicting severe weather events is examined. Also, the effectiveness of predicting severe weather events in simulated weather images is examined, a method which could aid in increasing the "lead time" on computational weather forecasts. The model is able detect storms in both the real and simulated data, with more accuracy on the real data compared to the simulated data.

The overall goal of this work is to develop a system to make predictions on new radar images, and investigate how effective these predictions are on simulated images.

First, I will discuss the background of some of the technology that I use in my project. Then, I will discuss the dataset, and the model that I built which makes effective predictions on real radar images. Finally, I will discuss the model I built for the simulated images, and the promise for future work in this area.

1. Motivations for Weather Prediction

Predicting the weather has profound economic and social impact. Economically, businesses can better plan for weather incidents when shipping their products, airlines can better plan flights to avoid weather related delays, farmers can understand how their crops might be affected in the coming year. Socially, the government can be more ready to address weather related natural disasters, people can plan their days better and be safe from all types of dangerous

weather. There are two components that can be improved about weather predictions: First, the accuracy. Accuracy of weather predictions can be evaluated in many ways, did it rain as much as predicted, did it rain at the time that was predicted, did it rain where it was predicted? All of these questions are important to improving weather prediction systems. The other area in which weather predictions can be improved is the lead time in making a forecast. It is more valuable to know two days in advance what the weather will be than to know what the weather will be in one hour.

2. Related Work and Project Goals

Feature extraction is key to detecting desired objects in images. Some popular features such as edge detection, texture analysis and motion detection have shown promise in computer vision applications [15]. One feature extracting tool that has been effective in weather related tasks is the Gabor filter. In [16], Yuan et. al. describe how the Gabor filter is effective in quantifying texture in vision tasks, with storm detection being a proposed use case. Another important feature in detecting objects in images is to detect the edges. In [17] a method for detecting coastlines from satellite images is presented. In order to aid in detecting the edges of the coastline, the Canny edge detection algorithm was utilized and was able to find the edges. Another feature that has been shown to be effective is the Histogram of Oriented Gradients. In [2] the Histogram of Oriented Gradients method was used on the storm detection task.

Detection of storm events has become very refined in recent years. In [2], a method for detection on satellite images is presented. The work uses a region proposal system and then makes predictions on the regions detected. Predictions on radar images has also been effective.

In both [3] and [11] machine learning methods are applied to radar images to detect storms. In [12] storms are detected in real time using conventional modelling with success. Through all these methods, detection is made in a short time window. The major difficulty in weather forecasting is making long-term severe weather predictions. In [18], predictions are made on severe weather events, but even then the predictions are only 12 hours or less in advance.

The goal of this project will be to overcome the lack of ability to predict severe weather events far into the future. By using similar methods to those introduced in this section, applied to simulated radar data up to 48 hours in the future, this thesis will attempt to increase the lead time of detection of severe weather events.

Chapter 2

Background

1. History of Weather Prediction

The prediction of weather has been important to society for thousands of years. Weather impacts society in many ways. Some examples are people knowing what to wear on a daily basis, businesses like farmers managing crops, and communities planning for the potential impact of severe weather events. Early weather prediction began in 650 B.C. when the Babylonians tried to predict short term weather changes based on the appearance of clouds. However, these predictions were largely unsuccessful due to their reliance on speculation and personal observations. [1] Around the 15th century, it became apparent that more information was needed to accurately predict the weather. The advent of the first hygrometer, an instrument to detect humidity; the first thermometer, an instrument to detect temperature; and the first barometer, an instrument to detect air pressure, all came around this time. By the 19th century, these and other tools were more refined and widespread. As communication channels opened worldwide, researchers were better able to effectively share their observations with others and the first weather maps showing wind patterns and storm systems were created. With these charts and the increased amount of information available, the first large-scale, or synoptic, weather forecasts were made.

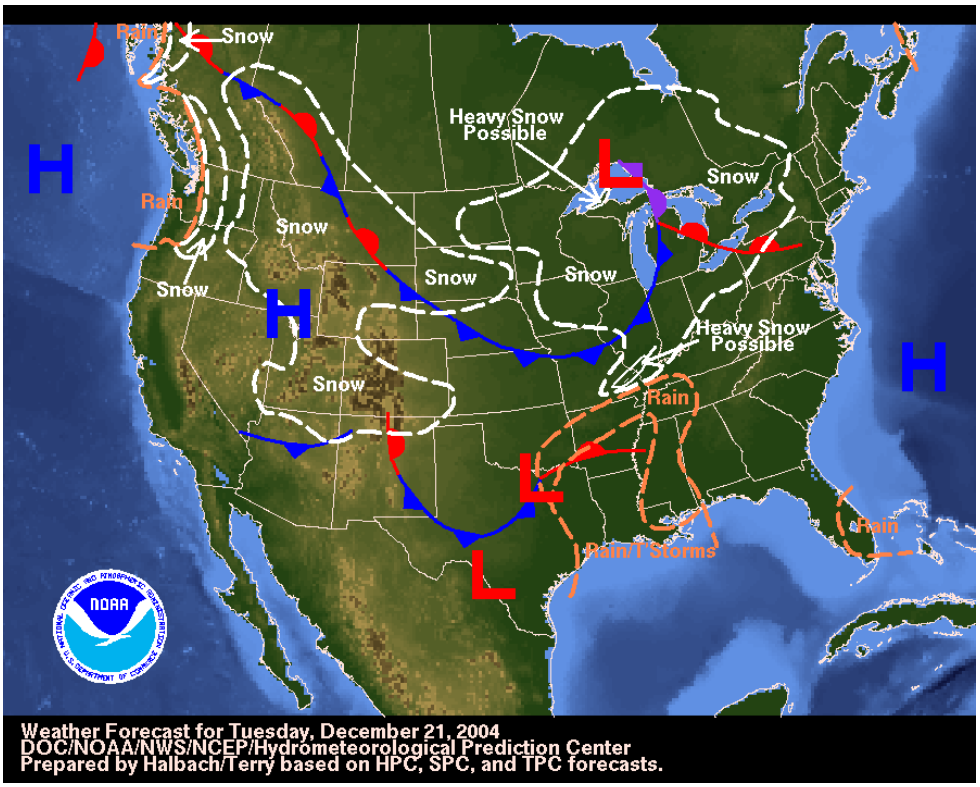

Figure 1. A Synoptic Weather Forecast. These forecasts help meteorologists map how high and low pressure systems move across the Earth. Based on the movement of these systems, weather predictions can be made for large regions. [4]

These forecasts used meteorologists' understanding of high and low pressure systems and how they historically moved across the country in order to predict the weather in locations covered by these fronts.

By the early 20th century, numerical weather predictions started to be made. Lewis Fry Richardson made the first mathematics-based weather prediction when he proposed a method to solve a system of differential equations using understood properties of thermodynamics which were developed around this time. It took several months for him to arrive at his six hour forecast,

which did not end up being accurate. Regardless of the lack of accuracy, he laid the groundwork for future predictions using similar methods.

With the advent of computation, Richardson's methods were able to be revisited, and by 1950 the first 24 hour forecast was successfully made. Moving towards the present, as computing power becomes stronger and stronger, more data is able to be fed into these traditional numerical forecasting methods and more accurate predictions are able to be made. Furthermore, with more advanced satellite and radar images available to meteorologists, it has allowed meteorologists to visually see weather formations to allow them to better forecast and track storm movement.

However, with how complex the physical system that determines the weather is, predictions are still not able to be made perfectly. Recent advancements in machine learning have aided the accuracy of forecasts.

2. Machine Learning Applications in Weather Forecasting

One of the key ways in which machine learning will be able to impact weather forecasting is in its ability to analyze image data quickly. Beyond the numerical forecasts, much of weather forecasting is interpreted by experts looking at satellite images to detect cloud formation and movement. This can cause delay, and limits the ability to quickly make forecasts about storms, which is a big problem. With the advent of deep learning and other pattern recognition tools, we are potentially able to train models to detect storms much quicker and on a larger scale than human forecasters can. In this paper,[2] the power of computational methods for detecting comma shaped clouds, a strong indicator of storm formation, are examined and shown to be quite successful.

Many new weather systems combine the power of machine learning with the traditional numerical techniques to make very powerful predictions. For example, IBM has been working on a super high resolution weather forecasting system called GRAF which will allow weather predictions to be made at an unprecedented scale and accuracy with much higher localization. [6]

3. Technology Overview

In this section, I will review the specific technology/algorithms used in my project. From an introduction to machine learning, to an explanation of the convolutional neural network (CNN) used to make predictions in this work, I give a general introduction to some of the important ideas used later on in this thesis.

A. Machine Learning Overview

Machine learning is a cross section of statistics and computer science which has gained popularity in recent years due to the vast increase in computational power that is now available. In essence, machine learning uses statistical methods to fit a model to a given example, or training data. The model attempts to "learn" some patterns inherent in the data, so that when given new, or testing, data the model can make accurate predictions about some target variable. Machine learning has many different applications, such as text processing, computer vision, numerical forecasting, etc. All machine learning tasks can generally be divided into supervised learning, unsupervised learning, and reinforcement learning. For the purposes of this project, I will focus on supervised learning. In supervised learning, one presents the model with training data which is labeled with the sought target variable, and then predictions are made on the test

variable. For a pertinent example, a supervised model could be constructed to detect storms, with training data being radar images labeled as containing either a storm or no storm. The model would then use this data to identify patterns in the various categories to work so that when presented with unlabeled data, it can try to identify these same patterns to make a prediction. This type of task is called classification, where there is a number of categories that the model must place new data into. [5]

An important part of machine learning is optimization. In order to "learn," the model is minimizing a function, called the loss function. This function is able to calculate how accurate the model's predictions are. Then, different methods from calculus and numerical analysis are used to find the model parameters necessary to achieve a minimum loss on segments of the training data. Often times, this minimum loss is then able to achieve effective results on the testing data. In regression, where numerical values are predicted, this loss function is often times the sum of the squared differences between the predicted values and the actual values. In classification tasks, this same loss function is often used as the model output is a probability that the model believes the data point belongs to that class. The difference is then computed as this probability subtracted from 1, meaning that the data point actually is that class. This sum of least squares loss function is useful because it has an easily computed derivative which aids in minimizing the value of the function. [5]

B. Neural Networks

The neural network is an effective way for computers to approximate very complex functions. Ultimately, the purpose of any machine learning algorithm is to approximate the

underlying function that generates the data. For example, if the task was to predict sales of a certain product based on different features such as season, prior performance, customer interest, and many other factors, then there is some function given all possible data that can exactly predict the amount of sales. However, since for most problems of interest one cannot typically account for every feature that would go into this underlying function, the goal is to develop a function that can as closely approximate it as possible, which can be scored by the loss function.

The neural network can be thought of as a composition of functions, where each function takes some input and gives an output. When chaining these functions together, as the number of functions increases, the ability to approximate the underlying function increases as well.

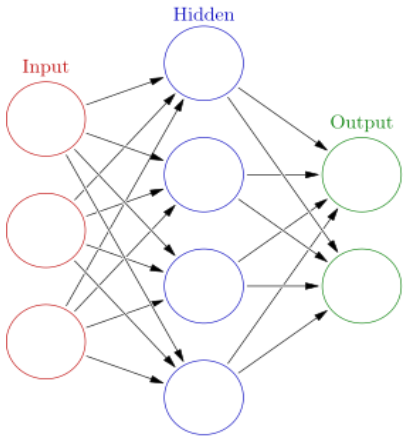

Figure 2 An example of the structure of a neural network. The network consists of nodes which take input, compute a function and send an output to the next layer. By stacking these layers, [7]

In Figure 2, an example of neural network structure is shown. The circles represent functions, with arrows representing inputs and outputs. These functions are initialized using some method chosen by the user, and then as data is fed through the chain it eventually reaches the output layer. The output layer outputs its predictions for the target variables and these are compared with the actual known values. Then, loss is computed as well as the gradient in order

to update the functions used along the way. Given enough data and a relationship between the data and their target values, the neural network is able to approximate the target function with strong accuracy. [8]

C. The Convolutional Neural Network

The CNN has shown to be quite effective in problems where the data takes on a "grid" shape, like image data. The important feature in the convolutional neural network is the convolution layer. The convolution layer, as its name suggests uses the mathematical operation of convolution using a learned filter. This filter passes over sections of the data and convolves it resulting in a single number that is representative of the area that the pixel is in. This is important because in an image, the pixel by pixel information does not provide a great deal to the model, but when using convolution we are able to get a larger context for different areas of the image and get an understanding of how different parts of the image are composed. This allows for more interesting features to be understood by the model than if one simply fed the image data into a standard neural network. CNNs have been very effective in a multitude of tasks related to computer vision and are the foundation of exciting technologies such as self-driving cars. [8]

Chapter 3
Data Description

There are several data sources used in this project. I will detail them in this chapter. The first is radar reflectivity images taken from 2017 in the plains area between the Appalachian and Rocky Mountains. The pixel intensity represents the radar reflectivity value. There are images taken every five minutes so there is a vast amount of data. [13]

Figure 3 An image from the real images dataset. This radar image was taken on April 26, 2017 at 9:45 pm. It shows a cloud with strong storm features in the center of the image.

The second part of the data is the severe weather record archived in the website of

NOAA Storm Prediction Center (SPC). The storms are selected based on the region that the reflectivity values are from. So for every image in the first data set, if there is a storm present, there is information in these records detailing whether it was a hailstorm, tornado, or strong wind and where it happened.

The third part of the data is simulated radar images. These simulated images are produced using physical properties of cloud formation and simulate what will happen in the next 48 hours. The images are produced at 12:00 AM on the day starting the simulation and have one image per hour of simulated radar data. [14]

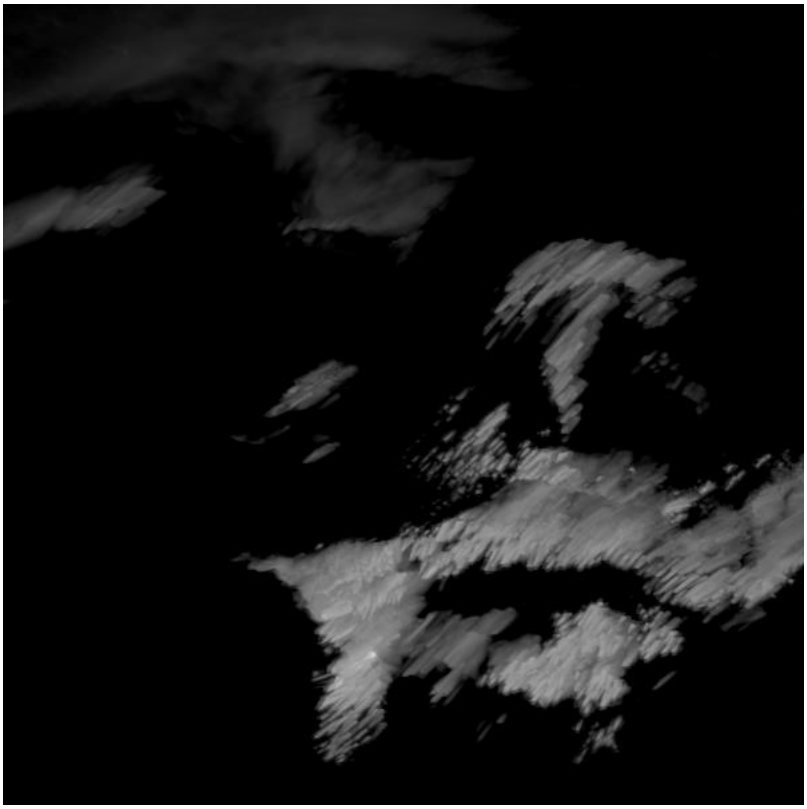

Figure 4 A Sample Simulated Image. This image was produced the morning of January 2, 2017 and is simulating 17 hours ahead. So this simulation represents 5:00 pm on January 2, 2017 [14]

The fourth part of the data is similar to the second part, but is a matching of storms to the simulated radar images. So if a storm happened in real life, there is a correspondence between the simulation and the true value.

Chapter 4

Model for Real Images

1. Outline of Structure

The idea behind the creation of a model for this problem is to build a classifier to examine subsections of the original radar images, and classify these patches as either storms or not storms. The motivation for predicting on subsets of the data is twofold. First, since the radar images cover a large section of the United States, we want our predictions of where a storm is happening or going to happen to provide some level of additional knowledge. We would not want to just say that there is a storm happening somewhere in the image, because that would not give an indication of where the storms are happening. Secondly, feeding a smaller image into the model allows for less computational cost, which makes the model easier to build and train, and allows for faster predictions.

2. Region Proposal

To begin, I set out on a system to detect areas of interest for further analysis from the input radar image. The first step taken to find these regions is to set a minimum threshold to filter out any small clouds with low intensity. A Gaussian filter is then applied to the thresholded image, and afterwards another threshold is applied. The logic behind this is that any densely packed areas of clouds will overcome the second thresholding, but any small regions will be punished more heavily. After the noise has been thresholded, k-means is applied to the indexes

of the pixels that are not 0. A k-value is selected by increasing k until an additional increase in k does not decrease the error more significantly than the previous decrease. After this process, there are 2-4 areas of interest for any one image. The idea is to then create a model to detect whether a storm is present in this region or not.

The reason this works and the motivation behind this method is that regions we want to examine will be regions with closely packed clouds. And similarly, regions we are not interested are regions where there is no radar reflectivity. Thus, through this method we are able to isolate interesting sections of the images, which are more likely to be storms so that we can then distinguish between them in our model.

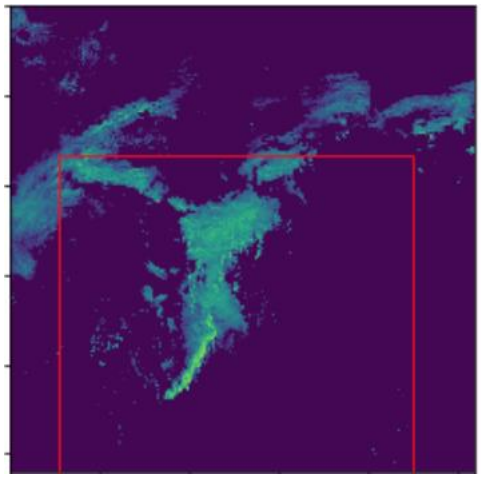

Figure 5 The Region Proposal Method. The red box denotes the region detected by the k-means algorithm as being "interesting"

3. Feature Creation

A common feature of storm formation is the comma shape and the bow echo. These structures have curvature and texture properties that we can try to exploit in order to give the model some more information. Below are some common cloud shapes where you can see these types of features.

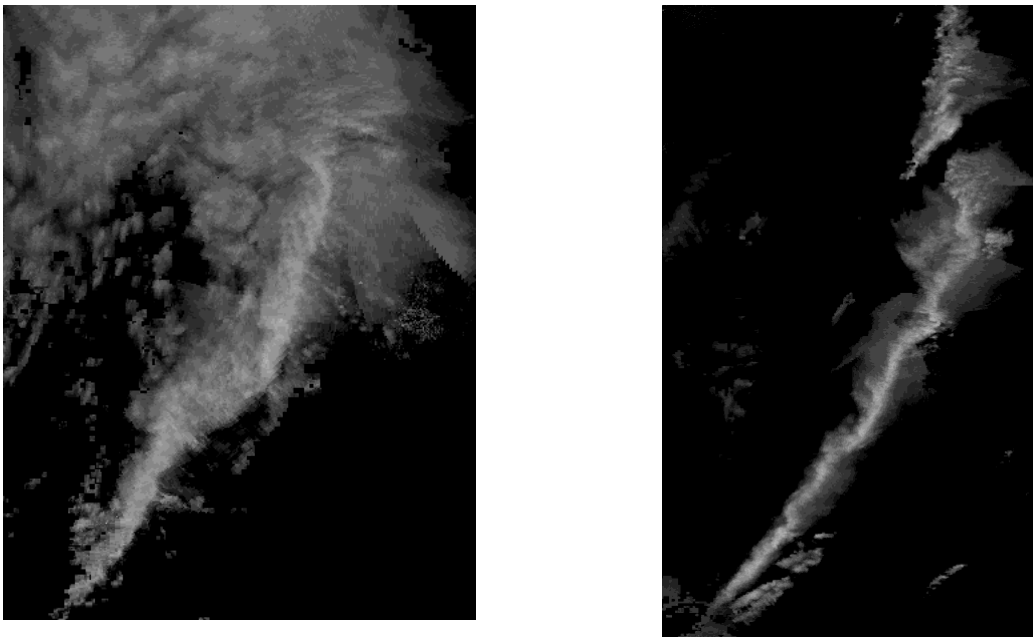

Figure 6 Common Cloud Shapes. On the left, a comma shaped cloud. On the right, a "bow-echo" shape

A . Curvature

One important feature that was examined was the degree to which the left side of the cloud curved. In both the comma shaped and bow echo cloud formations, they tend to have an inward curvature on the left side of the cloud. So the idea was to see if we could determine whether a given cloud had this curvature or not. In order to try to identify these features, the "derivative" of the image was evaluated by passing the Sobel filter over it in the x direction. This allows us to find the leftmost edge of the cloud by selecting positive image derivative values.

Points were then sampled from the edge and a line was drawn connecting them. Then, we determine whether the midpoint of the line crosses through the cloud or not. This method is certainly not foolproof, but it does lead to fairly good results. In the image below, most lines drawn are over the image, so we could say that this cloud exhibits the desired curvature.

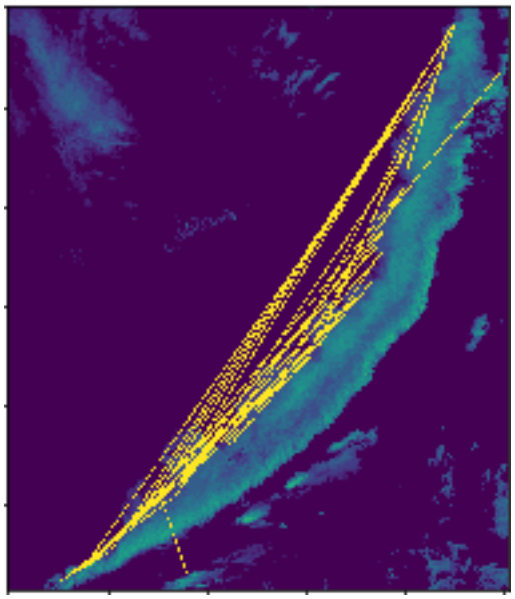

Figure 7 An example of the output of the curvature method. The yellow lines above the cloud indicate that this cloud likely exhibits curvature.

The approach will be to take the percentage of these lines that do not overlap with the cloud formation. This way, we will have a way to quantify how curved the cloud appears to be. A threshold of 50% was used to determine if a cloud was indeed curved or not.

B. Frame Correlation

Another area of investigation was frame differencing, that is to take the correlation in value between two subsequent images. This way, once a storm was identified, its movement

could be tracked onto the next frames following. I was able to implement this, with an example shown below, and can now reconstruct a path of a storm by taking the maximum correlation to be the new center of the storm, also shown below. Once the model was created, and I was able to detect storms, this would be a way for the storm to be tracked across multiple frames. Below is an example of a correlation between two subsequent frames. The point of highest correlation would indicate close to where the storm center is. Also, the overall trend reveals some information about the direction heading to the northeast.

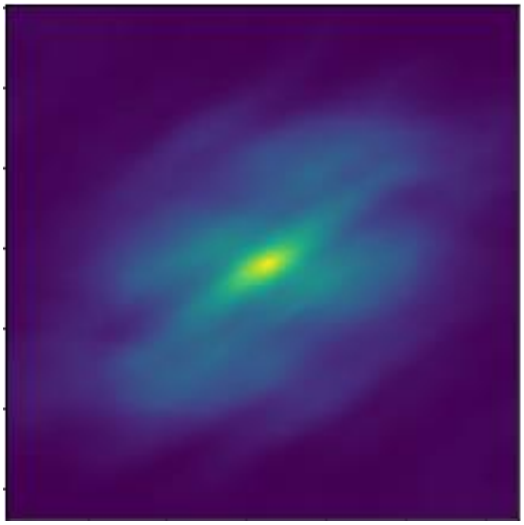

Figure 8 Results after frame correlation. The high intensity values indicate where the new frame shares similar characteristics with the old frame. The trend toward the upper right shows the direction of movement.

4. Model Setup

For this model, the goal was to be able to feed in images cropped from the region proposal method. To prepare the data, sections were cropped from the positive storm examples where the storms were located and made sure that these examples showed some features indicative of a storm. After cropping these images, we are left with 1302 positive examples to give to the model. Then, random crops were taken of images that did not contain storms and

those crops which had greater than 25% cloud coverage were kept. I then set up a training and testing set by splitting all training examples that were from before July so that there was no examples that were essentially repeated between the training and test set.

5. Results

Upon training the model and making predictions on the test set we get the following results. The model trained was a simple neural network with 3 layers. The training set was mainly focused on very strong examples, so the task was relatively easy, but this exercise was more to better prepare for training on the simulated examples

	Negative Examples	Positive Examples
Precision	94.1%	85.9%
Recall	66.4%	98%
F1	77.9%	91.5%

So we see that the model is somewhat over predictive on positive examples and a little under predictive on negative examples. However, these results indicate that the difference between positive and negative examples can be picked up even by our very simple model. We will take this information and use it to continue forward and make build a model on the simulated data.

Chapter 5

Simulated Model

The next stage of my work focused on creating a similar model to the previous, but with simulated radar images. The simulated images are created at the beginning of a day, and project out what the radar reflectivity will look like for every hour, over the next 48 hours. An example can be seen below.

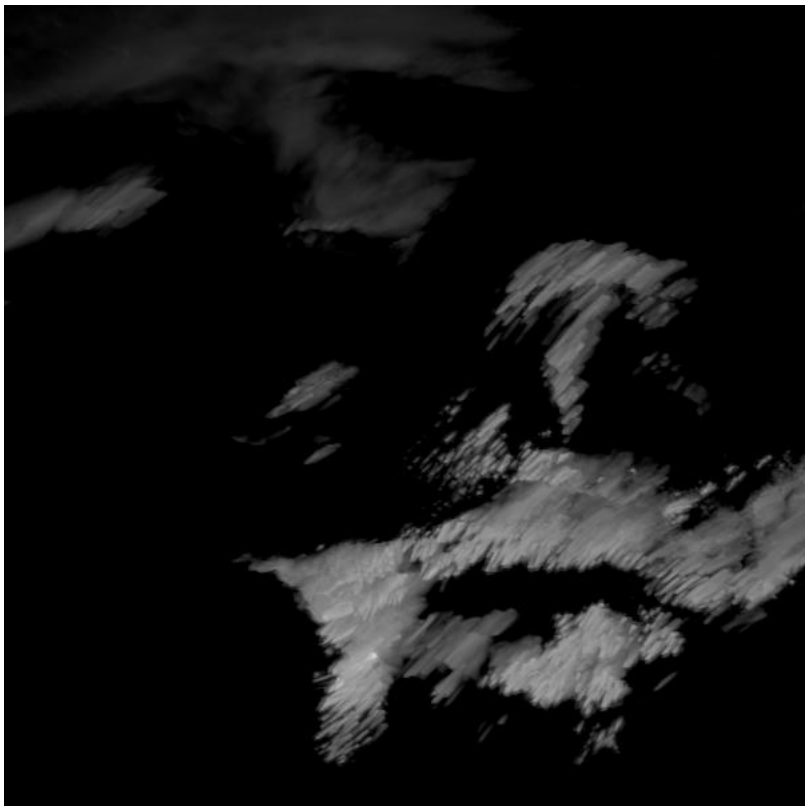

Figure 9 A sample image from the simulated data set. Near the bottom middle it exhibits storm-like features. This image is a duplicate of Figure 4, created on January 2, 2017, forecasting 17 hours ahead.

The simulated images present a little more difficulty in the detection task because of how randomness is introduced in the simulation. This leads to simulated images where storms are

present not having the same consistent shapes as what we saw in the real radar image data. Similarly, the line between what is and what is not a storm is blurred further with this randomness. Accordingly, we will have to take extra efforts to ensure we are able to try to distinguish storms from non-storms.

1. Initial Model

The first thing I did when looking at the simulated data was to use the same method I used on the real data, to see how it might work. I used the exact same setup as the last model made for the real images on the simulated images. I trained on hand selected images that displayed storm-like features as positive examples and selected negative example images from random crops that were filled by at least 25% clouds. I then slightly modified the blurring thresholds, but kept all else the same. The results were very poor, with the model not learning much about the data.

Some of the main problems I determined were the following: First with the large amount of variation among the positive examples, more has to be done to generalize the shape or extract more features from the images to learn something about their structure; Second, I was using a simple model for detecting on the real images, a more complex model could increase the amount of power the model has to make predictions; Third, since there are not many positive examples of storms in the dataset, less than 400 after filtering out examples without a strong signal, I needed some way to create additional samples in order to give the model the ability to identify more patterns throughout the data.

2. Feature Creation

Since the initial attempt at modelling did not work well, more features had to be created. Three additional features were fed into the model, each providing its own insight into different parts of the shape of the cloud formations.

A. Gabor Filter

The first additional feature I created to feed into the model was applying the Gabor filter to each image. The Gabor filter is used for texture analysis, and can be passed over an image in a certain orientation, in order to see if there is frequency content in a given direction. I chose a 45 degree angle for the direction because the texture of the clouds changes the most in that direction and most of them are oriented in that way, especially storms. I kept the rest of the parameters as the default for the opencv implementation. Below is an example of an image with the Gabor filter applied. [9]

Figure 10 An image following the application of the Gabor filter at a 45 degree angle. This indicates there is a lot of changing texture in the direction.

B. Histogram of Oriented Gradients

The next feature that I created was the histogram of oriented gradients for each image. This method gives the "count of occurrences of gradient orientation in localized portions of the image". By passing a derivative filter in both the x and y direction, we get an image gradient value for each pixel in the image. We then divide the image into smaller cells, where in each cell the pixels come together to provide an overall distribution of oriented gradients in that region of the image. So for each small cell we have a relationship between different parts of the image and the amount of change occurring in the image at that point. This is useful for understanding the shape of the image and also the change in intensity and relative direction of change throughout the image. These ideas can be important for determining whether we have a storm occurrence or not, as it would be similar to how we would start to evaluate an image with our eyes. [10]

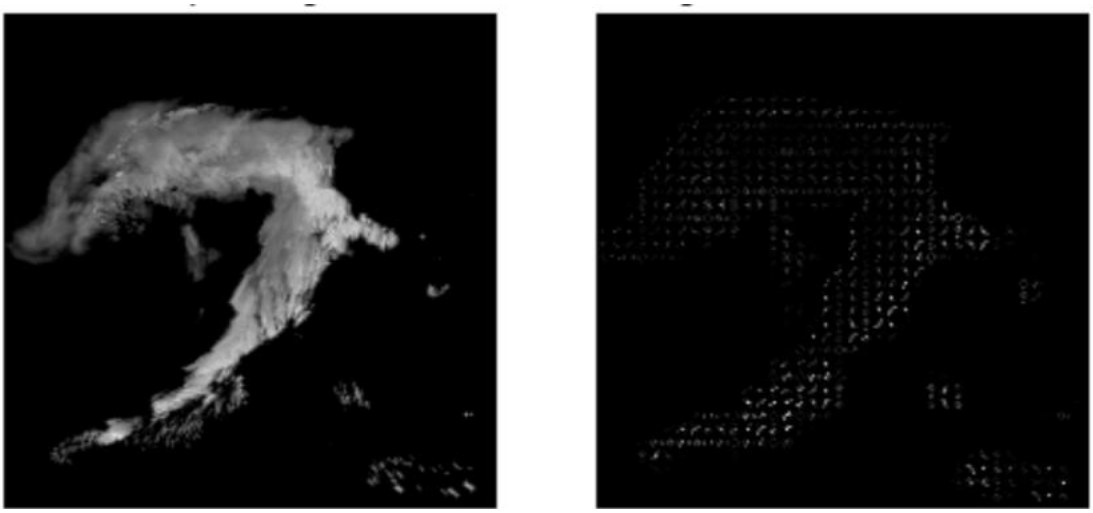

Figure 11 On the left, an image from the simulated dataset, and on the right the result of application of the Histogram of Oriented Gradients method.

C. Canny Edge Detector

The final feature added was the Canny edge detector, a method that extracts the edges from in an image. The method works in multiple steps. First, a Gaussian filter is applied to the image in order to smooth it. Then a gradient filter is applied to the image to find areas of high change in color. Then, a technique called non-max suppression is used to center in on the areas where the most change is detected: these areas are the ones most likely to be the edges of the object in question. Next, a threshold is applied to eliminate any detected points where the gradient is too low. Finally, the detected points of high gradient that are connected strongly are connected and any paths of points that are too small are eliminated. At the end of this process, one has an image where the edges of all objects in the picture are detected. This can be useful as it gives the model an idea of the shape of the storm and also the relative curvature of the cloud and other edge related features. Below is an example of an image before and after the Canny edge detection algorithm is applied. [10]

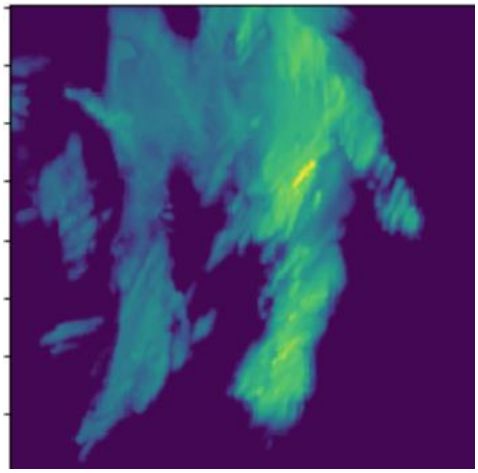

Figure 12 An image of a storm before application of the Canny Edge Detector

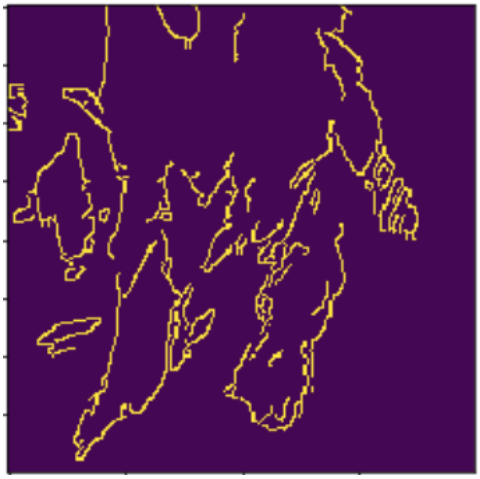

Figure 13 An image of a storm after application of the Canny Edge Detector

3. Modelling Attempts and Framework

A. Subsetting of Data

Since the raw image data was not the most useful format to pass into the model for training and detection, there had to be a way to subset the data to simulate the data that would eventually be fed in through the region proposal system introduced in Chapter 3. One problem that needed to be addressed in this process was that many images in the simulated dataset that corresponded to storms did not have any visual features for detection. It was important to avoid on these images as if there are no discernible features, the model will have difficulty detecting them, if at all. With this in mind, 300 of the 1,000 simulated images that contained storms in the dataset were cropped so that the resulting images had strong features indicating a storm. I then needed to find negative examples so I randomly sampled 200 by 200 regions from the dataset which did not contain storms. I then further partitioned these samples by only keeping those which contained at least 25% of non-empty space to replicate those examples which would be captured by the region proposal algorithm.

B. Creation of Additional Data Samples

Since the amount of data left with after this subsetting was quite small for the desired complexity of the model, I had to come up with a way to create additional data samples from what was left.. First, I translated the images to both the left and right. I also added random noise to each image. This multiplied the amount of available data by four.

C. Transfer Learning

In order to learn more complex features about the cloud formations, we want to use a more complex model. The Keras python package provides a multitude of easy to use, pretrained models which have been previously trained on the ImageNet dataset, one of the most popular image detection challenges in the machine learning community. If we take the weights of one of these models that was previously trained, and remove the final output layer, we have a sequence of layers which extract features from the image provided. Then, we can take the output from this penultimate layer and train our own simple model to our own task, in this case determining whether an image is a storm or not. This process is called transfer learning and is an effective way to combat both lack of data and complexity of a difficult problem. I settled on using the ResNet50 model which is easily implemented through Keras. The architecture of ResNet50 is shown below. It consists of many repeated convolutional layers, with the final output layer being a vector of length 1000. For my purposes, I kept all of the layers except for this final layer, froze the weights of the previous layers to those learned from the ImageNet dataset, and appended my own simple model with one layer of size 100, a layer of size 50 and a single output node with sigmoid activation function. This sigmoid activation function allows for a binary probability to be outputted by the model to give the confidence it thinks a given image is a storm or not.

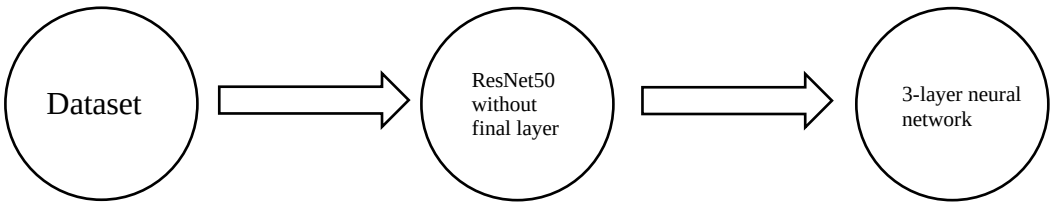

Figure 12 An illustration of the proposed model architecture. The model uses the ResNet50 model, and takes the output and feeds it into a 3-layer neural network which is trained for our task. [15]

D. Training Process

I again divided the data into a training set and a testing set using July as a cutoff. I had 1200 images for training and 800 for testing. The training data had each feature above applied to it, and then these features were stacked on top of each other. I then passed these images into the model described in the previous section, then I predicted on the test set. An example of the stacked training data is below.

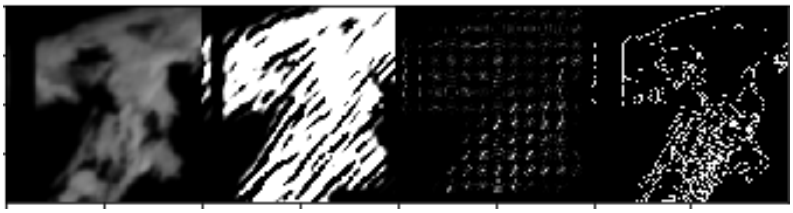

Figure 13 Stacked Training Data. First, the blurred image, followed by the Gabor filtered image, the HoG image, and the Canny edge detector image.

E. Results

The results of training on multiple permutations of the features to determine which worked the best are shown below. In order turn the probabilistic output of the model into predictions a threshold was chosen that most highly divided the positive and negative predictions. This threshold would have to be model specific, so in future experiments, a validation set could be used to set this parameter, and then proceed to make predictions on the test set.

Features	Accuracy
Just blur	71%
All	85%
No Gabor	78%
Blur and Canny	87%
Blur and Gabor	84%
All but HoG	81%

From these results we see that the best combination is the blur and Canny features combined.

.

Chapter 6

Conclusion and Future Work

Based on the aforementioned analysis this model shows great potential in using simulated images for detection of storm events, specifically in the use case of attempting to increase the lead time on detection. By using popular computer vision techniques we can detect storms in these simulated images with reasonably high accuracy.

Some areas that might need to be refined are the region proposal system to crop the images to feed into the model. Since I self-cropped the images to be passed into the model there would need to be a better system for replicating the hand cropping that I did. However, there is definitely promise in identifying the storms in the simulated images using a similar process to the one that I used. Furthermore, there are still many examples of storms that show no clear features and thus were not included in the final training and testing datasets. So the accuracy numbers are a little inflated based off of that.

With this being said, a few avenues to work on in the future would first be the region proposal system. Since I was using k-means, an unsupervised method, there is likely to be a supervised method that could better approximate my hand crops. This way, we could generate better crops to feed into the trained model to make predictions live as new simulations come through.

Also, it would be interesting to examine why there are examples in the dataset that do not show any discernible features of having storms, although they were reported as having storms when cross-referenced with the ground truth information .

Bibliography

1. "Weather Forecasting Through the Ages." *NASA*, NASA, earthobservatory.nasa.gov/features/WxForecasting/wx2.php

2. Xinye Zheng, Jianbo Ye, Yukun Chen, Stephen Wistar, Jia Li, Jose A. Piedra-Fernandez, Michael A. Steinberg and James Z. Wang, ``Detecting Comma-shaped Clouds for Severe Weather Forecasting using Shape and Motion," IEEE Transactions on Geoscience and Remote Sensing, vol. 57, no. 6, pp. 3788-3801, 2019.

3. Mohammad Mahdi Kamani, Farshid Farhat, Stephen Wistar and James Z. Wang, ``Shape Matching using Skeleton Context for Automated Bow Echo Detection," Proceedings of the IEEE Big Data Conference, pp. 901-908, Washington, D.C., December 2016.

4. Thornton, Mark A. "Synoptic Scale Forecasting: A Primer." *Lakeeriewx*, 2005, www.lakeeriewx.com/Meteo361/ResearchTopicOne/SynopticScaleForecasting.html.

5. M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of Machine Learning, Cambridge, MA, USA, MIT Press, 2012.

6. "IBM Global High-Resolution Atmospheric Forecasting System (GRAF): The Weather Company, an IBM Business." *IBM Global High-Resolution Atmospheric Forecasting System (GRAF) | The Weather Company, an IBM Business*, IBM, www.ibm.com/weather/industries/cross-industry/graf.

7. "Artificial Neural Network." *Wikipedia*, Wikimedia Foundation, 1 Apr. 2020, en.wikipedia.org/wiki/Artificial_neural_network.

8. Goodfellow, Ian, et al. *Deep Learning*. The MIT Press, 2017.

9. Mehrotra, Rajiv, Kameswara Rao Namuduri, and Nagarajan Ranganathan. "Gabor filter-based edge detection." *Pattern recognition* 25.12 (1992): 1479-1494.
10. Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. IEEE, 2005.

11. Y. Zhang, S. Wistar, J. Li, M. A. Steinberg and J. Z. Wang, "Severe Thunderstorm Detection by Visual Learning Using Satellite Images," in IEEE Transactions on Geoscience and Remote Sensing, vol. 55, no. 2, pp. 1039-1052, Feb. 2017.

12. Müller, Richard & Jerg, Matthias & Haussler, Stephane & Heizenreder, Dirk. (2018). A Novel Approach for the Detection of Developing Thunderstorm Cells. 10.20944/preprints201810.0728.v1.

13. C. Research Data Archive at the National Center for Atmospheric Re-search and I. S. Laboratory, "Ncar mmm 10-member, 3 km, experimental real-time ensemble prediction system," 2016.

14. Schwartz, C.S., G.S. Romine, R.A. Sobash, K.R. Fossell, and M.L. Weisman, 2019: NCAR's Real-Time Convection-Allowing Ensemble Project. *Bull. Amer. Meteor. Soc.*, **100**, 321–343, https://doi.org/10.1175/BAMS-D-17-0297.1

15. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015

16. Nixon, Mark, and Alberto Aguado. *Feature extraction and image processing for computer vision*. Academic Press, 2019.

17. X. Yuan, J. Zhang, X. Yuan, B.P. Buckles, Multi-scale feature identification usingevolution strategies, Image Vision Comput. J. 23 (6) (2005) 555–563.

18. M. Z. Y. Yasen, R. A. S. Al-Jundi and N. S. A. Al-Madi, "Optimized ANN-ABC for Thunderstorms Prediction," 2017 International Conference on New Trends in Computing Sciences (ICTCS), Amman, 2017, pp. 98-103.

Academic Vita

Kyle Bradley

bradleykjb@gmail.com

EDUCATION

The Pennsylvania State University, University Park, PA
Expected May 2020
Schreyer Honors College
Bachelor of Science in Computational Data Sciences
Bachelor of Science in Applied Mathematics
Thesis: *Machine Learning Applications for Storm Detection and Prediction*

WORK EXPERIENCE

Mars, Inc., Chicago, IL May 2019-August 2019
Data Analytics Intern – Mars Global Services

- Worked as member of the Supply Advanced Analytics Team focused on delivering solutions along the Mars supply chain
- Created a framework using computer vision to detect reference points in pet food images to determine size for quality assurance
- Presented model to associates in other business segments for use in similar projects

Applied Research Laboratory at Penn State, State College, PA May 2018-April 2019
Undergraduate Research Intern, Data Science – Embedded Hardware-Software Systems

- Worked in Department of Defense funded research lab using machine learning to predict diesel ship maintenance events
- Created an ensemble method for handling large amounts of missing data in order to extract more information from historical data
- Developed convolutional neural network to classify helicopter gearbox failure modes achieving 97% accuracy

TECHNICAL SKILLS

Python, R, Scala, SQL, MATLAB, C++

ACTIVITIES

Nittany Data Labs, Executive Board and Training Leader

- Developed and led training for 100+ members of data science student organization
- Project leader of data science projects employing analytical and predictive methods

SHO TIME Mentor, Schreyer Honors College
Competitive Coding Club
Math Club