Local Weather Classification Model Final Report

Jack Akers jackakers@umbc.edu
University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250

Nia Klender niak2@umbc.edu
University of Maryland Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250

Abstract

The main idea of this project was to develop a model which could identify if it was raining or not in the local Baltimore area. Weather prediction is a very difficult task to do accurately and is something machine learning typically gets applied to. Although the purpose of this model isn't to predict the weather, its goal is to be able to accurately identify current weather conditions in different areas. This data could then be used as inputs in other models which would then make predictions of future weather conditions.

1. Introduction and Motivation

Early in the Fall '23 semester, our Machine Learning professor, KMA Solaiman, would always make an analogy when talking about a binomial classification based task saying "is it raining or not". This gave us the idea to build a weather classification model that tells you exactly whether it is raining or not. Initially the idea was to solve this problem using random forests as the course textbook has a weather classification example using a decision tree (Mitchell, 1997) as well as decision trees being used for weather prediction/classification in many research papers (Hasan et al.). In order to differentiate our project from the example in the course textbook, instead of using a decision tree, we would instead use random forests and have significantly more input classes and input data. The choice of using random forests was so we could use a bigger tree and not have to worry about overfitting. Traditionally using a single large decision tree could cause potential overfitting. Random forest addresses this issue by using multiple decision trees and averaging the results. Furthermore each tree is only given a random subset of the data instead of all the data. This randomness and diversity results in a more robust model that is not as prone to overfitting. Once we presented our idea to the professor he recommended us to use multiple different types of detection. So in addition to the random forest and the sensor data collected at different weather stations we would use radar data centered on the Baltimore area and a Convolutional Neural Network (CNN) to identify if it was raining or not from the radar images and then feed that output data into the random forest which would would use the combined sensor data and CNN output data to output the final result. We eventually expanded our scope from classifying if it was raining or not in just one location in the Baltimore area to four different areas in and around Baltimore. Another motivating factor was that there was already a lot of research in this area which is nice because it gives us a foundation to build off.

2. Problem Definition

The goal was to semi-accurately (over 50%) identify if it is raining or not in the following locations in Maryland: Carroll County, Baltimore, Gaithersburg, and Baltimore Martin Airport. We planned to use data from www.weather.us to train and test the models. We were only looking to classify current weather conditions at the current point in time (no forecasting or predicting). Some of the main constraints for this project were data acquisition, computer hardware, and time.

3. Proposed Methods

3.1 Gathering Data

When beginning the project, we wanted to see if there was data available that we could use to build our models. Finding historical weather data was easy, but finding data that also had accompanying radar data for the same point in time that the data was collected from radar station sensors proved to be challenging. What we ended up doing was scraping the data from www.weather.us This website had both weather sensor data and radar data and the data somewhat matched timewise. The only problem was that the data wasn't downloadable so we decided to scrape the data from the site. Data was scraped based on the measurement, so one scraper would be pulling radar images, the other temperature data, another precipitation data, and so on. While collecting, the data was stored in individual JSON files for the measurement being collected. Once collected data would be combined from the individual JSON files into one JSON file which would hold all the data. Once completed this data would be then "cleaned" by removing data instances which were missing data or didn't have an image associated with the sensor data. If possible, data imputation would be used to fill in missing data this would be done with K-Nearest Neighbors (KNN) with a low k value (weather can change at any moment so we don't want to use a big k value). In addition, in this step we would format our labels. We decided to format our labels as an array of 5 elements with each element representing one of the areas that we are looking to classify the current weather condition for and the fifth area being a control value which would be 1 for it is raining in one of the four areas or 0 for it is not raining in any of the areas. Each other element in the array would also be able to take up a value of 1 for it is raining in that area or 0 for it is not raining in that area. This label would mean that our problem is a multi-label binary classification problem with each element in the array being a label with 0 or 1 being the classification for that label. Once the data was "cleaned" the data would be saved into a CSV file which would be used for training and testing the model. When it came to weather sensor data (non radar data) we decided to use the following sensor data from the following weather stations around the Baltimore Maryland area:

- **Pressure**
  - FSNM2
  - Baltimore
  - Gaithersburg
  - SHIP
  - FSKM2
  - BLTM2
  - TCBM2
  - Baltimore-Martin
- **Dew Point**
  - Carroll County
  - Baltimore-Martin
  - Gaithersburg
  - Baltimore
  - SHIP
- **Dew Point Spread**
  - Baltimore
  - Gaithersburg
  - Carroll County
  - Baltimore-Martin
  - SHIP
- **Wet Bulb Temperature**
  - Gaithersburg
  - Baltimore-Martin
  - Baltimore
  - SHIP
- **Temperature**
  - Carroll County
  - FSKM2
  - Gaithersburg
  - TCBM2
  - Baltimore-Martin
  - Baltimore
  - FSNM2
  - BLTM2

When it came to designing the model we would use Pytorch to develop the CNN part and SKlearn to develop the random forest. When it came to selecting the model we would experiment with resnet50, vgg16, and a custom built model that Jack developed in hw3 which achieved 87.3% accuracy on the cifar10 dataset. We would also experiment with ReduceLROnPlateau scheduling and OneCycleLR scheduling and different optimization functions such as Adam, AdamW, and SGD, we would also analyze if different loss functions would affect the model.

3.2 Training

When it came to training we would attempt to use our own machines and if the model used up too much GPU memory we would see if we could run it in Google Colab. In addition, we planned that if Google Colab wasn't working we would lower the image size and train on smaller sized images. When it came to testing changes to the models we would run 100 epochs and based on a graph of average loss over time which would be recorded after each epoch during training we would decide if this model had potential to improve more on higher epochs or if it wasn't complex enough. If we felt like the model could learn more (or further improve) we would retrain the model at higher epoch counts. Once we felt like our model was sufficiently trained we would load the trained model and test images that were associated with the data for the random forests. It would make predictions on this data and those predictions would then be stored in a file. When it came to then testing the random forest, those CNN predictions would be merged in with the test data for the random forest and the model would then be tested on that data.

4. Intuition

We expected this proposed method to work relatively well because decision trees have been used many times before in similar projects and found success and we were going to be using a lot of data of different forms. So when combined with data from the CNN, we expected our model to perform quite well. In addition, the website seemed to have more than enough data and was relatively scrapeable for somewhat easy data collection. We also expected the CNN to perform quite well. We already had experience building CNNs from Homework 3 in the class and were quite confident that we would be able to achieve greater than 80% accuracy if we used a complex model such as Resnet in combination with the random forest. Initially looking at the radar images made us believe that this would be a trivial task for a CNN to solve.

5. Experiments

All experiments were done with a batch size of 32, learning rate of 0.001, and 5 classes unless stated otherwise. Furthermore, trials that used the OneCycleLR scheduler used a max learning rate of 0.01, with steps per epoch equal to the length of the training dataset. Trials that used the ReduceLROnPlateau scheduler used mode = “min”, factor = 0.1, patience = 10, threshold = 0.0001, cooldown = 0, and a min learning rate of 1e-6. Trials that used SGD used a momentum of 0.9. Data was always split randomly 80/20 with 80% being training data and 20% being used for validation. Besides these arguments everything else was left default for PyTorch version 2.1.0 and torchvision = 0.16.0.

5.1 First Trial

Optimizer	Adam
Scheduler	OneCycleLR
Model	Resnet50
Epochs	300
Accuracy	22.91%
Dataset	1,791 Doppler Radar Images Base Reflectivity (dBZ)

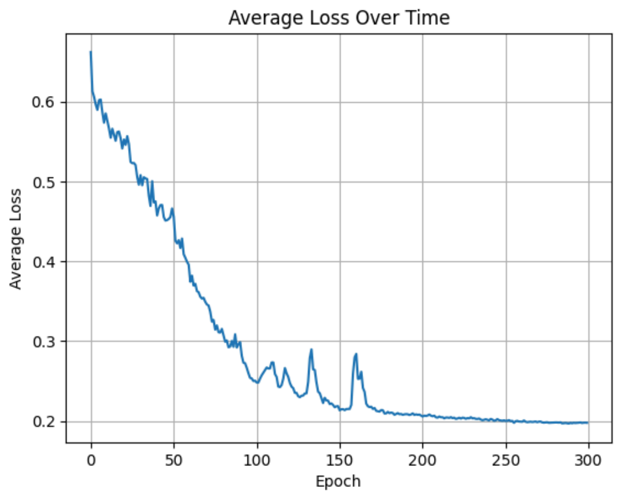

figure 5.1.1: Average loss over time for the first trial

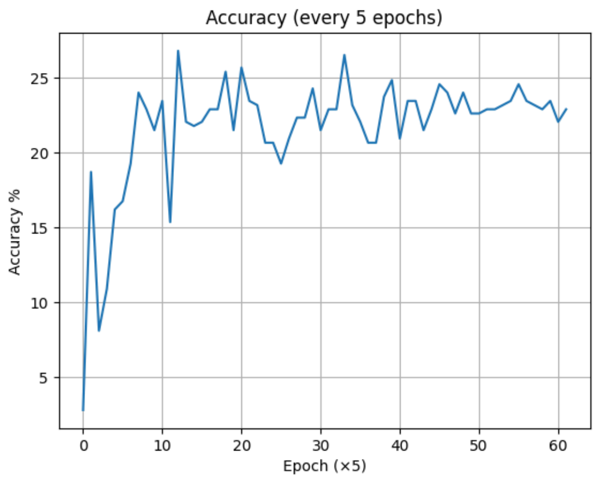

figure 5.1.2: Accuracy over time for the first trial

5.2 Second trial

Optimizer	Adam
Scheduler	OneCycleLR
Model	Resnet50
Epochs	300
Accuracy	27.09%
Classes	4
Dataset	1,791 Doppler Radar Images Base Reflectivity (dBZ)

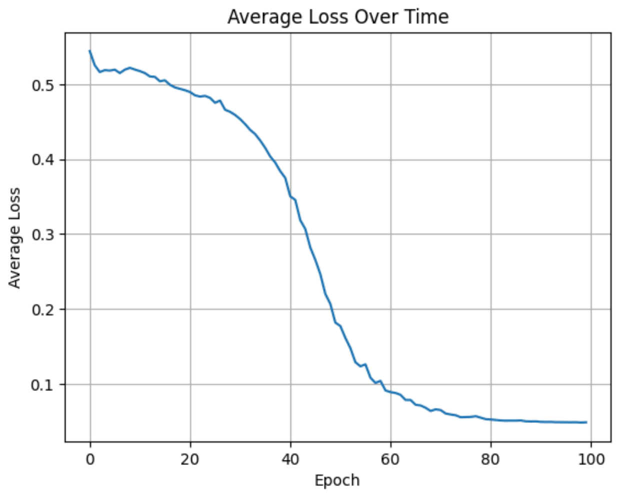

figure 5.2.1: Average loss over time for the second trial

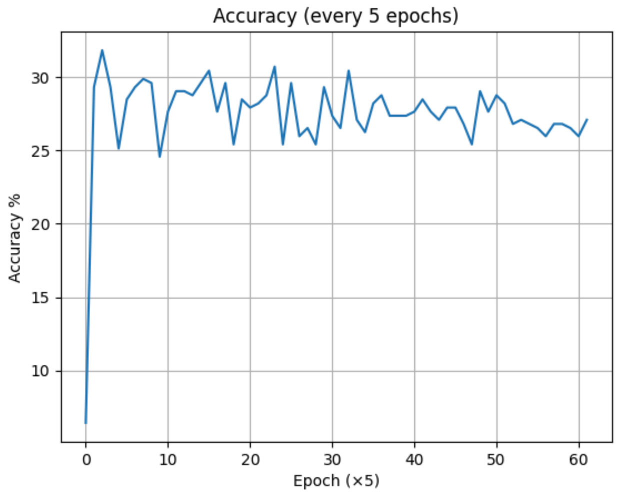

figure 5.2.2: Accuracy over time for the second trial

Due to the increase in performance from decreasing the number of classes from 5 to 4 the rest of the following trials were done using 4 classes instead of 5.

5.3 Third trial

Optimizer	Adam
Scheduler	OneCycleLR
Model	Resnet50
Epochs	100
Accuracy	38%
Dataset	4,314 Hourly Accumulated Precipitation Radar Images.

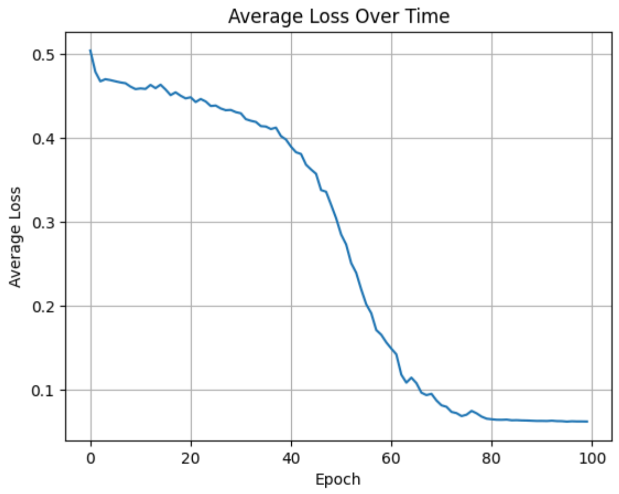

figure 5.3.1: Average loss over time for the third trial

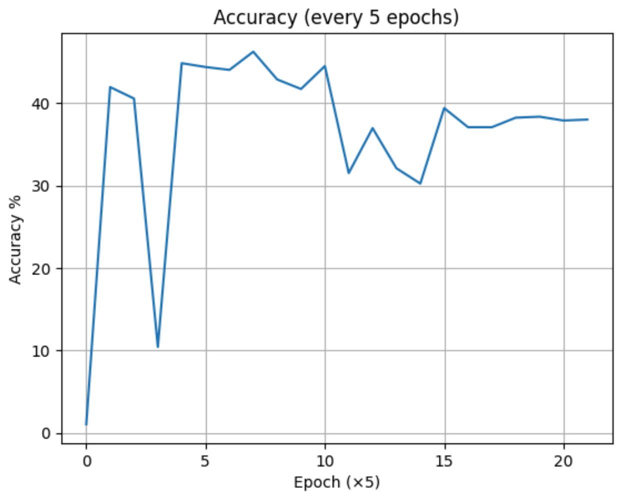

figure 5.3.2: Accuracy over time for the third trial

In this trial new radar images were used which more closely matched the observed precipitation data. In addition the data didn't seem to have as many errors in it and there was a lot more data. This dataset will be used for all other tests unless otherwise stated.

5.4 Fourth Trial

Optimizer	Adam
Scheduler	OneCycleLR
Model	HW3 Model
Epochs	100
Accuracy	33.26%

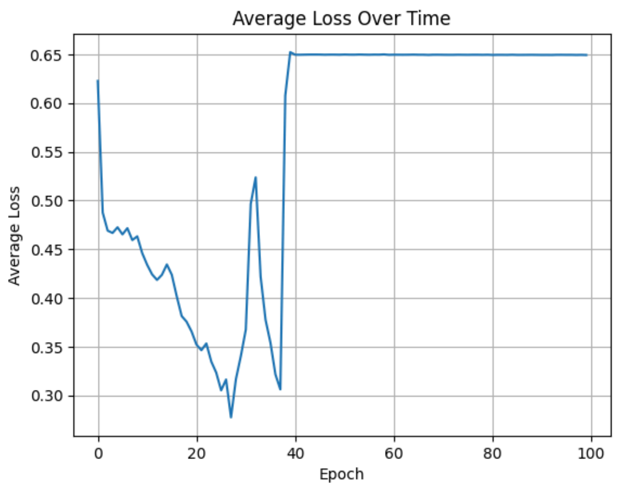

figure 5.4.1: Average loss over time for the fourth trial

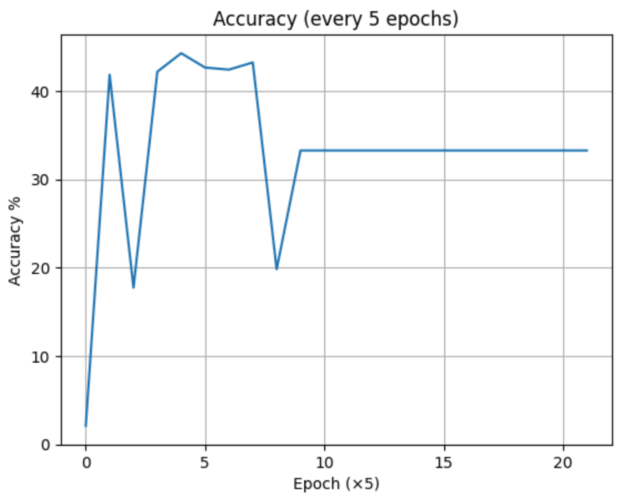

figure 5.4.2: Accuracy over time for the fourth trial

5.5 Fifth Trial

Optimizer	Adam
Scheduler	OneCycleLR
Model	ResNet50
Epochs	300
Accuracy	39.17%

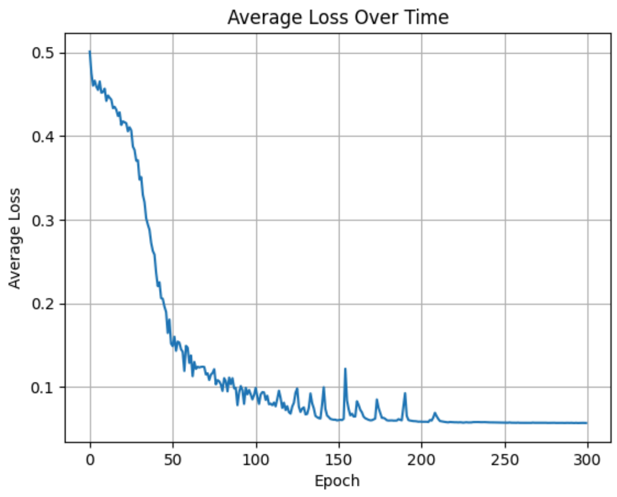

figure 5.5.1: Average loss over time for the fifth trial

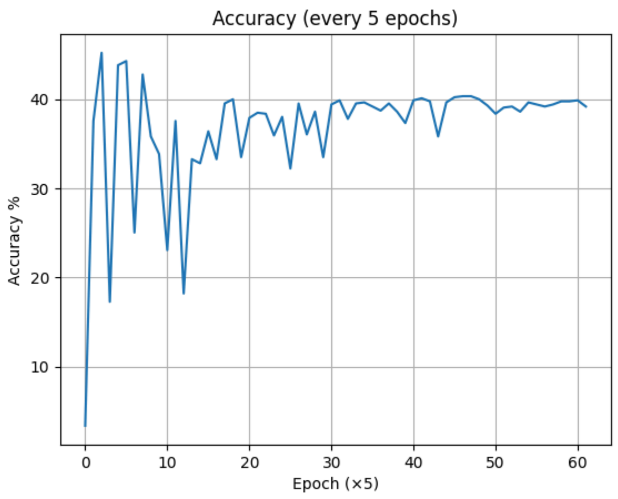

figure 5.5.2: Accuracy over time for the fifth trial

5.6 Sixth Trial

Optimizer	SGD
Scheduler	ReduceLROnPlateau
Model	ResNet50
Epochs	300
Accuracy	39.28%

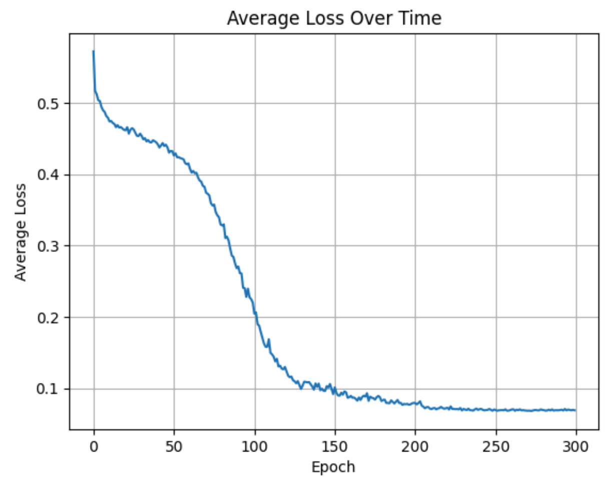

figure 5.6.1: Average loss over time for the sixth trial

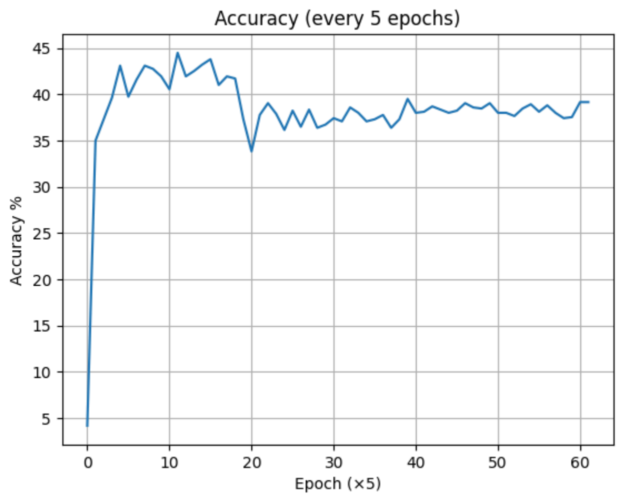

figure 5.6.2: Accuracy over time for the sixth trial

5.7 Seventh Trial

Optimizer	Adam
Scheduler	OneCycleLR
Model	Vgg16
Epochs	100
Accuracy	29.08%

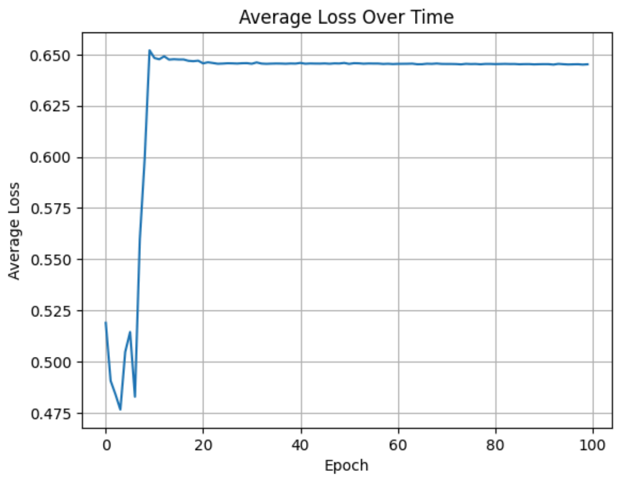

figure 5.7.1: Average loss over time for the seventh trial

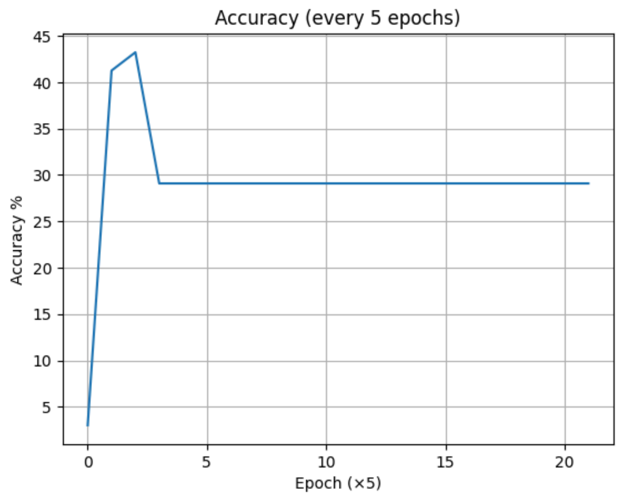

figure 5.7.2: Accuracy over time for the seventh trial

5.8 Random Forest Feature Importance

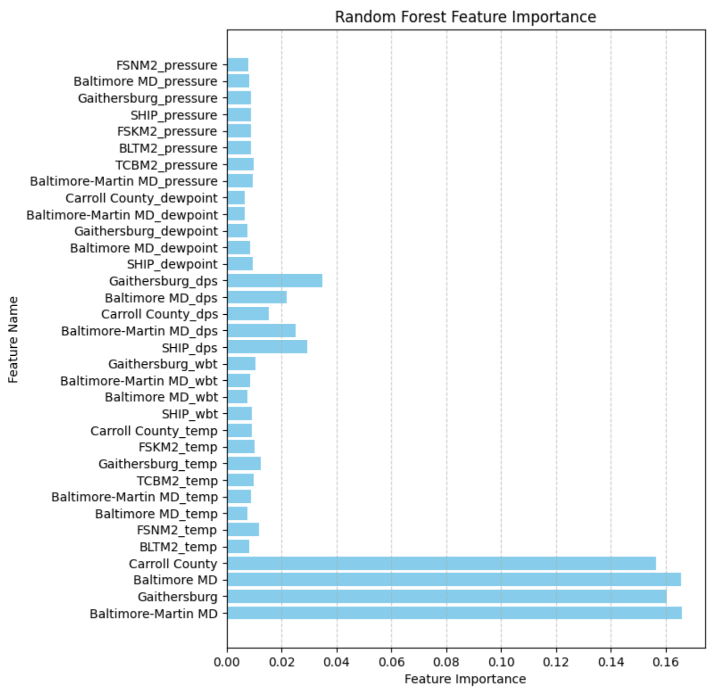

figure 5.8: Random Forest feature importance using the 5.6 model for the CNN and a random forest with 100 estimators. The features Carroll County, Baltimore MD, Gaithersburg, and Baltimore-Martin are the outputs from the CNN for if it is going to rain in that area.

6. Changes after Project presentation

The main change that we made after the project presentation was that we got new radar images. This gave us more data to work with (going from 1,791 images to 4,314 images) that was also more accurate (in that the images more closely matched the associated precipitation data). Despite this, the data still had some issues regarding accuracy. Another change was removing the fifth class that was used to identify if it was raining in any of the other 4 classes. This resulted in a pretty big increase to model performance and was something that we should have tested earlier. We also switched to exclusively using BCEWithLogitsLoss (before we were experimenting with MultiLabelSoftMarginLoss), as well as testing the vgg16 model and trying to get a model called "RainNet" to work. RainNet is a Convolutional Neural Network for radar-based precipitation nowcasting, which is very similar to what we were doing (Ayzel et al., 2020). Unfortunately, we were having trouble getting it working in Pytorch (it was made in Keras). Another problem is that it took images that were 928 x 928 pixels in size which meant that we would need to upscale our images, and we would probably have problems training the model because of how much GPU memory it would take up. We also re-ran all of our testing a second time with the new data and got more graphed data so we could have more visual aids for this paper. We also looked at the random forest more closely and did feature analysis, which revealed that feature selection was a major cause of our model's low performance.

7. Conclusions

In conclusion our model is not as accurate as we would like, only achieving an accuracy of 39%. However, we have learned a lot and we feel that we know what the problems are with the model and why our results were not as good as we had hoped. First, our data quality wasn't as good as we would have liked. Some radar images are blank, which would suggest no rain, but the accompanying precipitation data says that it is raining. In other cases the radar shows images which would suggest a catastrophic storm, yet the accompanying precipitation data suggests no rain. We were able partially address this issue after our presentation, which is shown in the difference in model accuracy from 27.09% to 38% after getting better quality images. It is also apparent that Resnet50 seems to be the best current model design that we have tested so far and that using this model would probably be the best move for future work. We also found out why our random forest didn't seem to make much of a difference when it came to classification accuracy. As can be seen in figure 5.8, the feature selection wasn't adequate. It shows what measurements don't have anything to do with indicating if it is raining or not. This information is useful because we can improve model performance drastically by using dimensionality reduction and removing most of these features. This insight shows that we or others should look for different features in future work. This feature importance graph also explains why the random forest result is essentially the exact same as the output from the CNN. Due to most of the features being largely irrelevant in determining if it is raining or not, the random forest largely relied on the CNN classifications when making its decisions. Overall out performance although not as high as we would have liked is pretty good. Others who have tested other machine learning methods also ran into similar problems with obtaining data and their overall accuracy not being as high as professional weather forecasting services (Holmstrom et al., Machine Learning Applied to Weather Forecasting).

References:

AbdulRaheem, Muyideen, et al. “Weather Prediction Performance Evaluation on Selected

Ayzel, G., Scheffer, T., & Heistermann, M. (2020). RainNet v1.0: a convolutional neural network for radar-based precipitation nowcasting. Geoscientific Model Development, 13(6), 2631–2644. https://doi.org/10.5194/gmd-13-2631-2020

Hasan, Nasimul, et al. “Automated Weather Event Analysis with Machine Learning.” IEEE

Holmstrom, Mark, et al. Machine Learning Applied to Weather Forecasting. 15 Dec. 2016. https://cs229.stanford.edu/proj2016/report/HolmstromLiuVo-MachineLearningAppliedToWeatherForecasting-report.pdf

Know your sky. Weather.us. (n.d.). https://weather.us/

Machine Learning Algorithms.” IAES International Journal of Artificial Intelligence (IJ-AI), vol. 11, no. 4, 1 Dec. 2022, p. 1535, https://doi.org/10.11591/ijai.v11.i4.pp1535-1544.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

National Oceanic and Atmospheric Administration. (n.d.). *How reliable are weather forecasts?*. SciJinks – All About Weather. https://scijinks.gov/forecast-reliability/

Xplore, 1 Oct. 2016, ieeexplore.ieee.org/abstract/document/7856509.