Implementation of Deep Neural Network Algorithm for Dual Polarization Radar Product Classification and Prediction

Adubi Tunde
Colorado State University, Fort Collins, Colorado 80523 United States

The evident success of the convolutional Neural Networks (CNN) in computer vision, speech recognition, image classification and prediction has been gaining immense and continuous relevance in medical imaging, communication systems, remote sensing, in military and defense, for prediction, estimation and security purposes. By harnessing the CNN model, a type of Deep Neural Network (DNN) algorithm, we can extensively utilize deep learning techniques to accurately recognize, predict and classify images from radar fields.

In this work, the task is to design a heuristic CNN model for effective recognition, prediction and classification of storm observations from weather radar images. The feature extraction properties as well as the classification layers of this model would be trained, and validated with six different types of polarimetric radar field (observed storm).

The data used was obtained from the ARMOR radar observed on the 11th of December 2021. I analyzed the model with a testing dataset from the six classes which were Reflectivity (Z), Differential Reflectivity (ZDR), De-aliased Doppler Velocity (DVEL), Doppler Velocity (VEL), Differential Phase (PHIDP), and Cross Correlation Ratio (RHOHV) when training was completed.

This work was implemented with100 data samples for each of the radar fields of which 80 percent was to be used as training data set, while 20 percent of this dataset was generated for validation. In addition, 20 percent of the original data was labelled as the test datasets. Hence, a total of 600 radar field images were used in this project.

Upon dataset generation, the images were labeled and subdivided into training, test, and validation datasets. The task was completed by preprocessing images with MATLAB, where I resized, normalized and augmented the input image to fit the Model. The Google Collaboratory open source Integrated Development Environment (IDE) software was used for the execution of this work. Augmentation of both the train and validation dataset was carried out. I designed a CNN model with two convolutional layers which was alternated with a corresponding max-pool layer, a flatten layer, a drop out layer sandwiched by two dense layers.

With a training accuracy of 98.00%, the test and validation accuracy were 97.5% and 98.9% respectively. Overall, the experimental results indicate that the algorithm could almost always classify, recognize and predict the different classes of radar storms observed as evident on the confusion matrix.

TABLE OF CONTENTS

1 INTRODUCTION

In recent times, the most commonly used models for signal identification lies between the statistical pattern recognition otherwise called the Feature Based (FB) extraction and the decision theory approach which is also known as the Maximum Likelihood Based (LB) method. The Bayesian theory, a method of Maximum Likelihood Based (MLB) is best used for achieving approximate decisions instead of optimal judgment [1]. Unlike the MLB approach where the classifiers have poor computational complexity owing to huge numbers of buffered samples, the feature extraction approach extracts certain features from received signals without prior information. One of the most advanced type of the DNN algorithm used for identification, classification and prediction of images is the CNN model. This is due to its effective ability in discriminating between input images through robust classifiers after assigning learnable features and weights to the input data. Repeatedly, this supervised learning exhibited by the CNN model have shown to be a quite reliable tool in the extraction of mid-level and high-level characteristics which are normally utilized for adequate identification and image recognition [2].

Deep Learning (DL) has been gaining prominence in satellite communications where Jean et al [3] show that DL algorithm when adequately trained, then the implementation of scalable, inexpensive and accurate economic survey with satellite imagery in developing countries can be achieved. Likewise, in weather forecasting where CNN can be used to interpret output of numerical weather prediction automatically [4]. Also, Tripathi et al [5] was able to implement DL for downscaling. Currently, improved weather forecasting is being generated and achieved by DNNs [6]. Although these advances in applying DL techniques in nowcasting models, to predict future images of precipitation given a sequence of past images [7] are encouraging, but not much work has been put forward to design independent CNN Models that could recognize and predict radar field scans without which the overall quantitative precipitation estimation would not be achieved.

Similarly, related works in image prediction either in medicine, military or signal classification have relied heavily upon transfer learning-based CNN models, where pre-trained networks were harnessed. [8] used the AlexNet, VGG, and GoogleNET models that performed well on ImageNet dataset during the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) for fungus image classification. Also, in [9], a pre-trained CNN based model was utilized to demonstrate image visualization. While it is easier to use these types of architectures for large number of natural images as seen in ILSVRC, the performance of pre-trained CNN models on few randomly generated computer radar scans could be lower. Hence, a critical need to design a DNN architecture that could classify and predict dual polarization radar products image. In this project, I proposed and designed a CNN model that could accurately classify, and predict radar field images while avoiding over fitting and convergence problems that could arise from pre-trained models with these same datasets. In addition, we avoided complexity in the network structure but enhanced the performance by harnessing the appropriate optimizer and hyper-parameters. Training speed was augmented by the Tesla K80 GPU. The rest of this work is organized as follows. Section II describes the datasets. The method of work done was explained in section III. Results and discussion were mentioned in the section IV. Section V contains the conclusion.

2 DATASETS

The image data utilized in this exercise was curated from the ARMOR radar scans observed for the convective storm at Huntsville, Alabama on December 11th, 2021. Although the radar scanned for 24hours period, data between 11 UTC to 18 UTC were selected for this work. I made Plan Position Indication (PPI) plots from the radar scans. Radar product scans for Z, ZDR, PHIDP, RHOHV, DVEL, and VEL were plotted for the selected time frame. For the six different classes, 600 images of these field were curated with 100 images for each field. While 384 images were listed as training data, the validation and training dataset had 96 and 120 images respectively. As seen below, the PPI plots were made at an elevation angle of 0.7 degrees, and at about 15:01:55 UTC. The images had a dimension of 600 X 496 pixel in the RGB format before preprocessing.

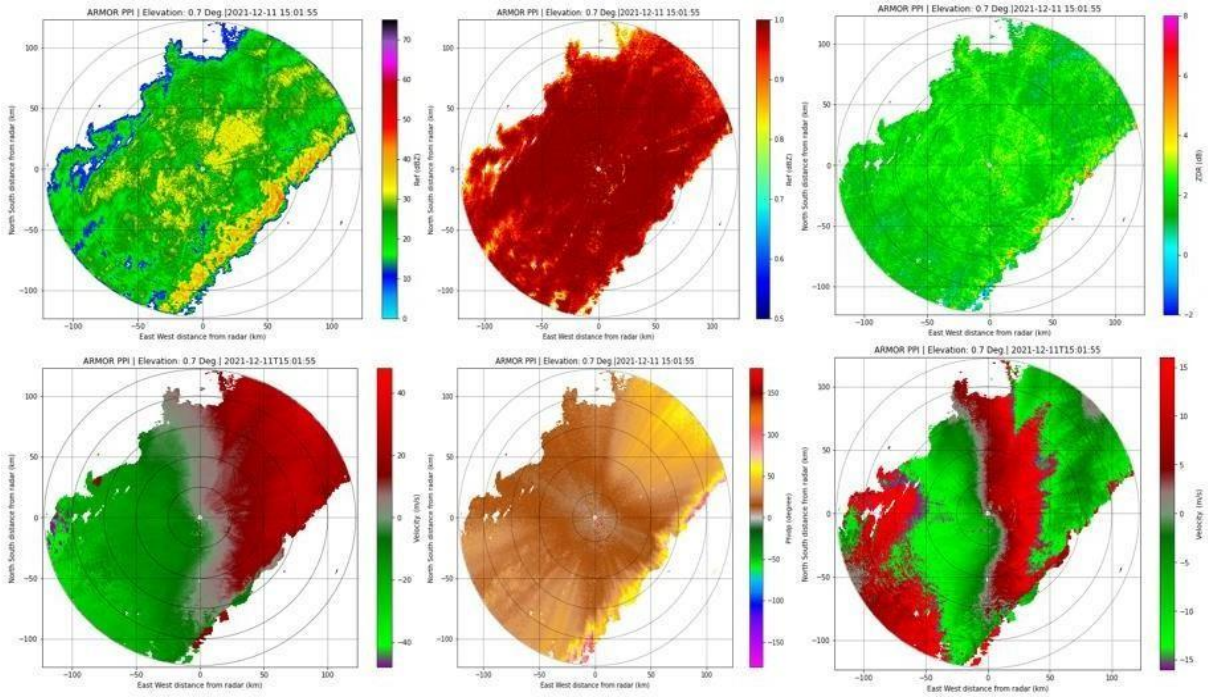

Figure: 1 ARMOR Radar Datasets

2.1 Data Preprocessing:

Data resizing and processing is a critical step just before the implementation of the machine learning (ML) model on our dataset. It involves the removal of any unwanted features which were derived from the simulation process and would not reliably be in necessary in training the system. It is an important process towards making dataset usable in the ML environment. I wrote the MATLAB script for this data resizing and processing stage. In addition, the data files were transformed from 600 X 496 pixel to 227 X 227 pixel, a fixed resolution in which our CNN algorithm can work with. The image in the figure below is the processed version of the PPI scans for Reflectivity, Differential reflectivity, Velocity, De-aliased velocity, Differential phase, and Cross correlation ratio fields for 15 UTC. This image was derived from the figure above. It clearly shows that both the title and labels associated with the plots have been eliminated as well as other unwanted features.

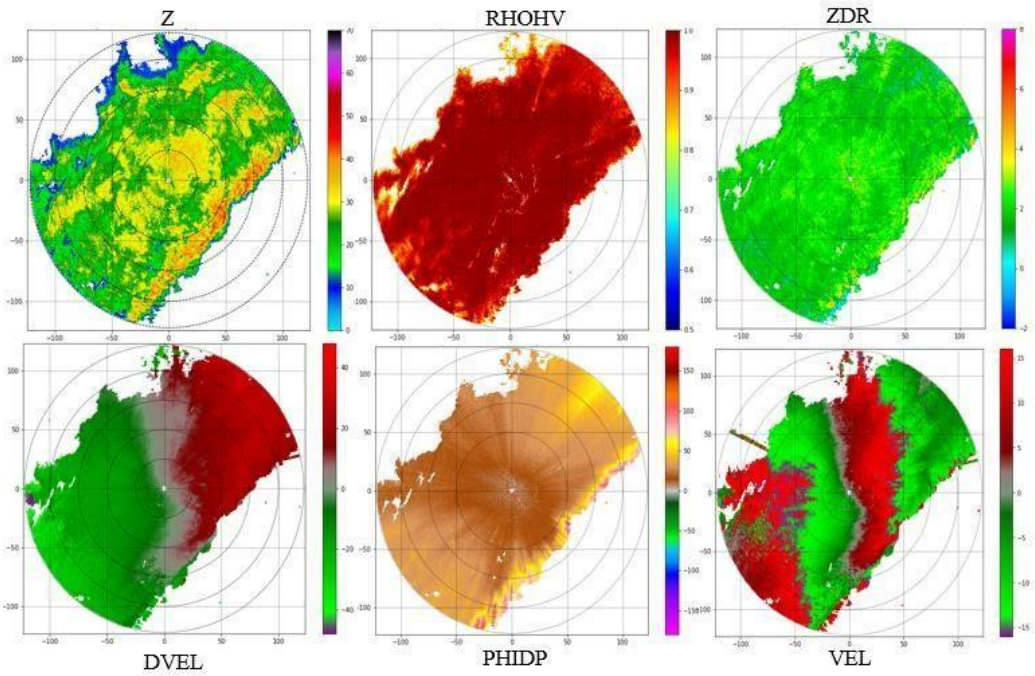

Figure 2. Processed Image datasets

2.2 Normalization

Before the training process began, I normalized the characteristics of the datasets in order to keep them in the same range of 1 and 0. The Minmax scaler was applied to reduce the variability in pixel range spanning from 0 - 255 of the input image. This helps to prevent challenges during the learning phase of the model while it also increases the convergence rate of gradient descent.

$$\boldsymbol{k} = (\boldsymbol{X} - \boldsymbol{min}(\boldsymbol{X}))/\boldsymbol{max}(\boldsymbol{x}) - \boldsymbol{min}(\boldsymbol{x}) \quad (1)$$

2.3 Augmentation

I modified the training datasets by implementing some changes on the image as this could improve the generalization and robustness of our model. These augmentation parameters include but not limited to brightness, rotation, sharing, flipping, and zooming. The test dataset was not manipulated as they are not needed to better the model's performance.

3 METHODOLOGY

3.1 Convolutional Neural Network (CNN) Model

The CNN model consists of stacks of layers that adopts convolutional mathematics to perform supervised learning on a large number of images (datasets). This learning helps the model to train and recognize our six different classes of radar fields through prediction from the learnt patterns. Unlike the neural network where neurons are interconnected, the presence of the varied multilayer perceptron that contains one or more convolutional layers entirely connected, supports the CNN model high accuracy for image recognition and widely used for weight sharing. As seen below, the architecture of our CNN model comprises of several layers. They include the input layer, two convolutional layers embedded with ReLU activation functions, two max-pooling layers, kernel filters, a flatten layer, fully connected layers, and the SoftMax activation function which classifies the fields at the output.

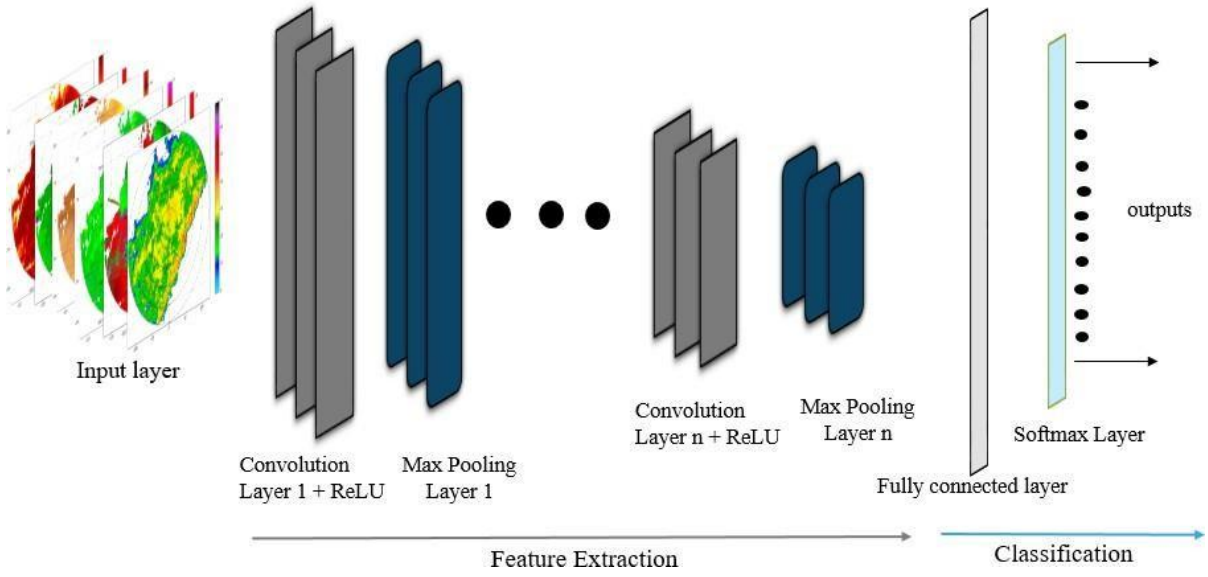

Figure 3. Architecture of the CNN Model

The mathematical expression for output feature maps of the convolutional layer is presented below. Where the elements of rows and columns of the resultant matrix is denoted by m and n, when f is the input image being convolved with kernel filter h.

$$\boldsymbol{G[m,n] = (f * h)[m,n] = \sum\sum h[j,k]\, f[m-j, n-k]} \qquad (2)$$

I present the description of these layers seen in the CNN architecture for better understanding of how the feature extraction, and classification of the datasets are achieved.

3.2 Model Architecture

In this work, the input image size was 227 X 227 X 3. A 2-D grid-like radar image is accepted at the input layer. The dataset that exhibits strong spatial and temporal dependencies are commonly used for this purpose. The colors of the image are captured in form of a 3 dimension which

creates a color image I consisting of H rows (height), W column (width) and D channels (depth) representing three colors which are red, green and blue.

I designed the model using two convolutional layers (conv Layers) each embedded with a kernel size of 3 X 3 while having 20 filters, and 40 filters for the first and second layer respectively. The Rectified Linear Unit (ReLU) activation function was applied directly in each Conv Layer to introduce non-linearity to the activation map threshold at 0. The Conv Layer is considered as the most important layer that uses feature maps and 3x3 size filters in order to learn features from the input datasets. The filters are applied to the input image of the previous layer so that 2-dimensional convolutional operation between the input and feature maps is achieved at the output. These stack of new feature maps that are produced are fed on to the next layer until the last output layer. The filter present at the output is used to learn specific properties of the input datasets and would be able to recognize such parameters if it sees them again.

The max-pooling was performed with a 2 x 2 filter which helps to reduce computation and the spatial size of the input image from the previous conv layer. I flattened the output of the second conv layer into a sing array before it was fed to the first fully connected (dense) layer embedded with 100 hidden layers and a ReLU activation function. A dropout layer of 20% was included between the two dense layers to reduce over fitting. The Model's output was from a dense layer fitted with six different nodes. The SoftMax activation function was used for the classification of this storm observations from the radar.

3.3 Training Model Parameters

Upon designing the CNN architecture, I compiled the model with the categorical cross-entropy loss, and the Adam optimizer to reduce loss function. The metrics accuracy was applied to judge the model performance. In addition, the training was completed in 10 epochs, with a batch size of 10, using the default learning rate of 0.001. The training process was controlled by the callback objects where the model checkpoint saves the best trained model at certain intervals in model.h5, and the early stopping monitors the minimum validation loss and determines when training would be stopped. Shown in the table below are the hyper parameters that was used in this work. Furthermore, I depicted my work in seven stages, as shown in the block diagram where the procedure started from image generation from the ARMOR radar scan to designing the training model, and evaluating the model's prediction performance from the obtained confusion matrix.

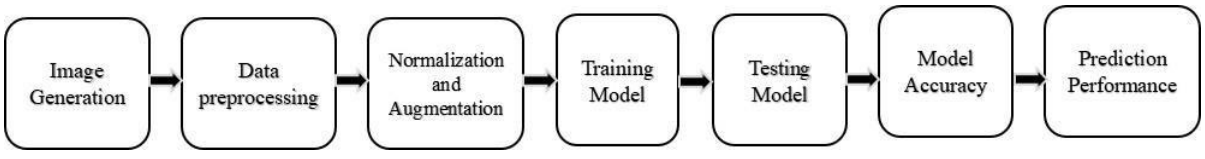

Figure 4. Procedural diagram for ARMOR radar field image classification and prediction

Table 1. Hyper parameters for our CNN model

Parameters	**Value**
Batch size	10
Epoch	10
Iterations	38
Validation steps	10
Learning rate	0.001
Verbosity	1
Optimizer	Adam
Metrics	Accuracy
Loss function	Categorical cross-entropy

4 RESULT AND DISCUSSION

The training result from my model showed that I initially had an accuracy of 93.58% with a validation accuracy and loss of 94.44% and 0.1298 respectively on the 1st epoch. At the end of the 10th epoch, there was an increase in both the training accuracy where I recorded a 97.58%, and a 98.88% validation accuracy was obtained with decrease validation loss of 0.0541. The unlabeled test datasets that the model had no previous knowledge of was used to evaluate our architecture's performance. Here, I obtained a test accuracy and loss of 93.33%, and 0.152 respectively. This indicated that my model had a strong likelihood of predicting the true label radar field images. Seen in the figures below are the training and validation accuracies of our CNN model with their corresponding loss against each epoch.

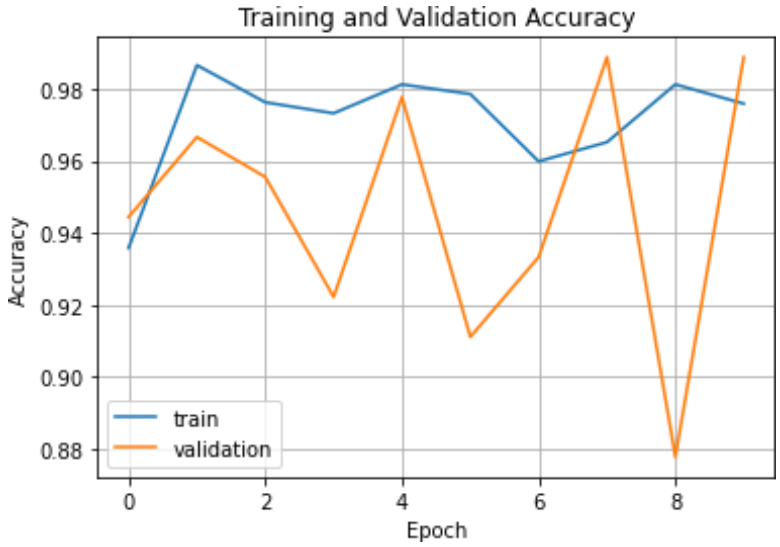

Figure 5. The CNN Model Accuracy for Training and Validation Dataset

Figure 6. The CNN Model Loss for Training and Validation dataset

4.1 Confusion Matrix

The 6 x 6 confusion matrix was further used to verify the performance of my model where the test datasets with 20 images from each of the six radar field classes was distributed between the predicted and actual labels. Hence, I determined the accurate number of predicted outputs for each radar class. The figure below shows the accuracy of my model at a glance by considering the diagonal elements of the confusion matrix. I obtained a 98.3% accuracy by dividing the sum of diagonal elements by the sum of total elements in the matrix. In other words, the model correctly predicted 118 radar field images from the 120 images in the test datasets. As seen, the model was 100% accurate when predicting all the samples from the cross correlation, De-aliased velocity, Differential phase, Differential reflectivity, and the Reflectivity radar field. Although the CNN model had difficulty in predicting all the samples in the Doppler velocity field which had 18 samples as true positive and two as false positive, this was quite understandable. The strong resemblance between the Doppler velocity and De-aliased velocity field was most like why the model placed the two samples as De-aliased velocity field images.

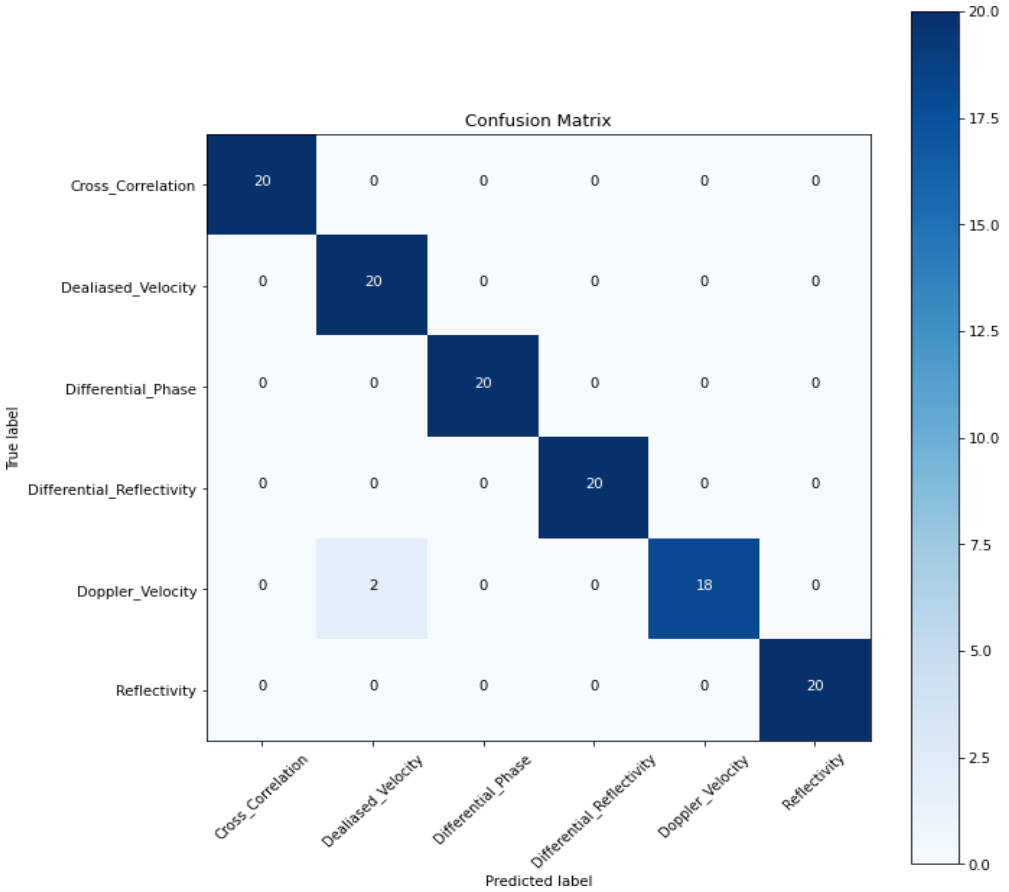

Figure 7. Confusion matrix of predicted result from test dataset

The table below show the classification summary report from the confusion matrix algorithm. In this table, the precision of predicted de-aliased velocity images was 0.91 while that of the other classes were 1. The recall of all true label classes was also one except that of the Doppler velocity class with a value of 0.9. Support for each radar field class was 20 samples in the dataset. The F1-score for both de- aliased and Doppler velocity was 0.95 with other classes having values of one. I obtained an overall accuracy of 98.0% seen during training. The mathematical relationships for the precision, recall and F1- score are given below.

$$\boldsymbol{Precision} = \frac{\boldsymbol{True\ Positive}}{\boldsymbol{True\ Positive + False\ Positive}} \tag{3}$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positive}}{\boldsymbol{True\ Positive + False\ Nagetive}} \tag{4}$$

$$\boldsymbol{F1\text{-}score} = \mathbf{2}\left(\frac{\boldsymbol{Precision \times Recall}}{\boldsymbol{Precision + Recall}}\right) \tag{5}$$

Classes	Precision	Recall	F1-Score	Support
Cross correlation	1.00	1.00	1.00	20
De-aliased velocity	0.91	1.00	0.95	20
Differential phase	1.00	1.00	1.00	20
Differential Reflectivity	1.00	1.00	1.00	20
Doppler velocity	1.00	0.90	0.95	20
Reflectivity	1.00	1.00	1.00	20
Accuracy	0.98			120
Macro average	0.98	0.98	0.98	120
Weighted average	0.98	0.98	0.98	120

5 CONCLUSION

In this project, I designed and implemented a CNN-based DNN model that could accurately classify and predict images from the ARMOR polarimetric radar field during storm observations. Although I worked with a limited dataset, I ensured that I avoided error due to under fitting as well as overfitting by selecting adequate parameters for the design purpose. The network was designed with 20 and 40 filters in the two convolution layers respectively. The kernel size was a 3 by 3 filter, while the ReLU activation function was used.

I optimized, complied, and fitted the model using metrics accuracy, categorical cross- entropy loss, and Adam - an effective learning discriminator of classifiers. The network was trained in 10 epochs, after which I evaluated the test data to see how the network performed in image classification. I plotted the model accuracy and loss to observe the overall behavior of both the validation and training dataset, and finally showed how it corresponded with the confusion matrix for prediction estimation. I executed the training model with tensor flow 2.8.0, an open source python library used to implement large scale ML models. The Keras high level deep learning API of Tensor Flow made the design, development and evaluation of my model easier as it permitted us to utilize the NVIDIA Tesla K80 GPU for this work.

6 REFERENCES

[1] C. Yang, Z. He, Y. Peng, Y. Wang, and J. Yang, "Deep Learning Aided Method for Automatic Modulation Recognition," *IEEE Access*, vol. 7, pp. 109063–109068, 2019, doi: 10.1109/ACCESS.2019.2933448.

[2] S. Rajendran *et al.*, "Classification With Distributed Low-Cost Spectrum Sensors," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 3, pp. 433–445, 2018.

[3] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, "Combining satellite imagery and machine learning to predict poverty."

[4] P. R. Larraondo, I. ˜ Naki Inza, and J. A. Lozano, "Automating weather forecasts based on convolutional networks," 2017.

[5] S. Tripathi, V. V. Srinivas, and R. S. Nanjundiah, "Downscaling of precipitation for climate change scenarios: A support vector machine approach," *J. Hydrol.*, vol. 330, no. 3–4, pp. 621–640, Nov. 2006, doi: 10.1016/J.JHYDROL.2006.04.030.

[6] X. Shi, Z. Chen, and H. Wang, "Convolutional LSTM Network : A Machine Learning Approach for Precipitation Nowcasting arXiv : 1506 . 04214v2 [cs . CV] 19 Sep 2015," pp. 1–12.

[7] S. Samsi, C. J. Mattioli, and M. S. Veillette, "Distributed Deep Learning for Precipitation Nowcasting."

[8] Q. Li, Institute of Electrical and Electronics Engineers, and IEEE Engineering in Medicine and Biology Society, *CISP-BMEI 2017 : proceedings, 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics : 14-16 October 2017, Shanghai, China*.

[9] S. Giri and B. Joshi, "TRANSFER LEARNING BASED IMAGE VISUALIZATION USING CNN," *Int. J. Artif. Intell. Appl.*, vol. 11, no. 4, 2019, doi: 10.5121/ijaia.2019.11404.