Multi-Class Weather Recognition from Still Images Using Different CNN Models

Pahuldeep Singh Dhingra
Department of Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, Florida, USA
pdhingra2024@fau.edu

***Abstract*—In the last few years, much advancement has occurred in the field of Deep Learning, which also includes a very effective and practical approach to image processing, Convolutional Neural Networks (CNNs) [1]. In order to perform Multi-Class Weather Classification from single images, different CNN models performed exceptionally well on that task [1]. In this study, we built a custom model with basic CNN architecture and compared its performance parameters with existing pre-trained models like VGG16, ResNet50 and MobileNetV2. For this, we used a comprehensive dataset of still weather images that includes four classes: Cloudy, Rain, Shine, and Sunrise, and further conducted experiments to evaluate the performance of each model based on numerous parameters [1].**

Keywords—Deep Learning, Convolutional Neural Network, Multi-Class Weather Classification, VGG16, ResNet50, MobileNetV2

I. Introduction

In this study, we are going to perform Multi-Class Weather Classification using Still Images. Over the past few years, it has become very crucial to have knowledge of accurate weather as different weather conditions directly influence the working of visual devices and systems like video surveillance and self-driven vehicle systems [6] [3]. Not only that, but sudden climate change also affected transportation routes, which ultimately led to car accidents, ship collisions, train derails, and even plane crashes [1]. Moreover, there are so many activities in the agriculture field, such as crop cultivation and harvesting, which are directly affected by different climate and weather conditions [1]. So, keeping all these scenarios in mind, it is of utmost importance to keep track of different weather conditions. Initially, in the past times, this task of weather classification was done either by human vision or some expensive sensors. Further, with the advancement in the field of Artificial Intelligence and Deep Learning, it became possible to forecast the weather, but that was not enough as it led to inaccurate results due to sudden climate changes and the greenhouse effect. Finally, this simple yet effective approach of Weather Classification from still images takes place. So, we are using different CNN models to perform Multi-Class Weather Classification [13] [1].

In order to handle this critical issue of Multi-Class Weather Classification, various researchers initially proposed a lot of methods. According to the researchers [2] [3] [4], the classification is performed only on those images captured by the vision systems in the vehicles, thus resulting in less generalization of the model. After that, the authors of [5] proposed a model that can classify weather images only into two categories: Sunny and Cloudy. Then, the researchers of [6] used different mathematical approaches to extract image features, such as pixel intensities, histogram of orientation gradients (HOG), and others. In short, they created the approach of multiple kernel learning in order to get the adaptive classifier. Then, one of the researchers [8] used image segmentation on the raw images to extract only the sky portion from the images. Later, the framework of multiple kernel learning was implemented to get the desired classification. Apart from that, there are some other researchers [9] who implemented the approach of a unique ensemble method called SAID in which they used KNN, Random Forest and Naïve Bayes models together for the correct classification.

This research deals with the critical problem of accurately predicting the multiple weather classes using a Convolutional Neural Network. Our approach is divided into two sections. Firstly, we are creating a custom Convolutional Neural Network (CNN) model using different convolutional layers, pooling layers and dense layers. Secondly, we are using three pre-trained models: VGG16, ResNet50, and MobileNetV2, which will be fine-tuned according to the weather dataset and then trained in order to perform classification [7]. Here, we consider the usage of pre-trained models just to simply avoid the optimization of CNN parameters. This is because these already existing pre-models are very powerful in extracting standard meaningful features of any domain. In this study, we are using a weather dataset collected from the Kaggle website that contains four classes: Cloudy, Rain, Shine, and Sunrise. After performing model training, we analyze all the models in order to compare the results of basic CNN to those of state-of-the-art pre-trained CNN models [1] [10] [11] [7].

II. Methodology

Our methodology first begins with collecting a suitable dataset of images consisting of different weather occurrences. That dataset was taken from the Kaggle website, which contains around 1125 weather images. The images are classified into four main categories: cloudy, rain, shine, and sunrise. After loading, the entire dataset goes through the stage of data analysis and preprocessing. Further, the dataset is split into training and testing sets, and the training set is used to build and train the model, which is later evaluated using the test set [1] [10]. Finally, the performance parameters are observed, based on which we can further fine-tune the hyperparameters and once again retrain the entire model for better optimization and performance. These steps are firstly followed by the custom CNN model and then later by all three pre-trained models: VGG16, ResNet50, and MobileNetV2. After doing all the model training and evaluation, all CNN models are compared based on their performance metrics.

A. *Dataset Description and Preprocessing*

The name of the dataset is ‘Multi-Class Weather Classification”, and it is collected from the website of Kaggle. This dataset contains a total of 1125 RGB images, consisting of different weather occurrences in multiple locations. The entire dataset is divided into four classes: Cloudy, Rain, Shine, and Sunrise where each class contains a different number of images as follows: the Cloudy class contains 300, the Rain class contains 215, the Shine class contains 253, and the Sunrise class contains 357 images. This entire split of images is shown in Fig. 3. All images consist of either of the two formats: .jpg and .jpeg.

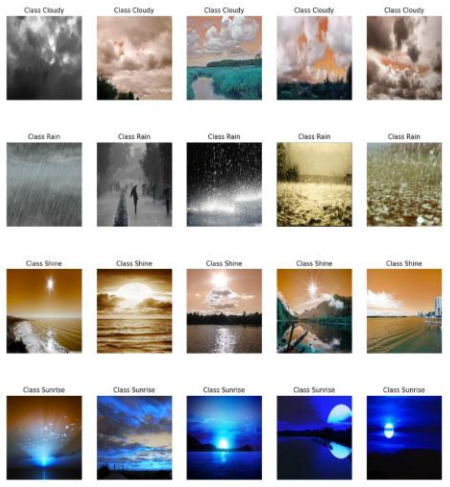
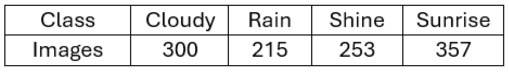


Fig.1 Sample of Images

Since images are of varying sizes, they are all first resized to the size 224 x 224, which is the standard size for processing any image. The reason for resizing images to a fixed size is to keep the size consistent for the convolutional neural network model. After that, all the images and their respective labels are converted to NumPy arrays for easy numerical and matrix computations. Lastly, the following steps of Normalization and Data Augmentation are also performed over the whole image. In Normalization, image pixels are rescaled from the range of [0,225] to [0,1] in order to improve the model performance and achieve faster convergence. In Data Augmentation, a new set of images is generated from the existing dataset just by performing rotation, flipping, and zooming. Lastly, the whole dataset is divided into a split of 80-20, with 80% used for training and 20% for model testing and evaluation.



Class	Cloudy	Rain	Shine	Sunrise
Images	300	215	253	357

Fig.2 Classes of Dataset

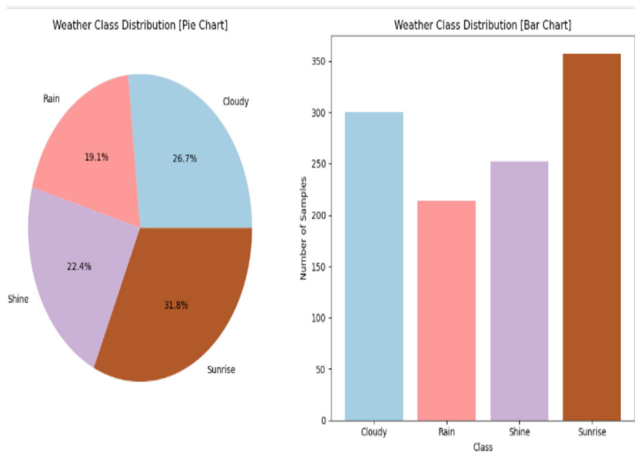


Fig.3 Data Distribution

B. *Network Architecture*

The neural network model architecture used for this study is called the convolutional neural network (CNN), which has different layers of convolution to extract meaningful data from images. In CNN, different layers serve different purposes; convolutional layers are used for feature extraction, pooling layers are used for down-sampling, and denser layers are used for the final prediction. In order to extract the features, there are some specific types of kernels called Convolutional Filters that are used to stride over the whole image to extract local patterns. In this study, first, we are building a custom CNN with different layers and functions, and then we are also using three pre-trained CNN models, VGG16, ResNet50, and MobileNetv2, on the same dataset in order to compare the performance results for all the models [7] [10].

1) Custom CNN Model: This is the Weather-Classification CNN model built using three convolutional layers, three max-pooling layers and two dense layers. This architecture of the CNN model first takes in the input, which is an image of size [224 x 224 x 3], and then passes it through the first convolution layer having 64 filters of size [3x3] followed by ReLU (Rectified Liner Unit) as their activation function. Then, the output of this layer is passed through the max-pooling layer of size [2x2], followed by a dropout of 0.25. Then, the output after dropout is passed through the second convolutional layer consisting of 64 convolutional filters [3x3] and ReLU activation at their end. After that, it is passed through the second max-pool layer of size [2x2], followed by the dropout of 0.25. Lastly, the output after the second dropout is passed through the third convolutional layer having 128 filters of size [3x3], followed by a max pooling layer [2x2] and dropout of 0.25. After all the steps, the final output is flattened and given as input to the first dense layer having 128 neurons. Then, the output is dropped by 0.1, and the remaining is passed as input to the final dense layer consisting of a number of neurons equal to that of a number of classes of the weather dataset, which is 5. This final dense layer output is given to the softmax function, which ultimately gives the output in the form of probability to classify that instance into one of the four classes: Cloudy, Rain, Shine, and Sunrise. In this CNN model, there are a total of 11,188,804 total trainable parameters. The network architecture diagram is presented in Fig.4, and the layer-wise parameter distribution, along with the output shape after each layer, is represented in Fig. 5.

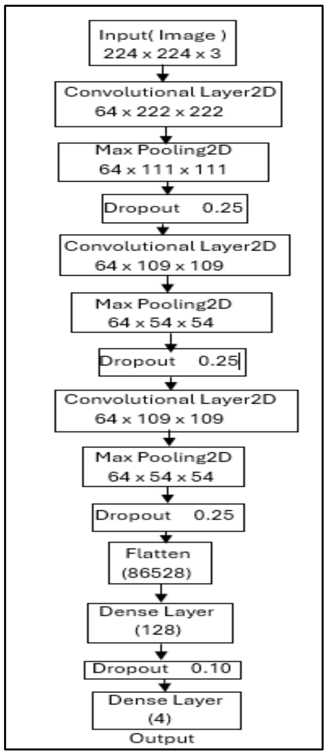
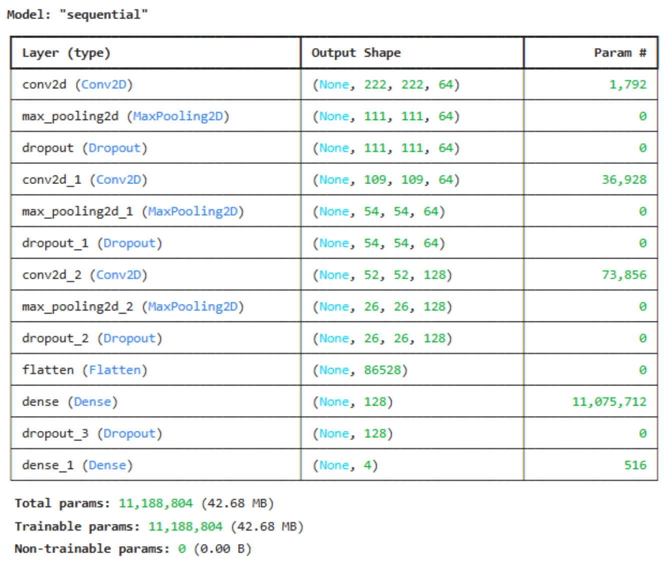


Fig.4 Overview of the Architecture Used



```
Model: "sequential"

Layer (type)                     Output Shape            Param #
conv2d (Conv2D)                  (None, 222, 222, 64)      1,792
max_pooling2d (MaxPooling2D)     (None, 111, 111, 64)          0
dropout (Dropout)                (None, 111, 111, 64)          0
conv2d_1 (Conv2D)                (None, 109, 109, 64)     36,928
max_pooling2d_1 (MaxPooling2D)   (None, 54, 54, 64)            0
dropout_1 (Dropout)              (None, 54, 54, 64)            0
conv2d_2 (Conv2D)                (None, 52, 52, 128)      73,856
max_pooling2d_2 (MaxPooling2D)   (None, 26, 26, 128)           0
dropout_2 (Dropout)              (None, 26, 26, 128)           0
flatten (Flatten)                (None, 86528)                 0
dense (Dense)                    (None, 128)          11,075,712
dropout_3 (Dropout)              (None, 128)                   0
dense_1 (Dense)                  (None, 4)                   516

Total params: 11,188,804 (42.68 MB)
Trainable params: 11,188,804 (42.68 MB)
Non-trainable params: 0 (0.00 B)
```

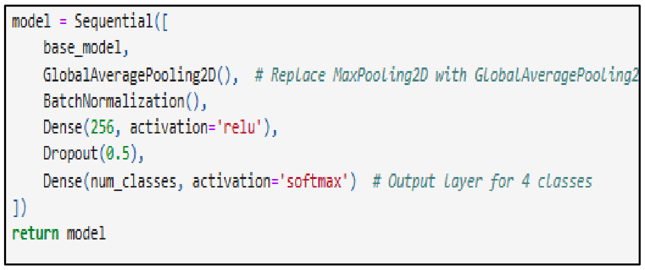
Fig.5 Trainable Parameters at Each Layer

2) VGG16 Model: This is the first pre-trained model used in this study. This model was trained using the ImageNet dataset, which consists of 14 million images and 1000 classes. There, it achieved an accuracy of 92.77% [12][10]. The architecture of this CNN model begins with input images of size 224x224x3 that enter the 2 Convolutional layers, 1 Max-Pooling layer, followed by 2 Convolutional layers and one more Max-Pooling layer [10]. After that, the architecture is provided with 3 Convolutional layers, 1 Max-Pooling layer, 3 Convolutional layers, one more Max-Pooling layer, 3 Convolutional layers, and once again a Max-Pooling layer [10]. Finally, it is followed by three fully connected Dense layers. Also, the number of filters keeps on changing in each layer and 1 SoftMax layer. Here, the filter size for each Convolutional layer is 3x3 and a stride of 1, whereas the Max-Pooling layer has a filter size of 2x2 and a stride of 2 [10] [12] [13] [15].

3) ResNet50 Model: This is the second pre-trained model used in this study. This CNN model consists of 48 Convolutional layers, 1 Max-Pooling layer, and 1 Average Pooling layer [10]. In deep learning neural networks, the network with multiple stacked layers has higher error rates than networks with fewer layers. To resolve this problem, this ResNet50 CNN model introduces the concept of skip connections in which the network is used to estimate the residue instead of the identity function, which results in more excellent performance and better training time. The direct connections of this architecture allow us to skip some layers of the model [10] [11] [15].

4) MobileNetV2 Model: This is the final pre-trained model used in our research. From its name, MobileNet, it is evident that this model is designed for mobile and embedded systems [11]. In order to extract the features, they use lightweight depthwise convolutions, which means these models don't need much computational power [11]. This MobileNetV2 CNN model consists of 53 layers, and the input size is considered to be 224 x 224. Due to the inverted residual structure and the usage of few parameters for model building, these MobileNetV2 models can be run on small devices with limited resources [11]. The main structure of the MobileNetV2 is based on the previous version, which is MobileNetV1 [16][11].

Overall, all three pre-trained models are modified using the exact specifications and settings in order to keep all three models at the same level. Later, they are compared and figured out which model has faster convergence and better results compared to others. For this, firstly, the pre-trained models are loaded as base models, keeping their weights frozen from further training. Then, they are followed by the Average Max-Pooling layer and Batch Normalization. After that, their output is passed into the dense layer, followed by dropout and the final dense layer with four neurons for classifying into four different classes: Cloudy, Rain, Shine,



```
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),  # Replace MaxPooling2D with GlobalAveragePooling2
    BatchNormalization(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')  # Output layer for 4 classes
])
return model
```

Fig.6 CNN Layer Structure for VGG16, ResNet50, and MobileNetV2

and Sunrise. The CNN layer structure for all three models is highlighted in Fig. 6.

III. EXPERIMENTS

In this study, we are performing Multi-Class Weather Classification on Still images using different CNN models, which include both custom CNN models with multiple layers and also three popular pre-trained models: VGG16, ResNet50, and MobileNetV2, which are trained on ImageNet. Our primary goal in this study is to compare the performance metrics of all CNN models trained on the weather image dataset and then analyze them based on their different evaluation parameters, and finally observe and conclude with some insights from that.

A. *Experimental Settings*

To perform all the analyses, we use Python programming language, and we execute those CNN models using the deep learning framework of TensorFlow and Keras. Apart from that, some Python libraries are also used, like NumPy for numerical computations, Scikit-learn for model evaluation, and Matplotlib and Seaborn for Data Visualization insights. Using all these tools and frameworks, we built four models. The first CNN model is the custom one made using three convolutional, three Max Pooling, and two fully connected Dense Layers.

Then, the remaining three models are basically pre-trained models (VGG16, ResNet50, and MobileNetV2) that are fine-tuned and used for prediction using the same weather dataset having 1125 images and four classes of Cloudy, Rain, Shine, and Sunrise. Finally, all the models are compared using evaluation metrics like accuracy, validation accuracy, and loss. Additionally, the performance is observed in the Confusion Matrices and the Classification Report, which includes precision, recall, and F-1 scores.

B. *Results*

1) Custom CNN Model: This model is compiled with Adam optimizer, Categorical Cross-Entropy loss function, and accuracy metric. The model used 80% of the whole dataset for training and 20% for testing. In this, we train the model only for 12 epochs because of early stopping. This CNN model achieved a training accuracy of 96.60% and a validation accuracy of 88.44%. The loss and validation loss for this model is 12.06% and 52.59%, respectively. All these results for training and validation are highlighted in Fig.7. Also, the performance is evaluated using the Confusion Matrix Fig.8 and the Classification Report Fig.9. The Classification Report highlights the Precision, Recall, F-1 Score, and Support with respect to every class of the Mutli-Class Weather classification dataset. It also shows average Precision, Recall, and F-1 scores.

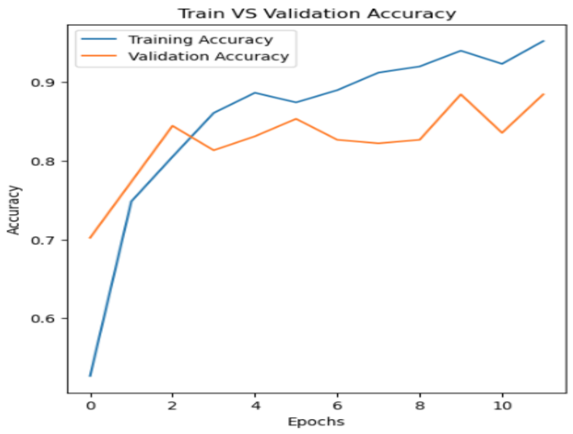


Fig. 7 (a) Training and Validation Accuracy

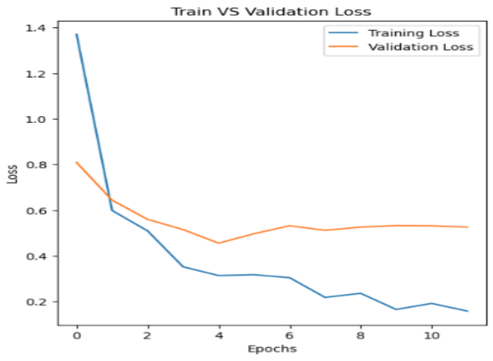


Fig. 7(b) Training and Validation Loss

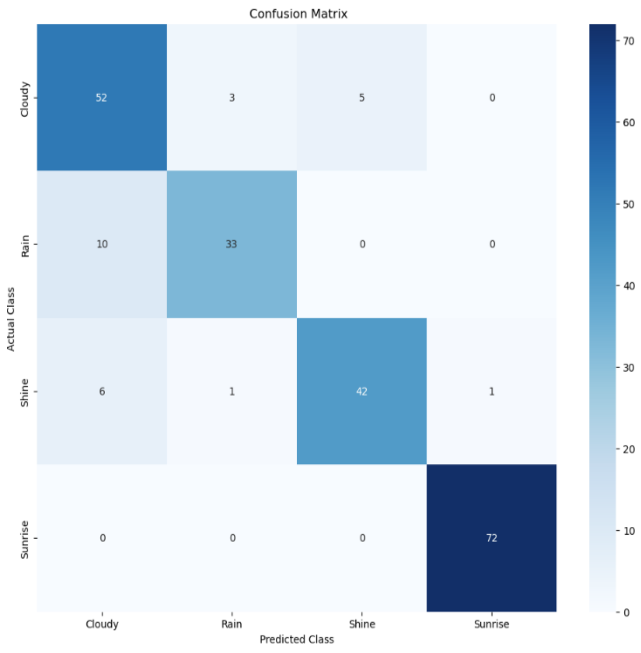
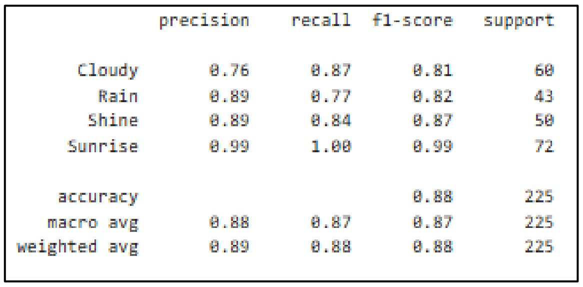


Fig. 8 Confusion Matrix of Custom CNN Model

In Fig.9, there are four evaluation parameters: precision, recall, f1-score, and support with respect to each of the classes of the weather dataset. In the experiment, the custom CNN model performs exceptionally well for class Sunrise, which has a precision of 0.99, a recall of 1.00, and an f1-score of 0.99. Apart from that, the Shine class also has good results compared to the Cloudy and Rain Classes, which show relatively low performance compared to Sunrise and Shine Classes. Some of the random samples and their predictions by the model are highlighted in Fig.10.



```
              precision    recall  f1-score   support

      Cloudy       0.76      0.87      0.81        60
        Rain       0.89      0.77      0.82        43
       Shine       0.89      0.84      0.87        50
     Sunrise       0.99      1.00      0.99        72

    accuracy                           0.88       225
   macro avg       0.88      0.87      0.87       225
weighted avg       0.89      0.88      0.88       225
```

Fig.9 Classification Report Using Custom CNN Model

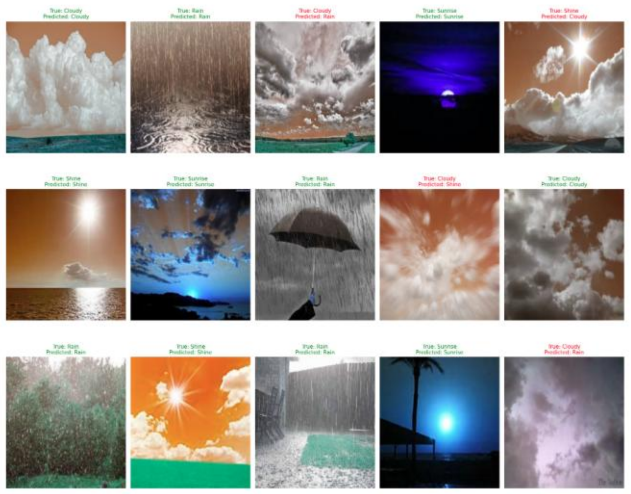


Fig.10 Some Random Predictions by Model

2) *Pre-Trained Models:* In the study, we use three pre-trained models, VGG16, ResNet50, and MobileNetV2, as an approach to transfer learning. These three models are also trained on the same multi-class weather dataset by freezing the base layers of each model and replacing them with dense layers, batch normalization and dropout. Their model architecture is fine-tuned using a Global Average Pooling layer and Dense layers for the final prediction. The model architecture for all three pre-trained models, along with their final fine-tuning, is shown in Fig.6. Firstly, the VGG16 model is trained, and it achieves a training accuracy of 96.85% and a validation accuracy of 95.11%. The loss for the model is 9.73%, and the validation loss is 21.8%. Secondly, the Resnet50 model is trained, with a training accuracy of 91.9% and a validation accuracy of only 58.22%. The loss for the ResNet model is 25.03%, and the validation loss is 88.55%. Lastly, the MobileNetV2 model is trained with an accuracy of 97.52% and a validation accuracy of 92.44%. Also, the loss for that model is 9.02%, and the validation loss is 22.24%. Based on these evaluation parameters, all models are compared in the Fig.11.

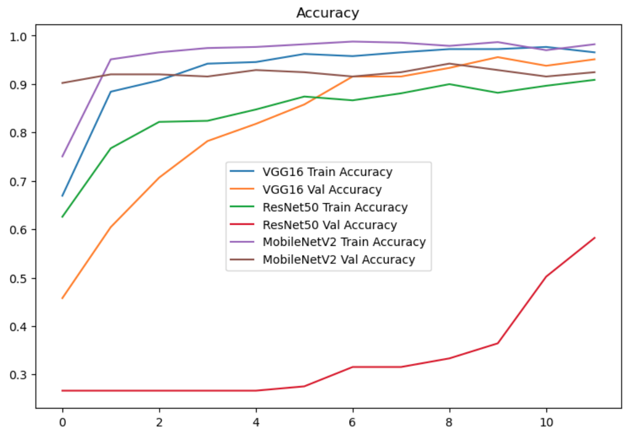


Fig.11(a) Training and Validation Accuracy of All 3 Models

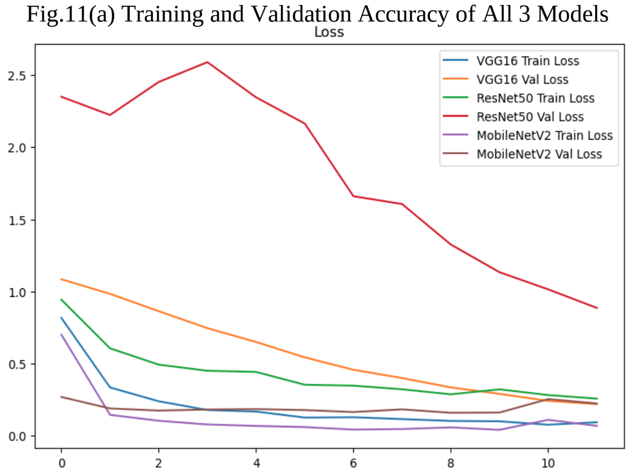


Fig.11(a) Training and Validation Accuracy of All 3 Models

C. *Comparison of Custom CNN with Pre-Trained Models*

Finally, all the performance metrics of the Custom CNN model are compared to that of all 3 Pre-Trained Models. From the results, it is observed that the transfer learning technique worked successfully well on the multi-class weather dataset for the task of image classification. The above-mentioned graphs in Fig.12(a) highlight the accuracy of training and validation for VGG16, ResNet50, and MobileNetV2. From all 3 Pre-Trained models plus the Custom CNN model, VGG16 stays at the top by achieving the maximum validation accuracy of 95.11%, followed by MobileNetV2 with 92.44% and custom CNN by 88.44%, and at last, ResNet50 with a validation accuracy of just 58.22%. ResNet50 model results in an average performance in this study, which shows that the ResNet50 model is not able to capture the generalized pattern of the weather dataset. There are three confusion matrices with respect to 3 pre-trained models in

Fig.12. It is clearly observed from the ResNet50 model's confusion matrix that the majority of the misclassified samples are from class Cloudy, which overall reduces the model's accuracy. Meanwhile, for VGG16 and MobileNetV2, most of the samples are correctly classified, as shown in their respective confusion matrices.

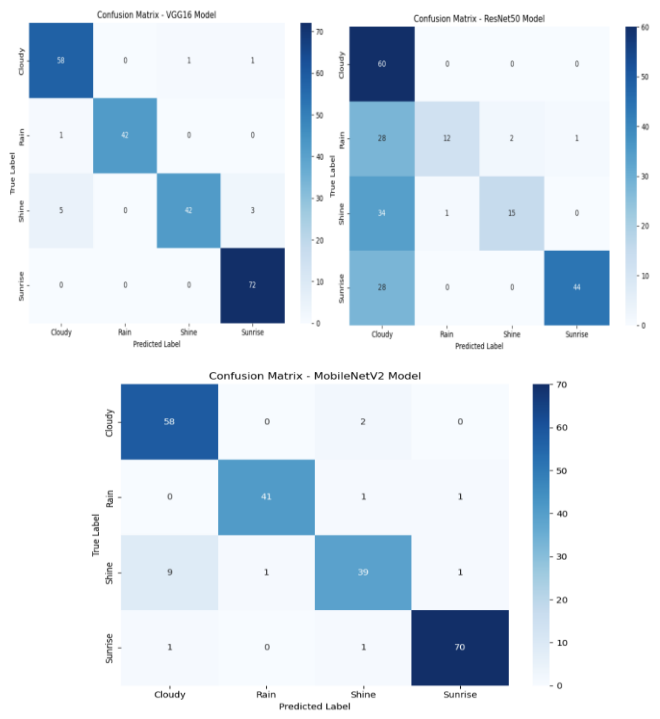
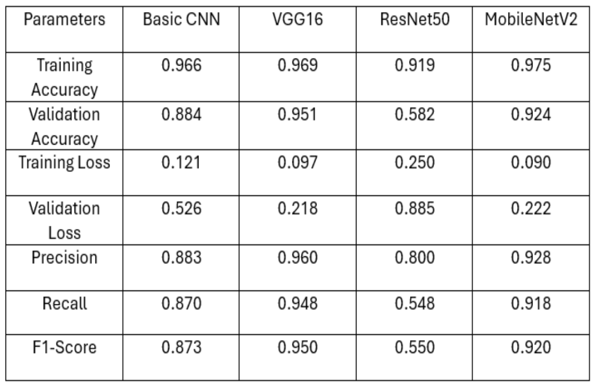


Fig.12 Confusion Matrices for VGG16, ResNet50, MobileNetV2

The table in Fig.13 covers the entire comparison between custom CNN, VGG16, ResNet50, and MobileNetV2, thus covering training accuracy, validation accuracy, training loss, validation loss, precision, recall and f-1 score. From this, it is concluded that VGG16 is considered the best model with the highest validation accuracy and other parameters like precision, recall, and f1-score. That model is followed by MobileNetV2, Basic CNN, and then ResNet50 at last.



Parameters	Basic CNN	VGG16	ResNet50	MobileNetV2
Training Accuracy	0.966	0.969	0.919	0.975
Validation Accuracy	0.884	0.951	0.582	0.924
Training Loss	0.121	0.097	0.250	0.090
Validation Loss	0.526	0.218	0.885	0.222
Precision	0.883	0.960	0.800	0.928
Recall	0.870	0.948	0.548	0.918
F1-Score	0.873	0.950	0.550	0.920

Fig.13 Comparison Table of Different CNN Models

IV. CONCLUSION

The paper highlights the multi-class weather classification task, which is one of the significant things to perform in so many domains before performing other day-to-day activities. To do so, Deep Learning, one of the famous approaches to using Convolutional Neural Networks, is used in this study. For the CNN Models to perform efficiently, a vast amount of data and a high computational device are required to achieve the optimal results. Due to the usage of a small weather dataset in this study, we, along with the custom basic CNN model, also used the approach of transfer learning in which 3 Pre-Trained Models, VGG16, ResNet50, and MobileNetV2, are also considered to be trained and evaluated using weather dataset. From the performance results of all four models, it is concluded that the VGG16 model excelled among all four models with the highest validation accuracy of 95.1%, followed by MobileNetV2, Basic CNN, and ResNet50. However, the difference between their performances is relatively small except for the ResNet50 model, whose performance is relatively poor, with just only 50% validation accuracy. Apart from that, different characteristics of all four models are highlighted and compared using numerous parameters, including Precision, Recall, and F-1 Score. This study clearly reflects the benefits of using the approach of transfer learning, especially for small datasets like weather datasets in which pre-trained models like VGG16, ResNet50, and MobileNetV2 with their already trained weights from ImageNet and some sough of fine-tuning, can efficiently get some promising results.

REFERENCES

[1] A. E. Nagib, H. Mohamed, and A. Ashraf, "Comparative analysis of pre-trained CNN architectures for multi-class weather classification: A study on deep learning techniques," Intelligent Methods, Systems, and Applications (IMSA), pp. 1–8, 2023.

[2] Hiroyuki Kurihata, Tomokazu Takahashi, Ichiro Ide, Yoshito Mekada, Hiroshi Murase, Yukimasa Tamatsu, and Takayuki Miyahara, "Rainy weather recognition from in-vehicle camera images for driver assistance," in Proceedings of IEEE Intelligent Vehicles Symposium, 2005, pp. 205–210.

[3] Martin Roser and Frank Moosmann, "Classification of weather situations on single color images," in Proceedings of IEEE Intelligent Vehicles Symposium, 2008, pp. 798–803.

[4] Xunshi Yan, Yupin Luo, and Xiaoming Zheng, "Weather recognition based on images captured by vision system in vehicle," in Advances in Neural Networks–ISNN 2009, pp. 390–398. 2009.

[5] Cewu Lu, Di Lin, Jiaya Jia, and Chi-Keung Tang, "Twoclass weather classification," in Proceedings of IEEE Computer Vision and Pattern Recognition, 2014, pp. 3718 – 3725.

[6] Z. Zhang and H. Ma, "Multi-class weather classification on single images," in 2015 IEEE International Conference on Image Processing (ICIP). IEEE, 2015, pp. 4396–4400.

[7] Ashish Sharma and Zaid Saad Ismail, "Weather classification model performance: using CNN, Keras-TensorFlow," *ITM Web of Conferences*, vol. 44, pp. 1–10, 2022.

[8] Z. Chen, F. Yang, A. Lindner, G. Barrenetxea, and M. Vetterli, "How is the weather: Automatic inference from images," in 2012 19th IEEE International conference on image processing. IEEE, 2012, pp. 1853–1856.

[9] A. G. Oluwafemi and W. Zenghui, "Multi-class weather classification from still image using said ensemble method," in 2019 Southern African universities power engineering conference/robotics and

mechatronics/pattern recognition association of South Africa (SAUPEC/RobMech/PRASA). IEEE, 2019, pp. 135–140.

[10] Sheldon Mascarenhas and Mukul Agarwal, “A comparison between VGG16, VGG19, and ResNet50 architecture frameworks for image classification,” 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), pp. 1–6, 2021.

[11] Patryk Młodzianowski, “Weather Classification with Transfer Learning - InceptionV3, MobileNetV2, and ResNet50,” *Ignacy Mościcki's University of Applied Sciences in Ciechanów*, pp. 1–6, 2021.

[12] K. Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] M. Elhoseiny, S. Huang, and A. Elgammal, "Weather classification with deep convolutional neural networks," in 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 2015, pp. 3349–3353

[14] Y. Hao and Y. Zhu, "Weather Classification for Multi-class Weather Image Based on CNN," *2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE)*, WuHan, China, pp. 1–8, 2022.

[15] D. Theckedath and R. R. Sedamkar, "Detecting Affect States Using VGG16, ResNet50 and SE-ResNet50 Networks," *SN Computer Science*, vol. 1, no. 2, p. 79, Mar. 2020. doi: 10.1007/s42979-020-0114-9.

[16] Ke Dong, Yihan Ruan, Chengjie Zhou, and Yuzhi Li, “MobileNetV2 Model for Image Classification,” *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pp. 1–6, 2020.