GEORGE BROWN COLLEGE
SCHOOL OF COMPUTER TECHNOLOGY
APPLIED A.I. SOLUTIONS

ADVANCED APPLIED MATHEMATICAL CONCEPTS FOR DEEP LEARNING

MINI-PROJECT

Professor: Dr. Mahdieh Khalilinezhad

Mohammad Ardalanasl

March 18, 2024

1.0 INTRODUCTION

This project focuses on the development and comparison of two Convolutional Neural Network (CNN) models designed to classify an 11-class database. The primary objectives of this endeavor are to gain practical experience in training a CNN model from inception and to explore the utilization of pre-trained models for classification tasks. The project utilizes a sample database sourced from Kaggle and employs a medium-sized architecture for the models. The process entails training a CNN model from scratch followed by the deployment of a pre-trained model. The ensuing sections of this report detail the methodology, results, and comparative analysis of the two models.

2.0 DATA

The Weather Image Recognition database utilized in this project was sourced from Kaggle [1] and comprises 11 classes: dew, fog smog, frost, glaze, hail, lightning, rain, rainbow, rime, sandstorm, and snow. The dataset consists of 6862 images across all classes.

Preprocessing of the data involved developing a model to read the raw data, generate a list of labels and image directories, and partition the data into training, development, and test sets. Subsequently, the model created new directories for each set of data to facilitate the training process.

Figure 1. Few sample images of database

3.0 MODELS

First Model:

The first model comprises 1900843 trainable parameters and 928 non-trainable parameters. Its architecture was selected through iterative steps in the initial stages of model development, incorporating Conv2D, Data Augmentation, Batch Normalization, ReLU Activation, MaxPooling2D, and Skip Connection layers. The model was trained on images resized to 128x128 pixels, with a batch size of 32, and for 30 epochs.

Second Model:

The second model utilizes the VGG19 pre-trained model, with its classification head removed and a Dense layer added with 256 nodes, followed by a new SoftMax layer consisting of 11 classes. A Dropout layer with a value of 0.15 is included to help mitigate overfitting. This model comprises 2100235 trainable parameters and 20024384 non-trainable parameters. It was trained on 128x128 images, with a batch size of 64, and for 30 epochs.

4.0 RESULTS

Following the training process, the models were evaluated using the test set, and the convergence process in the train and dev sets was depicted through graphs illustrating loss and accuracy values.

Results of Model 1:

The classification report of Model 1 indicates an F1-Score of approximately 0.67. The confusion matrix illustrates the model's predictions across different classes, highlighting areas of misclassification.

	precision	recall	f1-score	support
0	0.69	0.90	0.78	70
1	0.93	0.51	0.66	85
2	0.41	0.69	0.52	48
3	0.43	0.41	0.42	64
4	0.85	0.37	0.52	59
5	0.94	0.79	0.86	38
6	0.56	0.89	0.69	53
7	0.60	0.91	0.72	23
8	0.90	0.67	0.77	116
9	0.70	0.87	0.77	69
10	0.57	0.55	0.56	62
accuracy			0.67	687
macro avg	0.69	0.69	0.66	687
weighted avg	0.72	0.67	0.66	687

Figure 2. the classification report of Model 1

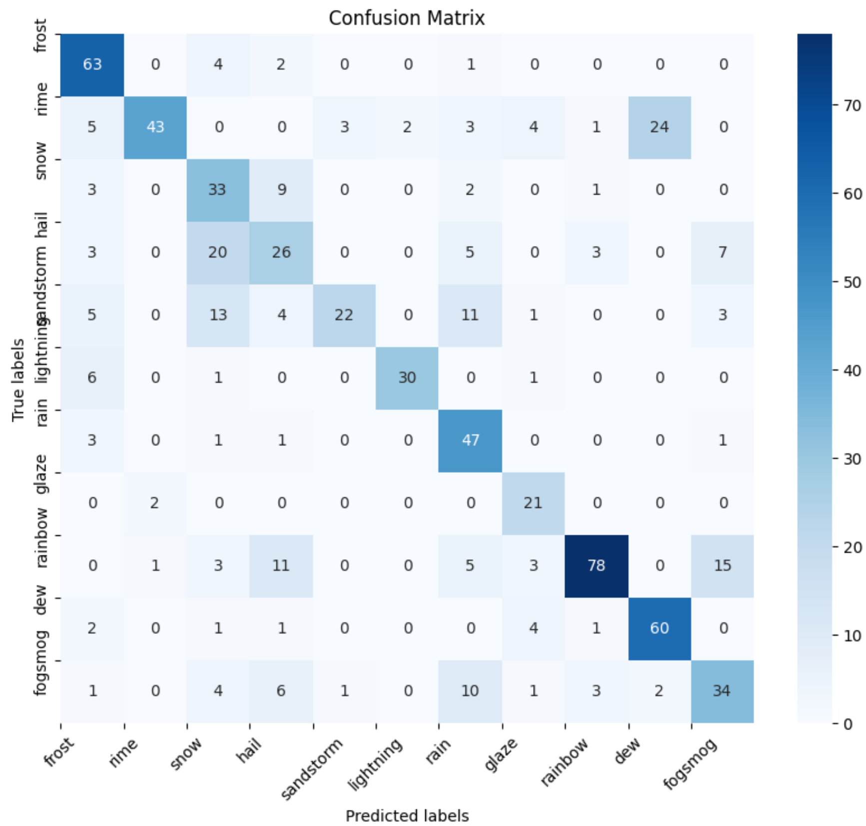

Figure 3. the Confusion matrix of Model 1

The graph depicting the loss values for the train and dev sets in each epoch shows that the loss values for both sets in Model 1 are close to each other, indicating that the model is performing well in terms of overfitting. Similarly, the accuracy graph for the train and dev sets shows that the model's accuracy in these sets is also comparable, indicating reasonable performance.

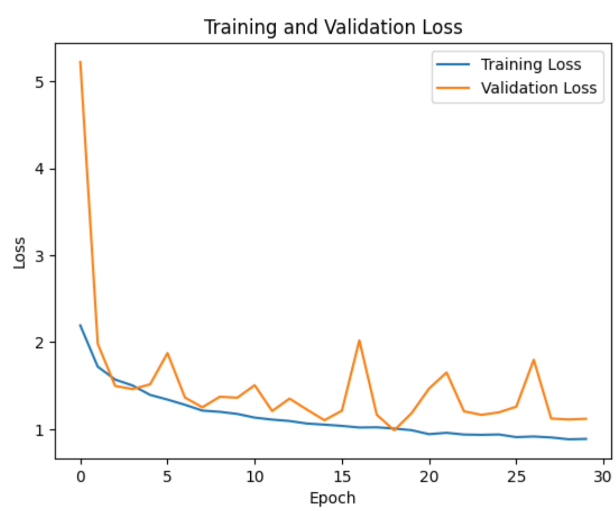

Figure 4. the graph of loss value for train and dev sets in Model 1

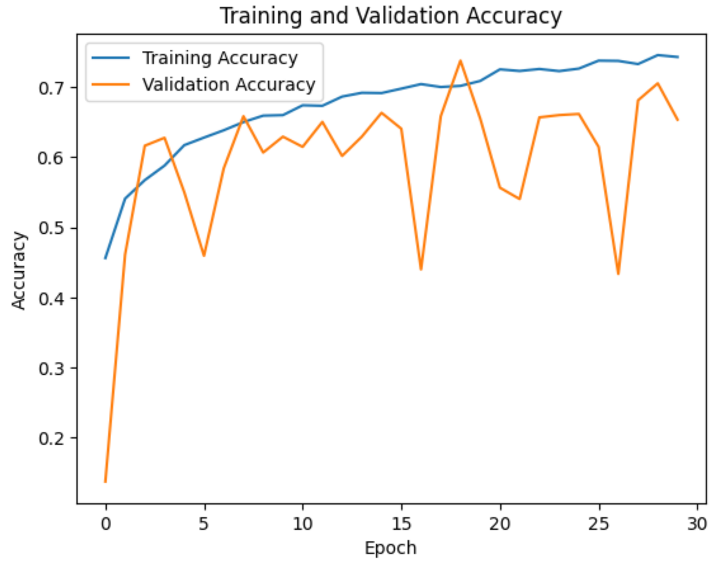

Figure 5. the graph of accuracy value for train and dev sets in Model 1

Results of Model 2:

Model 2 achieved an F1-Score of approximately 0.79, which is more than 15% better than the F1-Score of Model 1. The confusion matrix for Model 2 demonstrates a better classification performance compared to Model 1.

	precision	recall	f1-score	support
0	0.87	0.84	0.86	70
1	0.87	0.73	0.79	85
2	0.59	0.73	0.65	48
3	0.71	0.61	0.66	64
4	0.80	0.90	0.85	59
5	0.88	0.97	0.93	38
6	0.71	0.87	0.78	53
7	1.00	0.87	0.93	23
8	0.83	0.89	0.86	116
9	0.81	0.80	0.80	69
10	0.69	0.55	0.61	62
accuracy			0.79	687
macro avg	0.80	0.80	0.79	687
weighted avg	0.79	0.79	0.79	687

Figure 6. the classification report of Model 2

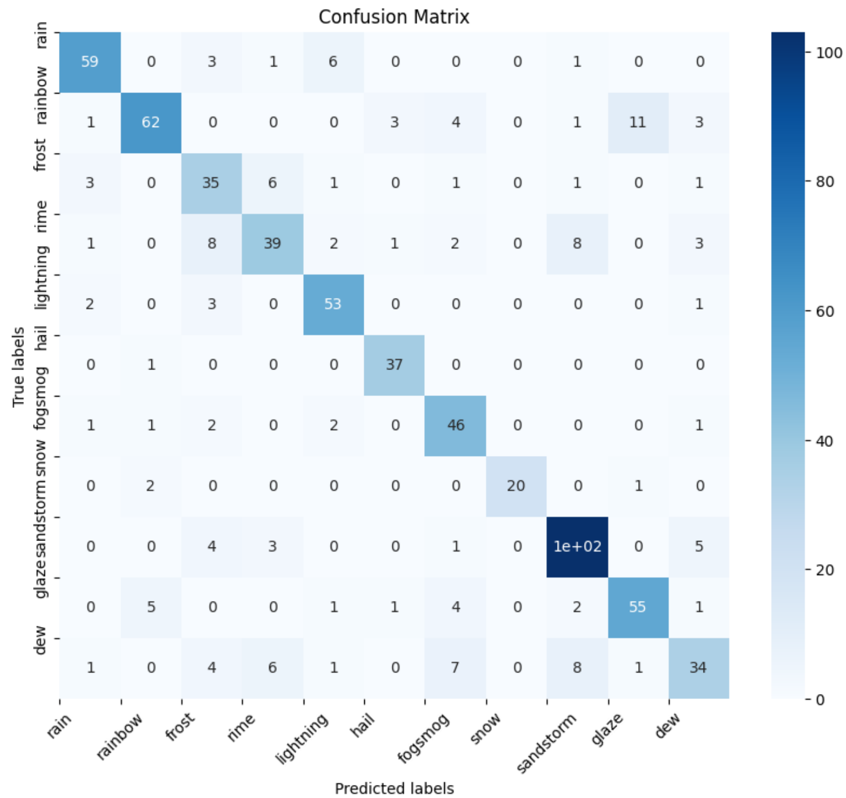

Figure 7. the Confusion matrix of Model 2

However, analysis of the loss value graph for the train and dev sets in Model 2 reveals a noticeable difference between the loss curves of the training and validation sets. This suggests that while techniques such as Data Augmentation and Dropout were employed to control overfitting, additional data may be necessary to further mitigate this issue.

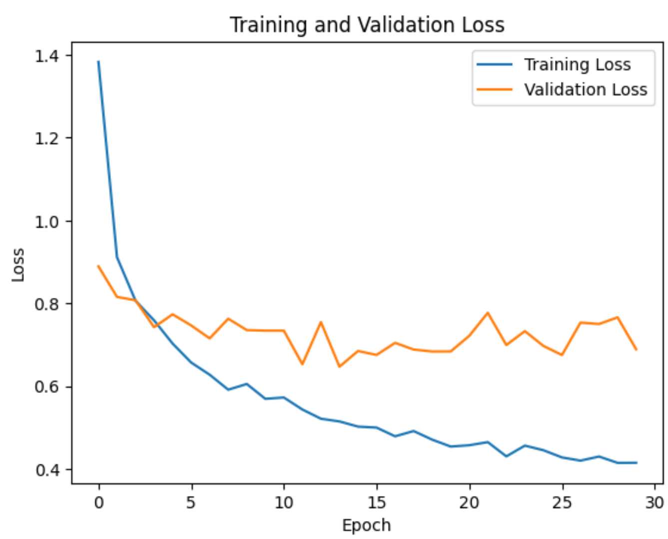

Figure 8. the graph of loss value for train and dev sets in Model 2

Similarly, the accuracy curves for the training and dev sets in Model 2 show a gap, with the accuracy on the training set reaching approximately 0.85, while the accuracy on the dev set plateaus at 0.79. This implies that increasing the size of the dataset could potentially enhance the accuracy of Model 2.

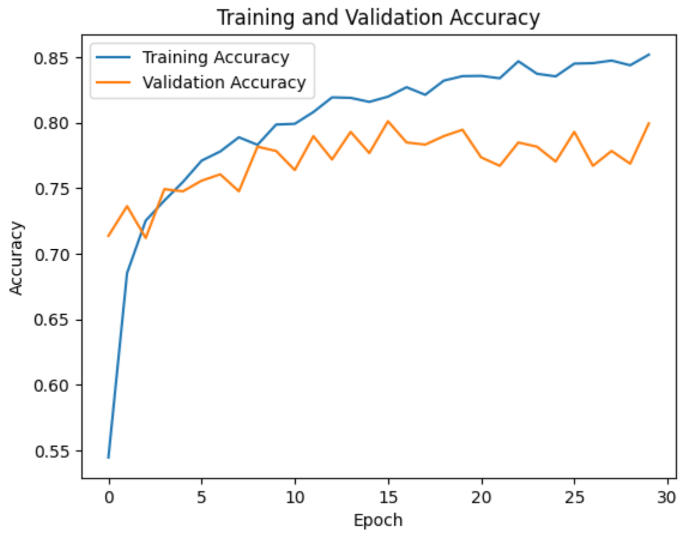

Figure 9. the graph of accuracy value for train and dev sets in Model 2

5.0 CONCLUSION

In conclusion, this project successfully developed and compared two CNN models for classifying an 11-class database, with one model trained from scratch and the other utilizing a pre-trained VGG19 model. While the first model exhibited reasonable performance, the second model, leveraging transfer learning, achieved significantly higher accuracy. However, both models displayed signs of overfitting, suggesting that further improvements could be achieved through the use of additional data and more advanced regularization techniques. Overall, this project provided valuable insights into the process of training CNN models and highlighted the efficacy of transfer learning in improving classification performance.

6.0 REFERENCES

[1] Jehan Bhathena, Kaggle, https://www.kaggle.com/datasets/jehanbhathena/weather-dataset