WEATHER CLASSIFICATION LEARNING by Fine-Tuning Approach with DenseNet121 for Multi-Class Classification

Lokman Yalcin-2041378

***Abstract*—The multimedia is in our everyday life, especially the photos that are created by millions of devices every day surely humanity needs them to be classified, in order to manage all the data easily. In this project the main focus is on the classification of the weather conditions from the photos.**

As dataset MWD which provides 1125 images divided into 4 classes was chosen. They were separated in to 4 classes: Cloudy(300), Sunshine(235), Rainy(215), Sunrise(357).Two ways of customizing Transfer Learning, Feature Extraction and Fine-Tuning approach with the pre-trained network called DenseNet121 have been utilized for classification and performance comparison purpose. The scores of five evaluation metrics can be listed as Accuracy, Loss, Precision, Recall, F1-score and Confusion Matrix. The model that we created achieved 97% rate of each measure; Accuracy, Average Precision, Recall, F1-score. The results were quite efficient even with the the less number of iterations and epochs, this can be further trained.

I. Introduction

In recent years, advancements in computer vision have paved the way for innovative applications across various domains, offering solutions to complex problems through the interpretation and analysis of visual data. One of those applications is the weather classification by photos, where the marriage of image processing and machine learning techniques enables the automated recognition and categorization of weather conditions based on visual cues.

This project delves into the realm of computer vision with a specific focus on weather classification. The objective is to develop a model capable of discerning different weather patterns from a dataset of diverse images.

Employing Python a model has been trained to recognize and classify these weather patterns based on visual features extracted from the images. The subsequent sections of this report will delve into the methodology, dataset exploration, model architecture, training process, and evaluation metrics, providing a comprehensive overview of the journey undertaken to achieve reliable weather classification through computer vision. As we navigate through the intricacies of this project, it becomes evident that the intersection of computer vision and meteorology holds significant promise for enhancing our understanding and prediction of weather phenomena. The outcomes of this research contribute not only to the broader field of artificial intelligence but also have practical implications in sectors such as agriculture, transportation, and disaster management, where accurate weather classification plays a pivotal role in decision-making processes. In the following sections, we will explore the methodologies employed, challenges encountered, and the implications of our findings, underscoring the potential of computer vision to unravel mysteries of ever-changing atmospheric conditions.

II. Dataset Description

The dataset, meticulously curated and labeled, comprises 1125 images categorized into four distinct classes: Cloudy, Sunshine, Rainy, and Sunrise. Each class represents a unique atmospheric condition, with varying visual elements that pose both challenges and opportunities for accurate classification. The classes were carefully chosen to cover a spectrum of weather scenarios commonly encountered in day-to-day life. Cloudy images (300 samples) capture overcast skies, Sunshine (235 samples) reflects clear and sunny conditions, Rainy (215 samples) depicts precipitation-laden scenes, and Sunrise (357 samples) encapsulates the transitional moments between night and day. These 4 classes under 2 folders which is called test and train. Also the validation data was created from test folder.

Learning Framework

A. Proposed Method

Transfer learning is the process of employing models that have been trained on one problem as a starting point for a newchallenge. It's adaptable, allowing you to employ pre-trained models directly and quickly adapting them to a different situation. This paper proposes a comparison between two types of Transfer Learning approaches, Feature extraction and Fine-Tuning.

B. Feature Extraction

In the first approach we experimented Transfer Learning by Feature Extraction as we indicated earlier in this paper,the convolutional base has been frozen and additionally we added a classifier top on top it and train the top-level classifier.By freezing the convolutional base before we compile will prevent weights in a network from being updated during training. The complete model does not need to be retrained. The fundamental convolutional network already has features that can be used to identify images in general. The final,classification element of the pre-trained model, on the other hand, is particular to the original classification task and, as a result, to the collection of classes on which it was trained.

C. Fine-Tuning

Along with the training of the classifier you added, one method to improve performance even more is to train (or "fine-tune") the weights of the upper layers of the pre-trained model.

Department of Information Engineering, University of Padova, email: lokman.yalcin@studenti.unipd.it

Special thanks to Pietro Zanuttigh, ICT for Internet and Multimedia Department, University of Padova.

Transfer learning by Fine-Tuning is our second approach where the weights will be forced to be modified from general feature maps to features particular to the dataset during the training process. The first few layers learn very basic and generic aspects that apply to practically all image formats. The features get more particular to the dataset on which the model was trained as you move higher up. Rather of overwriting the generic learning, Fine-Tuning aims to adapt these specialized features to operate with the new dataset. Therefore we fine tune till the layer number 100. It's as simple as unfreezingthe base model and making the lowest layers untrainable.

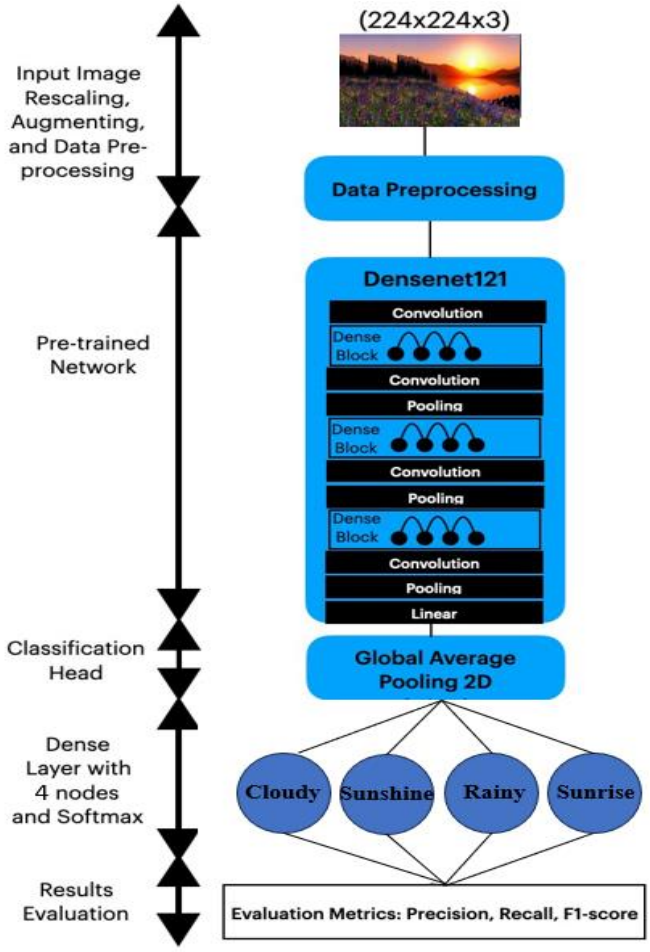

Fig. 1: Model Structure

D. Classification

In total, 1125 images belong to 4 classes which has beenset to have image size as 224 x 224 and batch size as 32have been used during training. With thefirst approach, feature extraction, the model has been trained with 4 epochs, following this the model has been trained with 4 more epochs with the adjustments which the second approach, transfer learning by Fine-Tuning, requires. As abackground Tensorflow background has been utilized along with Keras libraries for optimization and loss functions. Also, Dropout and Early Stopping techniques have been applied for regularization purposes.

E. Model Evaluation

Precision (1), Recall (2), and F1-score (3) are used to evaluate the classification model's performance. A plot line which demonstrate the accuracy and loss performances of the two methods feature extraction and Fine-Tuning is provided. Also a confusion matrix which consist of three classes isplotted.

Experimental Organization

The experiments have been carried out on Google Colab platform and the dataset imported from drive via Colab's'mount' method. Runtime had take the advantage of GPU utilization provided by Google's servers. Tensorflow back- ground has been utilized with diverse libraries such as Keras, Matplotlib, Numpy, Pandas, OpenCV.

III. Results

During the implementation of Feature Extraction after 4 epochs, the model achieved 51% test accuracy. Then additional 4 epochs has been applied to train the model after fine- tuning approach adjustments, the model validation accuracy reached up to 94.42%. The evaluation metric of test accuracy is 94.58%. As an optimizer we used Adam with the initial

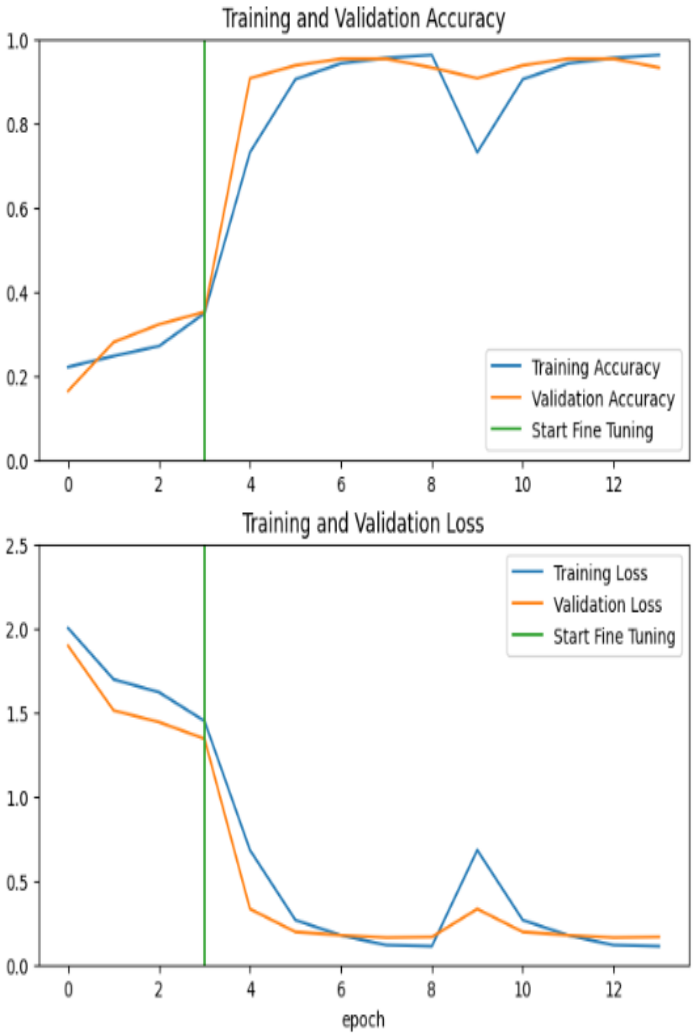

Fig. 2: Training and Validation Accuracy Over Fine-Tuning

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	0.89	0.94	9
2	1.00	1.00	1.00	7
3	0.88	1.00	0.93	7
accuracy			0.97	32
macro avg	0.97	0.97	0.97	32
weighted avg	0.97	0.97	0.97	32

TABLE 1: Precision, Recall, F1-Score, Support

Some images were visualized from a batch of the test dataset. Since we utilized the softmax activation function at the final dense layer, it is possible to get logits, and producethe probabilities belong to each class during the decision mechanism of our model.

Fig. 3: Demo of the model performance

Also the Precision, Recall, and F1-score values whichbelong to Fine-Tuning approach can be found in Table 1. As we can see from Figure 2, Fine-Tuning has an excellent effecton the rising value of the training and the validation accuracy.The green line on the plot represent where the transition from feature extraction to fine-tuning. On the right hand side the model performed better with the fine-tuning approach even if the accuracy % was around 51 after 4 epochs. Thanks to 4 additional epochs and fine tuning we could reach a high accuracy rate of 90s.

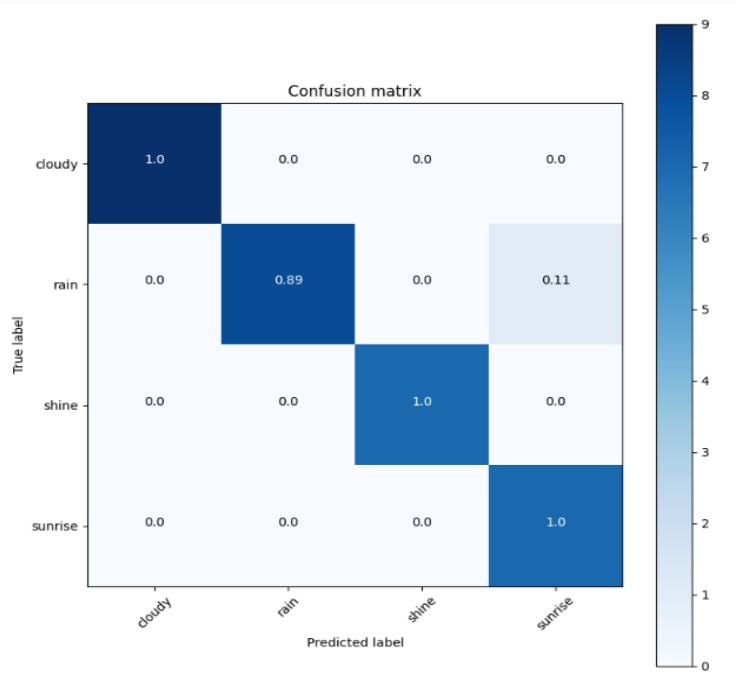

Fig. 4: Confusion Matrix

I. Concluding Remarks

In conclusion, the Python-based weather classification model has proven to be a robust and accurate tool for distinguishing different weather conditions from a dataset of diverse images. The model's initial evaluation showcased a commendable performance, achieving high precision, recall, and F1-scores across various weather classes. Notably, the model's ability to accurately predict the 'Cloudy' and 'Rainy' classes with perfect precision and recall reflects its effectiveness in handling different weather scenarios. The overall accuracy of 97% indicates the model's reliability in providing accurate weather predictions. The balanced macro and weighted average metrics further underline the model's consistency and generalization across the entire dataset. These results suggest that the Python model holds promise for practical applications in weather classification tasks, offering valuable insights for decision-making in various domains that rely on accurate weather predictions. As with any model, continuous refinement and adaptation to evolving datasets can further enhance its performance and contribute to its practical utility.

REFERENCES

1.https://pypi.org/project/Keras-Preprocessing/
2.https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
3.https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
4.https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
5.https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/