Advanced Programming Final report

IT-2107 Abylai Nurske
https://youtu.be/m-A9aaF1NUo
https://github.com/AbylaiNur/adv-prog-final

Introduction

Problem

Classifying weather from images is the task of identifying weather conditions from pictures. Building a web application for uploading and classifying images.

Literature review

https://www.kaggle.com/code/hamzamanssor/weather-recognition-using-deep-learning-models

The solution used six pre-trained models - EfficientNetB7, ResNet, MobileNet, VGG19, Xception, InceptionResNetV2, and VGG16 - for image classification on a large dataset. EfficientNetB7 performed the best in accuracy, while MobileNet was the fastest. Overall most of the models on average had accuracy 79%
https://urvog.medium.com/weather-image-classification-with-keras-4eee9468ff2f

The same direction from the previous solution. Used pre-trained models.

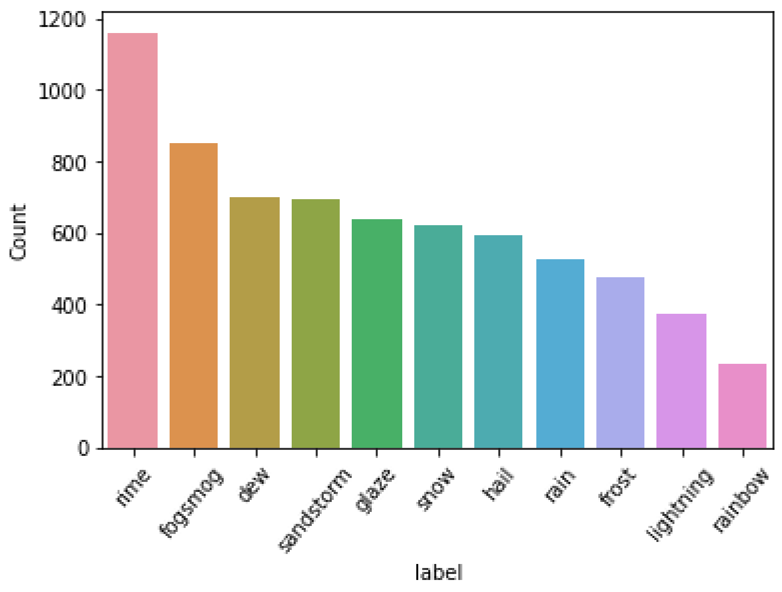
Current work

The work used Convolutional Neural Networks (CNNs) and transfer learning techniques from four pre-trained models (Xception, MobileNetV2, InceptionV3, and DenseNet121) for image classification. Transfer learning involves using pre-trained models to extract features from images and fine-tuning them on a smaller dataset. The goal of the work was to compare the performance of these pre-trained models on the classification of a dataset.

Dataset:
https://www.kaggle.com/datasets/fceb22ab5e1d5288200c0f3016ccd626276983ca1fe8705ae2c32f7064d719de

Data and Methods

Information about the data



There are 11 classes of images.
Rime, fogsmog, dew, sandstorm, glaze, snow, hail, rain, frost, lightning, rainbow.

Description of ML models

Convolutional Neural Networks (CNNs) are a type of deep learning model that is commonly used for image classification tasks. The key idea behind CNNs is that they can learn to extract hierarchical representations of features from images. The CNN architecture consists of several layers, including convolutional layers, pooling layers, and fully connected layers.

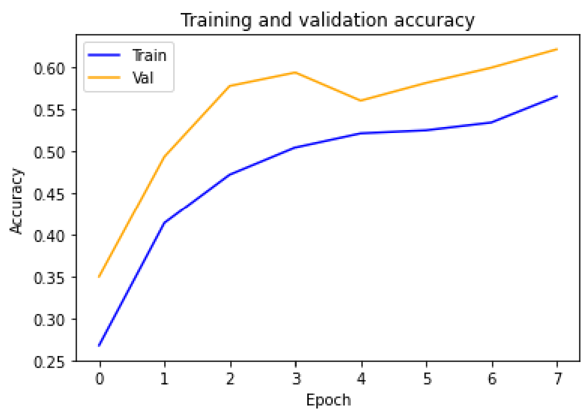
In a convolutional layer, the model applies a set of filters to the input image, which convolve over the input and produce a set of activation maps that represent the learned features. The filters are learned during the training process and are optimized to identify specific patterns in the input images. The pooling layers are used to downsample the activation maps and reduce their dimensionality, while preserving the most important features.

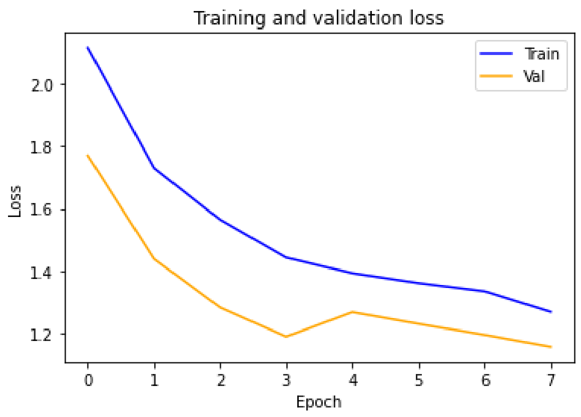
The fully connected layers are used to map the extracted features to the output classes. The output of the final fully connected layer is passed through a softmax function to obtain a probability distribution over the classes.

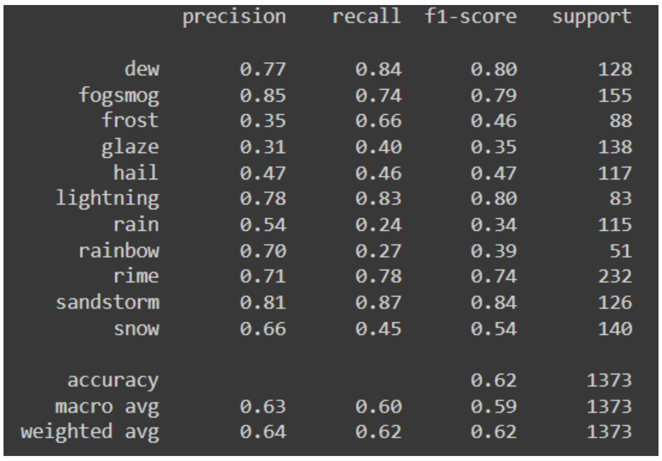
Results

Simple CNN

It is a two block convolutional neural network

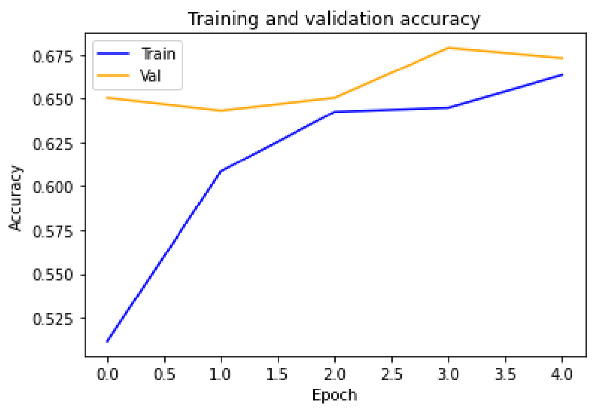


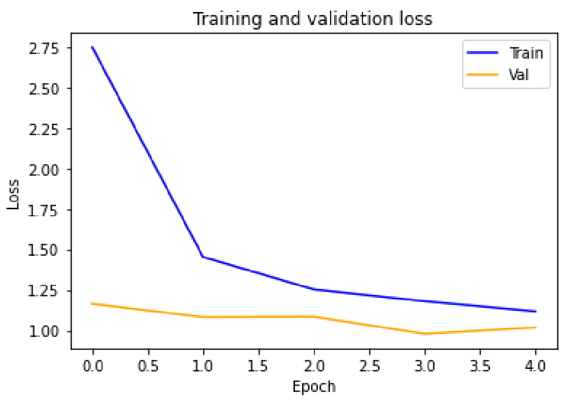


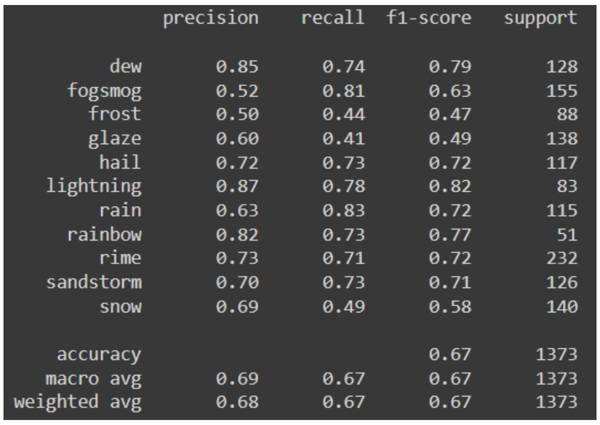


```
              precision    recall  f1-score   support

         dew       0.77      0.84      0.80       128
     fogsmog       0.85      0.74      0.79       155
       frost       0.35      0.66      0.46        88
       glaze       0.31      0.40      0.35       138
        hail       0.47      0.46      0.47       117
   lightning       0.78      0.83      0.80        83
        rain       0.54      0.24      0.34       115
     rainbow       0.70      0.27      0.39        51
        rime       0.71      0.78      0.74       232
   sandstorm       0.81      0.87      0.84       126
        snow       0.66      0.45      0.54       140

    accuracy                           0.62      1373
   macro avg       0.63      0.60      0.59      1373
weighted avg       0.64      0.62      0.62      1373
```

Xception

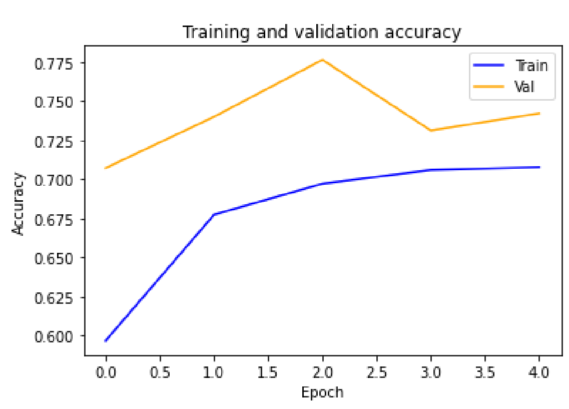


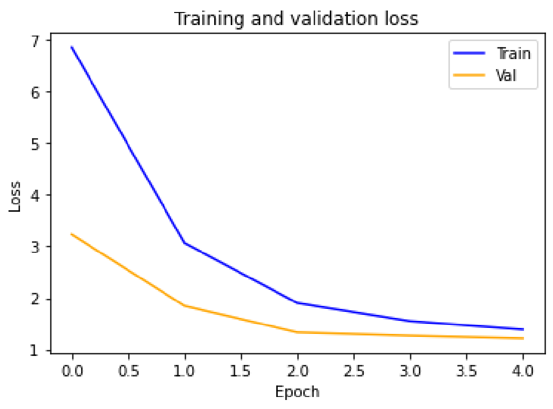


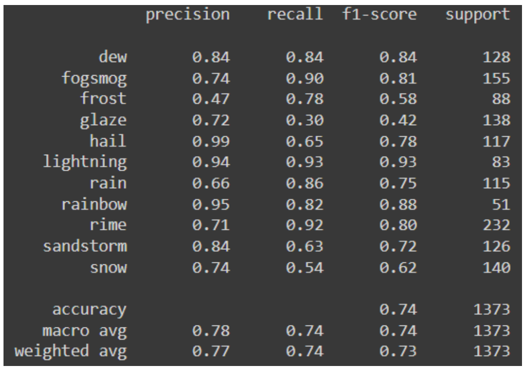


```
              precision    recall  f1-score   support

         dew       0.85      0.74      0.79       128
     fogsmog       0.52      0.81      0.63       155
       frost       0.50      0.44      0.47        88
       glaze       0.60      0.41      0.49       138
        hail       0.72      0.73      0.72       117
   lightning       0.87      0.78      0.82        83
        rain       0.63      0.83      0.72       115
     rainbow       0.82      0.73      0.77        51
        rime       0.73      0.71      0.72       232
   sandstorm       0.70      0.73      0.71       126
        snow       0.69      0.49      0.58       140

    accuracy                           0.67      1373
   macro avg       0.69      0.67      0.67      1373
weighted avg       0.68      0.67      0.67      1373
```

MobileNetV2

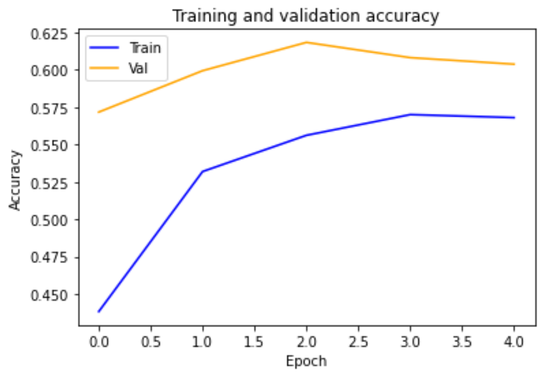


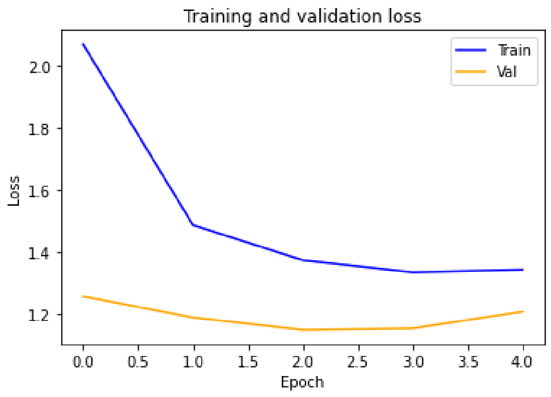


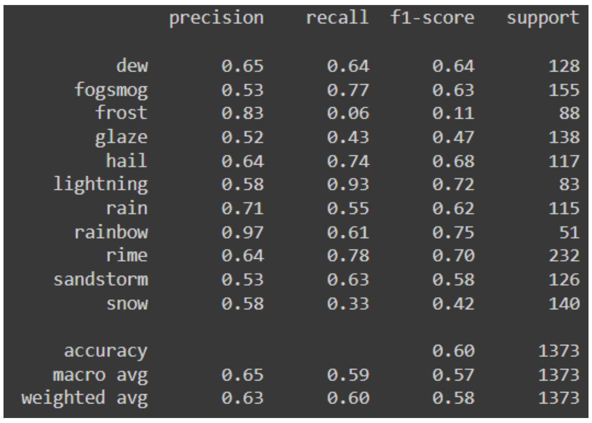


```
              precision    recall  f1-score   support

         dew       0.84      0.84      0.84       128
     fogsmog       0.74      0.90      0.81       155
       frost       0.47      0.78      0.58        88
       glaze       0.72      0.30      0.42       138
        hail       0.99      0.65      0.78       117
   lightning       0.94      0.93      0.93        83
        rain       0.66      0.86      0.75       115
     rainbow       0.95      0.82      0.88        51
        rime       0.71      0.92      0.80       232
   sandstorm       0.84      0.63      0.72       126
        snow       0.74      0.54      0.62       140

    accuracy                           0.74      1373
   macro avg       0.78      0.74      0.74      1373
weighted avg       0.77      0.74      0.73      1373
```

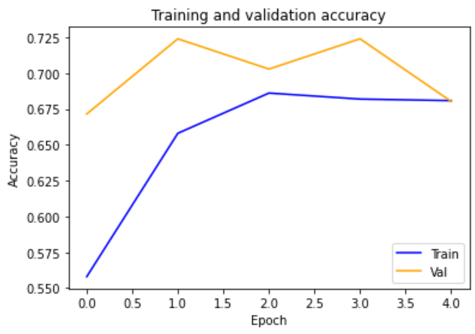
InceptionV3

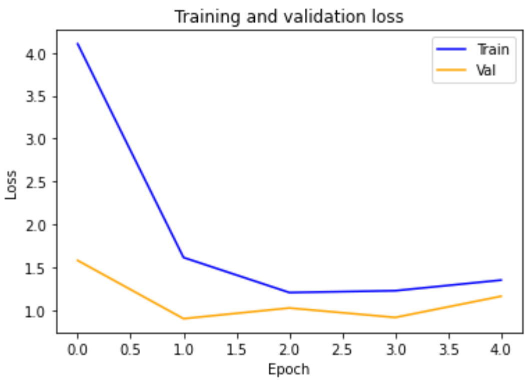


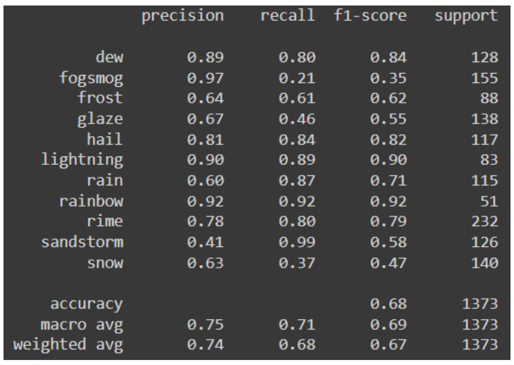



```
              precision    recall  f1-score   support

         dew       0.65      0.64      0.64       128
     fogsmog       0.53      0.77      0.63       155
       frost       0.83      0.06      0.11        88
       glaze       0.52      0.43      0.47       138
        hail       0.64      0.74      0.68       117
   lightning       0.58      0.93      0.72        83
        rain       0.71      0.55      0.62       115
     rainbow       0.97      0.61      0.75        51
        rime       0.64      0.78      0.70       232
   sandstorm       0.53      0.63      0.58       126
        snow       0.58      0.33      0.42       140

    accuracy                           0.60      1373
   macro avg       0.65      0.59      0.57      1373
weighted avg       0.63      0.60      0.58      1373
```

DenseNet121







```
              precision    recall  f1-score   support

         dew       0.89      0.80      0.84       128
    fogsmog       0.97      0.21      0.35       155
       frost       0.64      0.61      0.62        88
       glaze       0.67      0.46      0.55       138
        hail       0.81      0.84      0.82       117
   lightning       0.90      0.89      0.90        83
        rain       0.60      0.87      0.71       115
     rainbow       0.92      0.92      0.92        51
        rime       0.78      0.80      0.79       232
   sandstorm       0.41      0.99      0.58       126
        snow       0.63      0.37      0.47       140

    accuracy                           0.68      1373
   macro avg       0.75      0.71      0.69      1373
weighted avg       0.74      0.68      0.67      1373
```

Simple CNN - It is a two block convolutional neural network

Discussion

Based on the given results, we can see that MobileNetV2 outperformed the other models with an accuracy of 74%. The precision, recall, and F1-scores for most of the classes are also relatively high, indicating that the model is performing well overall. The second-best model was Xception with an accuracy of 67%.

The Simple CNN model had the lowest accuracy of 62%. While it did perform reasonably well on some classes (e.g., dew, lightning), it struggled with others (e.g., frost, glaze, and rain), as indicated by low precision, recall, and F1-scores, because of its simplicity and low number of layers in architecture.

InceptionV3 performed better than Simple CNN model but worse than Xception and MobileNetV2, with an accuracy of 60%. Its performance was especially poor for frost, rainbow, and snow, as indicated by very low precision, recall, and F1-scores. Probably it could have passed Simple CNN if it had more epochs.

In conclusion, we can see that MobileNetV2 is the best model among the ones tested, followed by Xception.

Sources:

https://www.kaggle.com/code/hamzamanssor/weather-recognition-using-deep-learning-models#%E2%9C%94%EF%B8%8F-EfficientNetB7

From this solution I used code for loading data

```
path = '../input/weather-dataset/dataset'
path_imgs = list(glob.glob(path+'/**/*.jpg'))
```

```
labels = list(map(lambda x:os.path.split(os.path.split(x)[0])[1], path_imgs))
file_path = pd.Series(path_imgs, name='File_Path').astype(str)
labels = pd.Series(labels, name='Labels')
data = pd.concat([file_path, labels], axis=1)
data = data.sample(frac=1).reset_index(drop=True)
data.head()
```

visualization of the number of images per label

```
counts = data.Labels.value_counts()
sns.barplot(x=counts.index, y=counts)
plt.xlabel('Labels')
plt.ylabel('Count')
plt.xticks(rotation=50);
```

data test

```
    # Predict Data Test
    pred = model.predict(test_gen )
    pred = np.argmax(pred,axis=1)
    labels = (train_gen.class_indices)
    labels = dict((v,k) for k,v in labels.items())
    pred = [labels[k] for k in pred]

    # Classification report
    cm=confusion_matrix(test_df.Labels,pred)
    clr = classification_report(test_df.Labels, pred)
    print(clr)
```

https://towardsdatascience.com/loading-models-into-tensorflow-js-using-react-js-f3e118ee4a59

I learned how to load a model in React.js, but I encountered an issue where the model wasn't loading in my local React application. I searched for a solution on Stack Overflow and found one that allowed me to load the model locally.

https://www.youtube.com/watch?v=S_Lg1bVbqY4

Learned how to work and predict with models in react.js.