Weather Image Classification using Convolutional Neural Networks

MSA8600 Deep Learning Analytics | Final Project Report

Hwan Oh | Sunwoo Lee | Voramate Pasharawipas| Yong Li

Introduction

Recognizing and forecasting changes in climatic conditions is essential for human activities and productions. For certain communities, their livelihoods can rely directly on weather conditions. The increase in climate change and variability has been causing problems in our society, especially in tribal and indigenous communities where building facilities and installing modern weather forecasting systems is quite difficult to implement due to the cost and maintenance by professionals. One of the alternative plans for these communities can be identifying weather conditions from color images using computer techniques. Outdoor security cameras are more affordable these days, which makes the computer vision interpretation more possible. Apart from rural communities, modern technologies such as robotic vision or vehicle assistant driving systems can also take advantage from the results of weather conditions.

The objective of this analysis is to understand and develop a Convolutional Neural Network (CNN) and Visual Geometry Group (VGG-16) models to classify weather conditions of images. The models can analyze and predict the weather condition based on a matrix of pixels of the image using a deep learning algorithm. Based on the models, we can have an effective means of distinguishing a large number of images with valid classifications.

Data

The original dataset can be obtained from the Mendeley website. It is the multi-class weather image dataset for image classification which consists of 1122 images of different weather conditions. Creating our own datasets that represent the different weather conditions is the initial step to begin this analysis. Labels of the weather conditions such as “sunrise”, “sunshine”, “rain”, and “cloudy” are created based on the filename of each image. Each image is redirected according to its visual class after being subdivided into training and testing sets.

Example of Rain image

rain130.jpg

rain131.jpg

rain132.jpg

rain135.jpg

rain136.jpg

rain137.jpg

rain140.jpg

rain141.jpg

rain142.jpg

Example of Cloudy image

cloudy17.jpg

cloudy18.jpg

cloudy19.jpg

cloudy22.jpg

cloudy23.jpg

cloudy24.jpg

cloudy27.jpg

cloudy28.jpg

cloudy29.jpg

Example of Sunrise image

sunrise152.jpg

sunrise153.jpg

sunrise154.jpg

sunrise157.jpg

sunrise158.jpg

sunrise159.jpg

sunrise162.jpg

sunrise163.jpg

sunrise164.jpg

Example of Shine image

shine180.jpg

shine181.jpg

shine182.jpg

shine185.jpg

shine186.jpg

shine187.jpg

shine190.jpg

shine191.jpg

shine192.jpg

The following table presents the sample size of the train, validation, and test datasets based on each class:

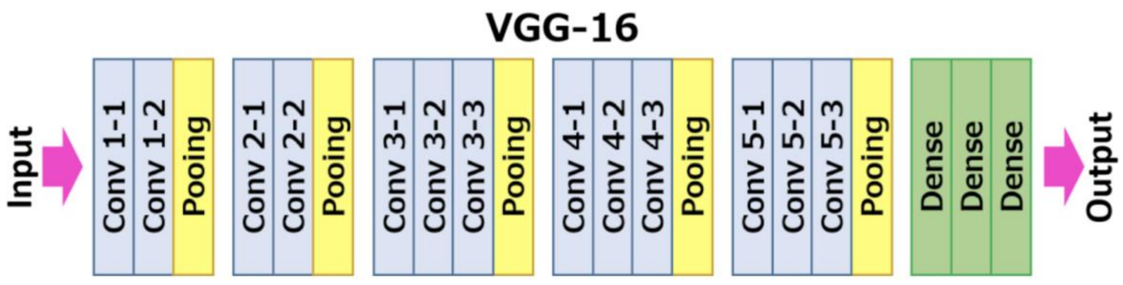
Class	**Train**	**Validation**	**Test**
Rain	139	29	45
Cloudy	193	55	52
Sunrise	236	56	64
Shine	149	40	64
Total	717	180	225

Since we have a rather imbalanced data with more sunrise images as shown below in the distribution graphs, we optimized our models to mainly maximize the F1-score while also considering other metrics such as recall, precision, and accuracy.

Methodology (examples)

VGG-16:

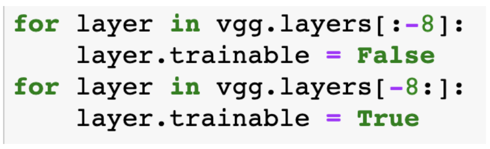
We use VGG-16 as a baseline model for our weather image classification project. VGG-16 is a convolutional neural network that consists of 16 layers with convolutional layers, max-pooling layers, and fully connected layers. In compiling the network, Adam is used as a momentum-based optimizer and categorical cross-entropy is selected as the loss function for the multiple classifications. The model is trained for 2 epochs (iterations) with a batch size of 64. Finally, the trained model is evaluated for the test set to determine the accuracy, precision, recall rates, and F1-score.



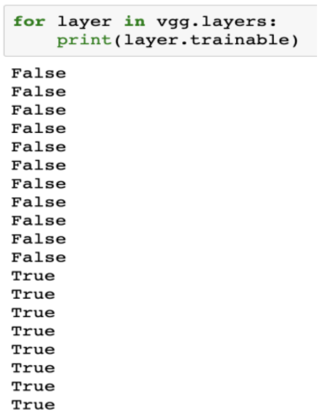
VGG-16 Layer Definition

Source: https://medium.com/datadriveninvestor/cnn-architecture-series-vgg-16-with-implementation-part-i-bca79e7db415

It’ s famous because the Oxford team made the model open sources and we are available to train the model ourselves. The convolutional layers are only 3*3 filter size which was much smaller than other pre-trained CNN models, but VGG has more layers which are much deeper. It was one of the best CNN architecture in localization and classification tasks in ImageNet challenge. The previous neural network usually has a bigger receptive field which were (7*7, 11*11), but VGG is deeper than those CNN models. We want to try this model and add some fully connected layers and test if it’ s going to work well or not on our weather condition recognition problems. We loaded the VGG from Keras application and set the last eight layers were trainable but fixed the previous layers. Because VGG is trained on over 15 million high-resolution images to belong to 22,000 categories, we can utilize the VGG model and train to last eight layers to adjust our dataset.



```
for layer in vgg.layers[:-8]:
    layer.trainable = False
for layer in vgg.layers[-8:]:
    layer.trainable = True
```



```
for layer in vgg.layers:
    print(layer.trainable)
```

```
False
False
False
False
False
False
False
False
False
False
False
True
True
True
True
True
True
True
True
```

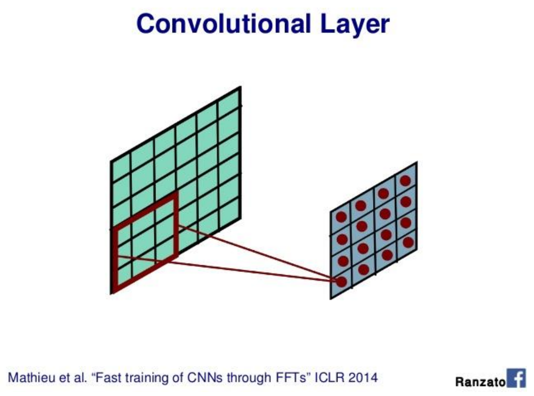
And finally, we add a fully connected layer to further customize the VGG model for our dataset.

Customized CNN:

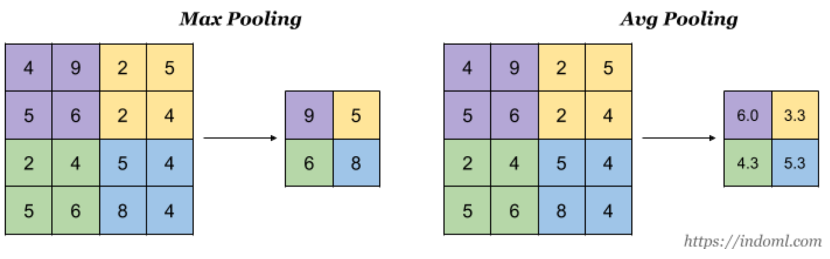
In this analysis, a deep learning approach is applied to analyze and predict the images. Regarding the brief explanation of deep learning models applied in this analysis, for example, the Convolutional Neural Network (CNN) model has the last node which uses the ReLU activation function. ReLU function is a half-rectified function that is, for all the inputs less than 0 the value is 0 while the value is retained if positive. In compiling the network, Adam is used as a momentum-based optimizer and categorical cross-entropy is selected as the loss function for the multiple classifications. The model is trained for 80 epochs (iterations). Finally, the trained model is evaluated for the test set to determine the accuracy, precision, recall rates, and F1-score.

To construct a CNN, we need to define:

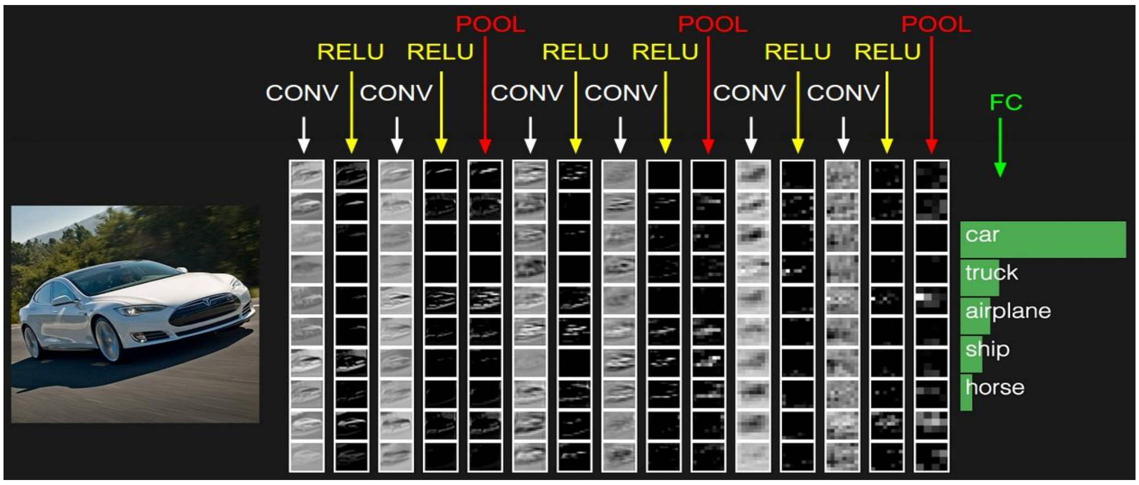
1. A convolutional layer: Apply n number of filters to the feature map. Relu activation function is used to add non-linearity to the network. Relu is given by f(x) = max(0,x).


Convolutional Layer
Mathieu et al. "Fast training of CNNs through FFTs" ICLR 2014
Ranzato

2. Pooling layer: The next step after the convolution is to downsample the feature map. Max pooling layer is used here to divide the feature maps into subregions and keeps only the maximum values. (Non-overlapping kernel as shown below.)

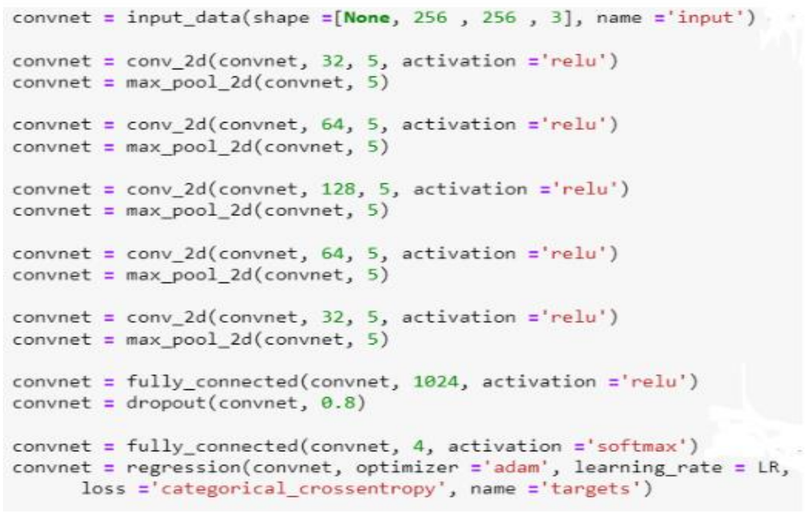


3. Fully connected layers: All neurons from the previous layers are connected to the next layers. The CNN will classify the label according to the featur es from the convolutional layers and reduce it with the pooling layer.



Source: https://cs231n.github.io/convolutional-networks/

Our CNN Architecture



```
convnet = input_data(shape =[None, 256 , 256 , 3], name ='input')

convnet = conv_2d(convnet, 32, 5, activation ='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation ='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 128, 5, activation ='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 64, 5, activation ='relu')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation ='relu')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 1024, activation ='relu')
convnet = dropout(convnet, 0.8)

convnet = fully_connected(convnet, 4, activation ='softmax')
convnet = regression(convnet, optimizer ='adam', learning_rate = LR,
      loss ='categorical_crossentropy', name ='targets')
```

- Input layer
  - Take input with shape (Batch size, Image height, Image width, 3) for RGB image data
- 1st Convolutional layer
  - 32 filters of size 5x5 with relu activation
- 1st Max pooling layer
  - Max pooling size 5x5
- 2nd Convolutional layer
  - 64 filters of size 5x5 with relu activation
- 2nd Max pooling layer
  - Max pooling size 5x5
- 3rd Convolutional layer
  - 128 filters of size 5x5 with relu activation
- 3rd Max pooling layer
  - Max pooling size 5x5

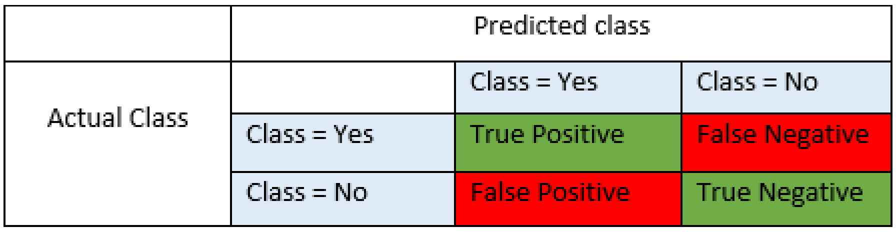
- 4th Convolutional layer
    - 64 filters of size 5x5 with relu activation
- 4th Max pooling layer
    - Max pooling size 5x5
- 5th Convolutional layer
    - 32 filters of size 5x5 with relu activation
- 5th Max pooling layer
    - Max pooling size 5x5
- 1st Fully connected layer
    - 1024 neurons with relu activation
- Dropout layer
    - Drop out 20 % of the output to avoid overfitting.
- 2nd Fully connected layer
    - 4 neurons with softmax activation to provide the output for each class to be between 0 and 1 and summation of the output of all 4 classes from 4 neurons equal 1.
- Estimator layer
    - Optimize Categorical cross-entropy loss function with Adaptive moment optimizer (Adam).

*** Note that: Padding of Convolutional layers are set to "same" as default, which means borders will be added to the matrix in the way that input shape will be the same as output shape.

Results

Metric used to evaluate the model

- Precision
    - the ratio of correctly predicted positive observations to the total predicted positive observations.
- Recall
    - the ratio of correctly predicted positive observations to all observations in the actual class
- F1 score
    - the weighted average of Precision and Recall.
- Accuracy
    - the fraction of predictions our model got right.
    -



	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

$$(10.1)\ \ Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

$$(10.2)\ \ Precision = \frac{T_p}{T_p + F_p}$$

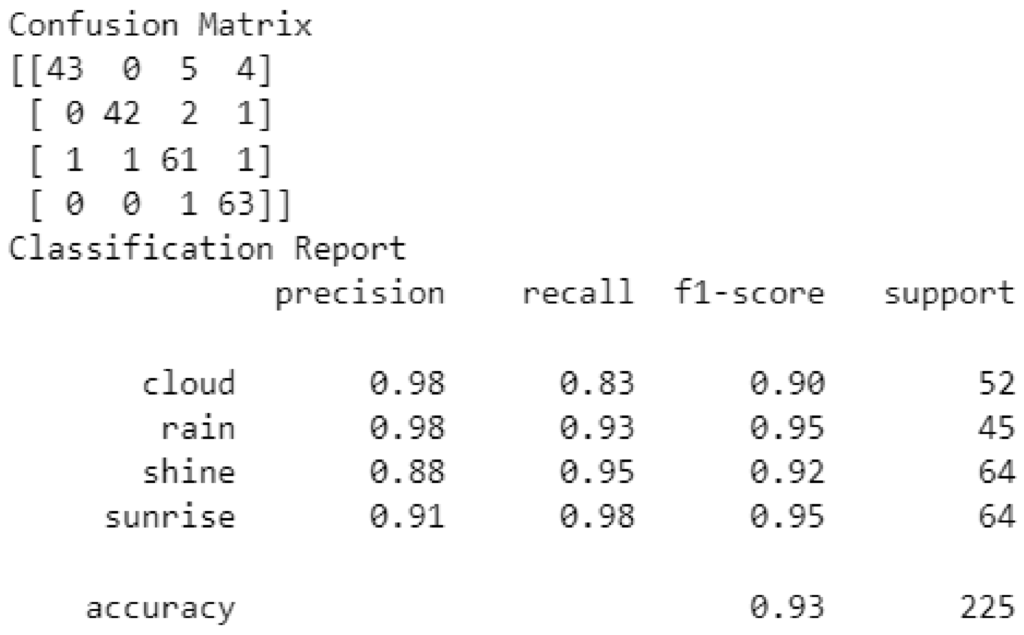
$$(10.3)\ \ Recall = \frac{T_p}{T_p + T_n}$$

$$(10.4)\ \ F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Source: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

The precision, recall, and f1-score rates for each of deep learning models are summarized in the tables below:

VGG-16:

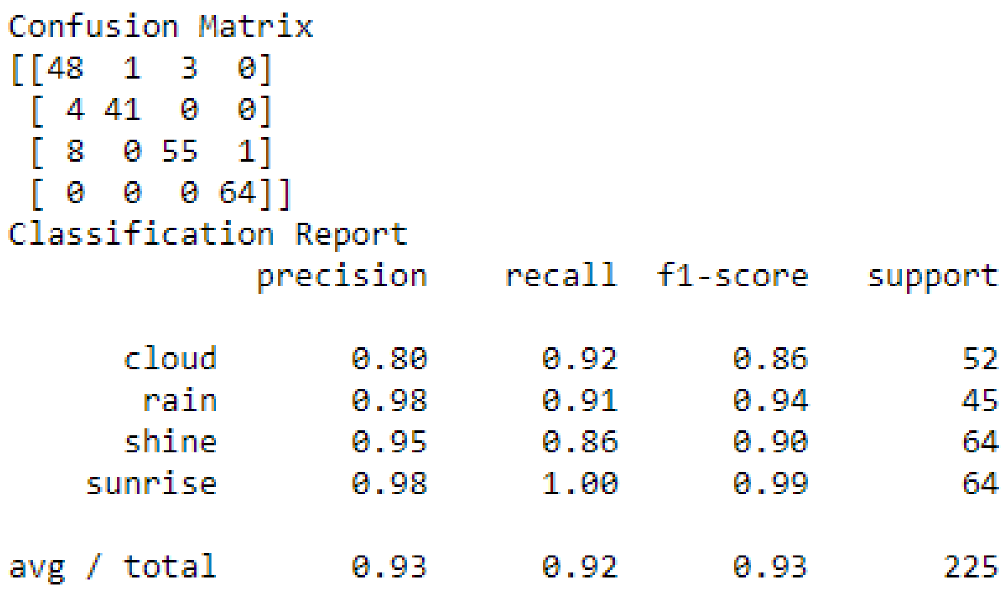


```
Confusion Matrix
[[43  0  5  4]
 [ 0 42  2  1]
 [ 1  1 61  1]
 [ 0  0  1 63]]
Classification Report
              precision    recall  f1-score   support

       cloud       0.98      0.83      0.90        52
        rain       0.98      0.93      0.95        45
       shine       0.88      0.95      0.92        64
     sunrise       0.91      0.98      0.95        64

    accuracy                           0.93       225
```

Customized CNN:



```
Confusion Matrix
[[48  1  3  0]
 [ 4 41  0  0]
 [ 8  0 55  1]
 [ 0  0  0 64]]
Classification Report
             precision    recall  f1-score   support

      cloud       0.80      0.92      0.86        52
       rain       0.98      0.91      0.94        45
      shine       0.95      0.86      0.90        64
    sunrise       0.98      1.00      0.99        64

avg / total       0.93      0.92      0.93       225
```

- Both models seem to have less ability to classify "shine" and "cloud" based on F1 score.
- VGG-16 seems to have less predicting power in identifying "cloud" class with least recall of 0.83 compared to other classes.
- VGG-16 seems to have predicting power in identifying "sunrise" class with the biggest recall of 0.98 compared to other classes.
- Customized CNN seems to have less predicting power in identifying "shine" class with least recall of 0.86 compared to other classes.
- Customized CNN seems to have predicting power in identifying "sunrise" class with the biggest recall of 1 compared to other classes.

Conclusion

In this analysis, we present possible approaches to detect and predict weather conditions from single images by implementing computer vision and deep learning methods. The models are capable of detecting 4 classes: shine, sunrise, rain, and cloudy. We intend to exemplify the application of deep learning methods and computer vision for identifying weather conditions from an image, which could be useful for autonomous applications especially in rural communities or modern technologies. The advantage of this analysis is in its practical application for identifying various conditions of more than just weather without pre-established constraints for processing images. This can be utilized for many purposes. For example, it may be usefu l for autonomous driving in cities or data automation for urban improvement schemes.

Two deep learning models are applied in this analysis. The first one is VGG-16, which is the pre-trained convolutional neural network model trained on a very large size of image dataset. The model is complex but powerful. It is a good initial step to retrain the model on your own dataset and obtain a high accuracy and prediction power. However, it takes a long time to train 1 epoch. If we have limited time and computational resources, this model might not be a good choice. Convolutional neural network is a very powerful model that is widely used in image processing and computer vision tasks. After training two deep learning models, CNN and VGG, on images of different weather conditions, the proposed models showed a strong performance in recognizing the different categories of a single image. Both of our models classify 4 different classes of weather image with up to 93% accuracy and above 85% f1 score for all classes.

The images used for training can be still limited, thus data augmentation techniques have been applied to enhance the training of each model. The development of this technique can enhance our training process and the overall performance of deep learning models.

For future work, we can experiment with different architectures of CNN models to enhance the overall performance of the model and allow further multi classifications to detect more weather conditions. To train CNN with limited time and computational resources, we can construct our model structure similar to VGG-16 with less number of layers and complexity. The size of the kernel for convolutional layer and pooling layer can be modified, and further methods can include applying different non-linear activation, pooling methods, or size of drop out percentage.