Individual Report

Introduction

In this project, our group is going to classify Scenes around the globe into one of six possible classes using Deep Neural Networks. The application of scene classification could range from organization of photos in Smartphones; assist in growth of country's economy through Tourism Planning and so on.

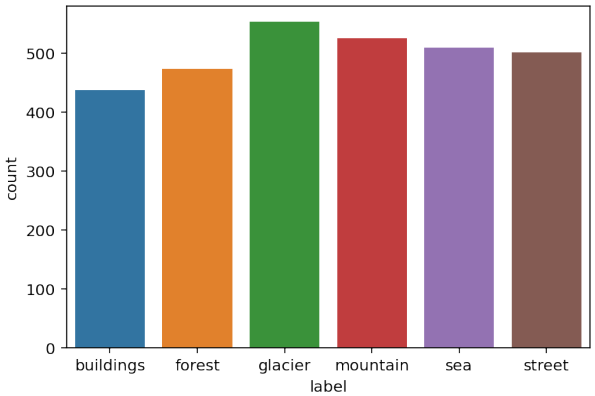
For shared works, Saurav's work on pre-trained models are prerequisite for my further predictions. Jiaoyue's work on data augmentation are also strong supports for model selection. And there are more details of works we shared with each other during the project. For me, my works contains data visualization, self-trained model constructing, and model predictions.

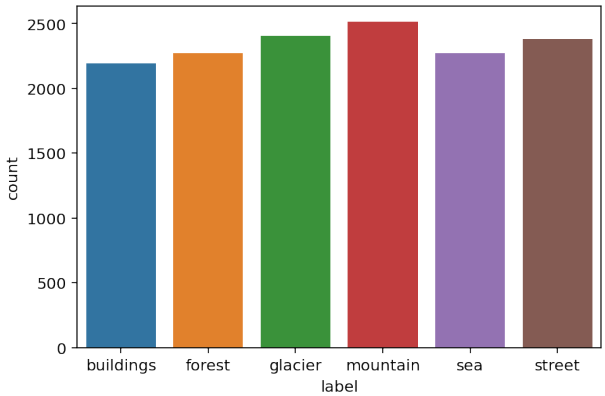
Description of individual work

Data visualization

- Dataset bar chart

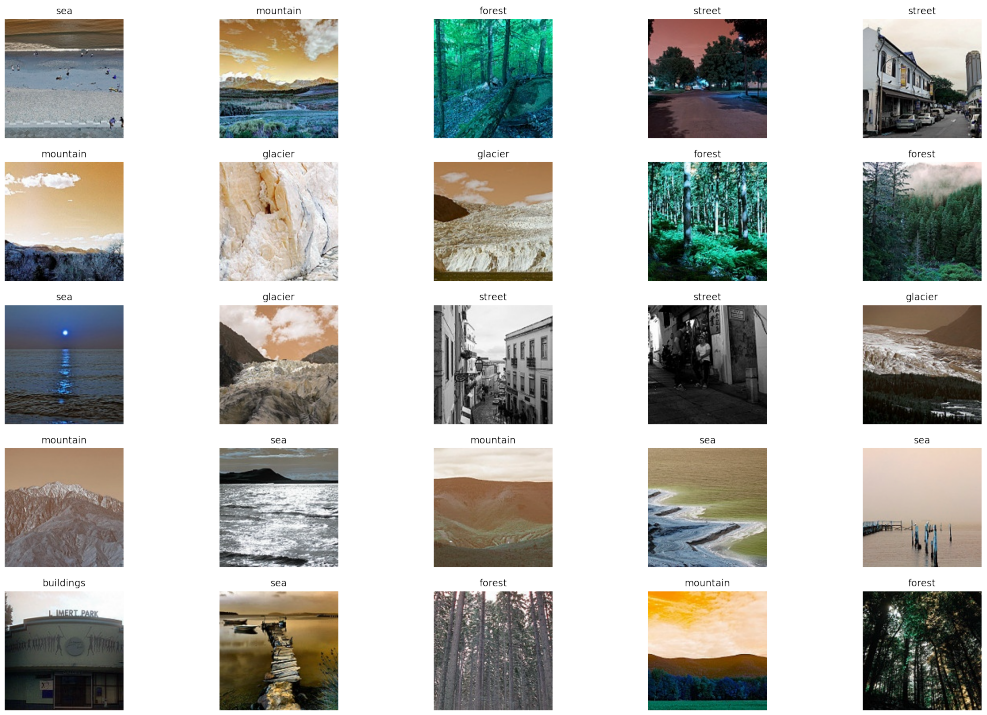
There are total of 25,000 images captured by Jan Bottinger. Out of them only 17,000 were labelled. The six possible categories are - Buildings, Forest, Glacier, Mountain, Sea or Street. The figures below show the proportion of each category in the train dataset and test dataset.





- Example of training images

Images from each class label were evenly distributed while we divide the data. And here's a sample of images randomly selected in our dataset.



Self-Trained model

I created a custom CNN Network using the Keras Framework. The structure of the model can be seen below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 200)     5600
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 146, 146, 180)     324180
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 29, 29, 180)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 27, 27, 180)       291780
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 25, 25, 140)       226940
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 23, 23, 100)       126100
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 21, 21, 50)        45050
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 50)          0
_________________________________________________________________
flatten (Flatten)            (None, 800)               0
_________________________________________________________________
dense (Dense)                (None, 180)               144180
_________________________________________________________________
dense_1 (Dense)              (None, 100)               18100
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 306
=================================================================
Total params: 1,187,286
Trainable params: 1,187,286
```

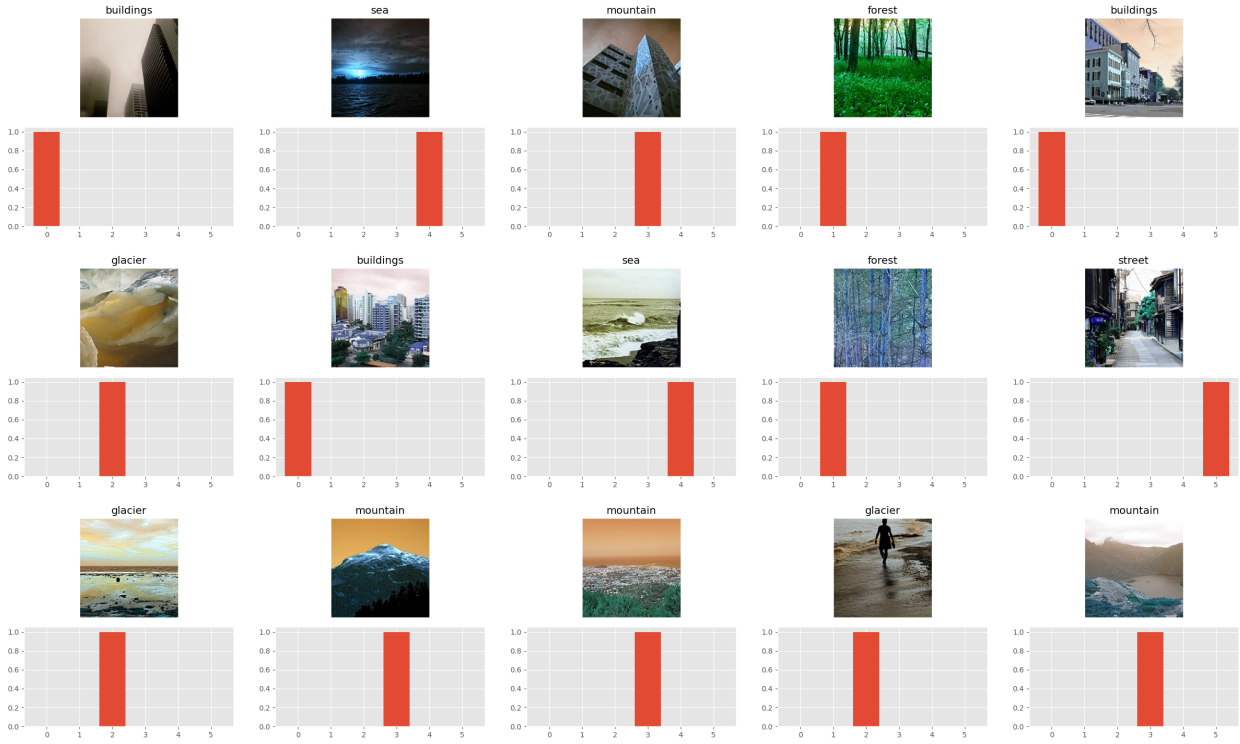
This Self-Trained model has six Conv2D layers, all of which had kernel size of (3, 3) and stride of 1 were connected end by end. Each had Relu activation unit, and second Conv2D layer and fifth Conv2D layer are followed by Max Pooling with kernel of (2, 2). The output was then flattened and passed into three Dense layers, each with 180 filters, 100 filters and 50 filters. Finally, with the dropout rate of 0.5, they were all Normalized again and passed into SoftMax Unit having filters equal to number of class labels. Learning Rate was set to 0.001 and with the batch size of 512 the model was trained for 40 epochs with Callback monitoring loss of validation set. And the Self-trained model accuracy is 86.09%.

Model	Accuracy	Cohen Kappa Score	Macro F1 Score
Self_Trained Model	86.09%	0.77	0.83

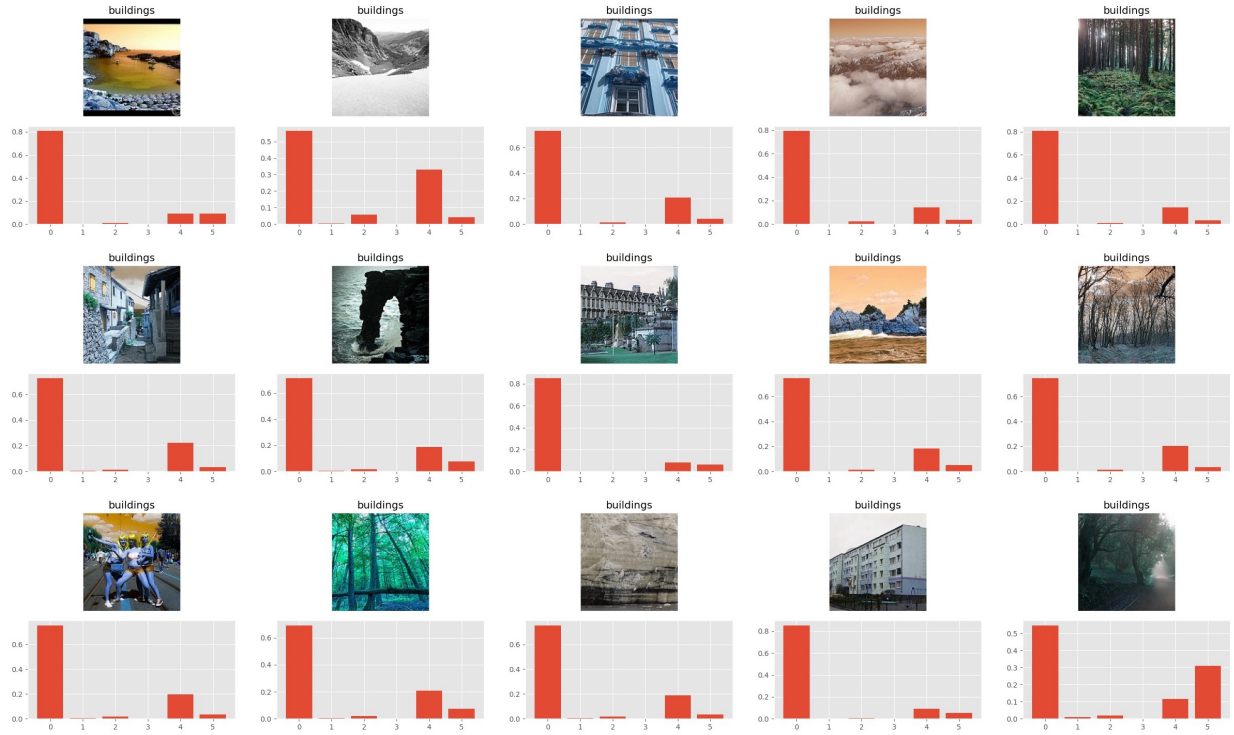
Model Predictions

In order to find out the best model for our project, we decided to compare the prediction ability about the top3 models, vgg16, vgg19, ResNet and our selftrained model. The prediction results are shown below:

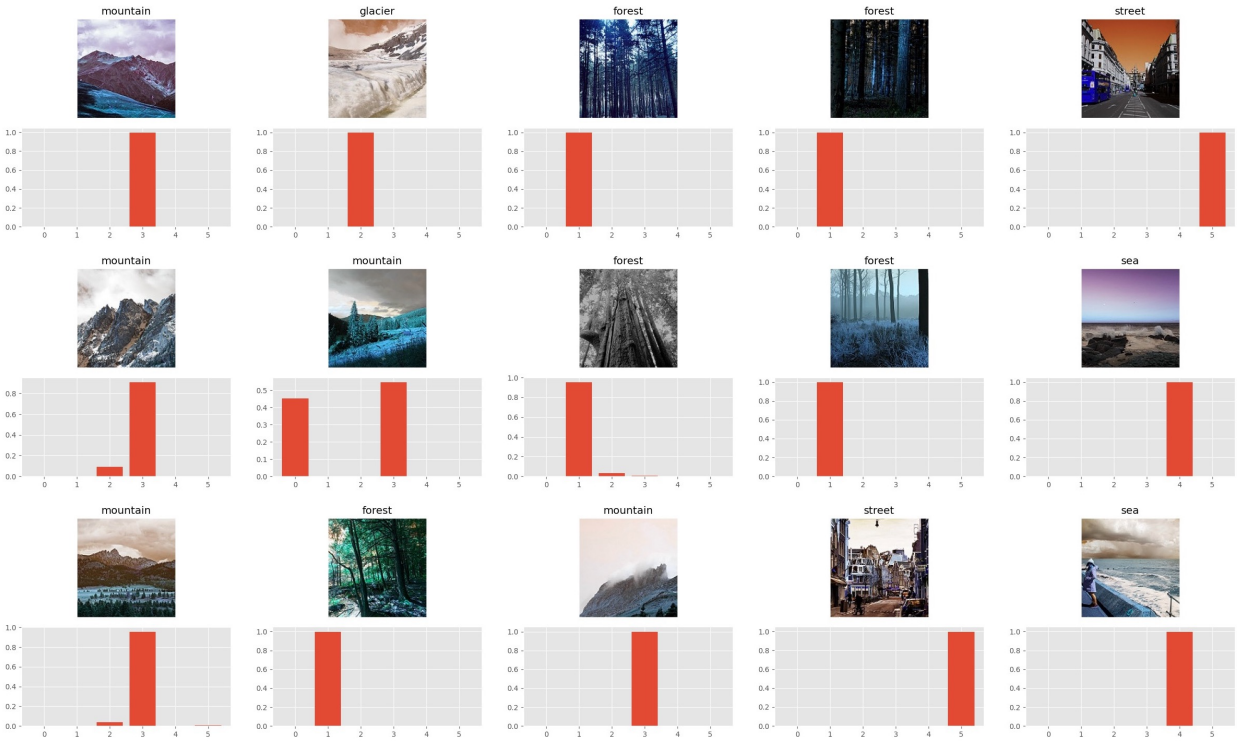
Self-Trained Prediction



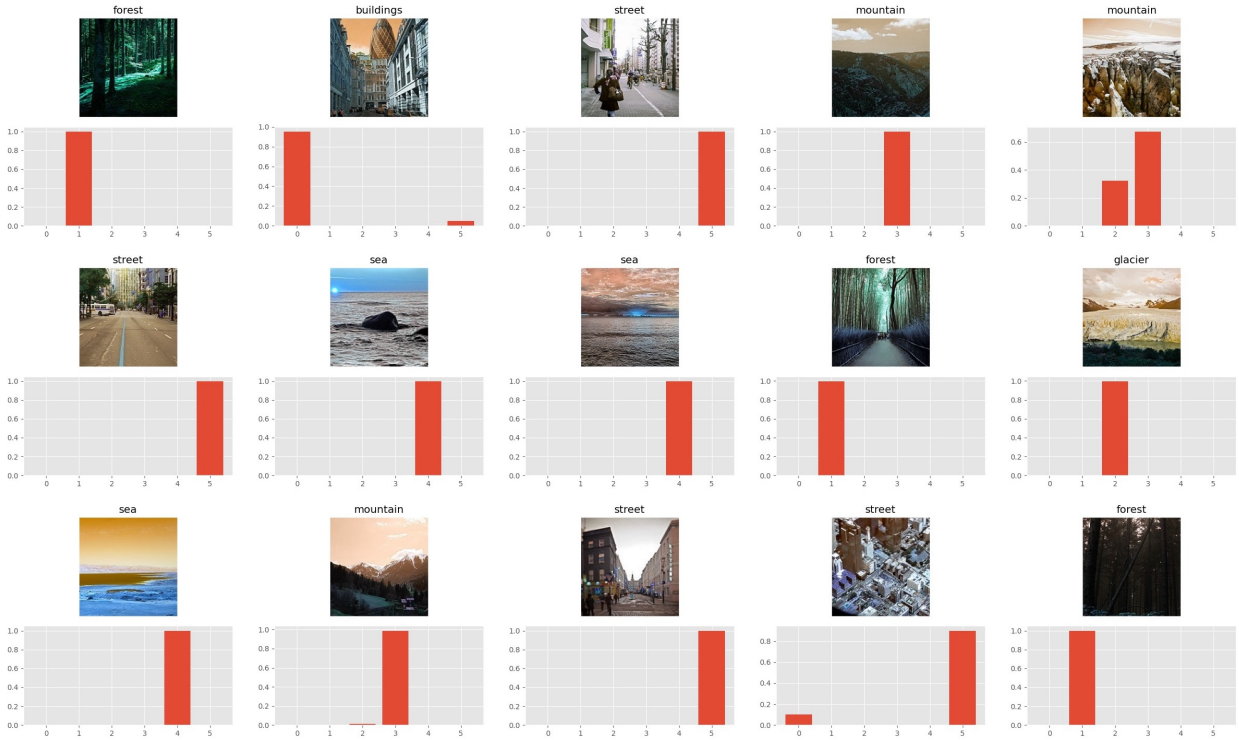
ResNet Prediction



Vgg16 Prediction



Vgg19 Prediction



The result in our self-trained model is good, but it seems to mix tall buildings and mountains to some extent, and the same situation happened in class sea and glacier. It may because these pictures have some similar features. Then in ResNet results, our previous best model did not show a good prediction as we were expected, as in the examples we show here, it mispredict a lot of them. That's not what we want from it. In the end, the Vgg16 and Vgg19 show very good prediction. And given Vgg16 has better accuracy, we decided to choose Vgg16 as our best model.

Summary

By comparing the prediction results between Self-Trained model and the pre-trained model we selected, the pre-trained models performed much better than the Self-Trained CNN model. This could be because pre-trained models were trained using huge dataset with very large number of trainable parameters.

In this project, I have learned how to load data from different directory and create the class label by using the folder name. In addition, I have getting familiar with fine tune a neural network, and check the performance of the model in different ways. Furthermore, I have learn how to create an image grade for prediction results, which will make the results much more readable. And last but not least, teamwork is another very important skill, worthy respect.

Percentage of outside code

32%

Reference:

- https://github.com/amir-jafari/Deep-Learning
- https://www.jianshu.com/p/502b4c637c0e