Selected Topics in Visual Recognition using Deep Learning
HW1 Natural Scene Image Classification Report

0856049 吳毓軒

GitHub repository link

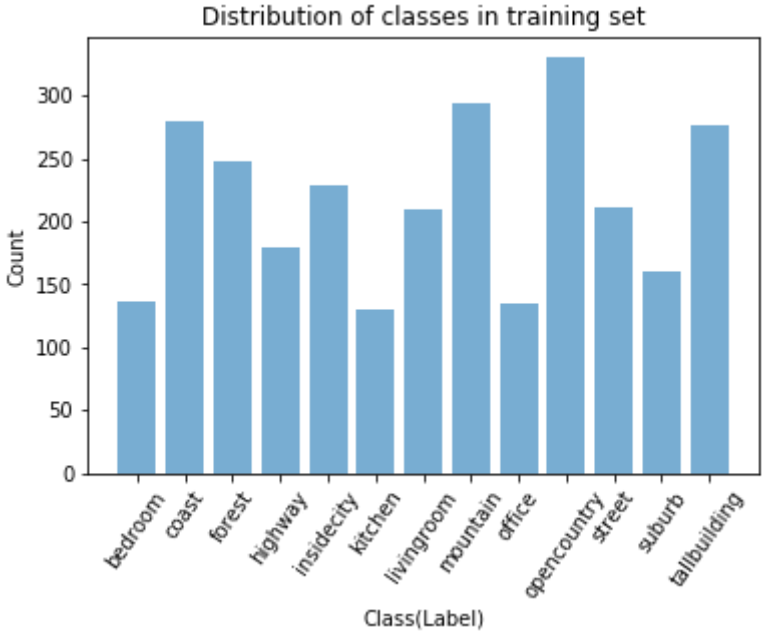
Link: https://github.com/yuhsuan1203/VRDL.git

There is only one jupyter notebook ***VRDL_HW1.ipynb*** in the directory ***HW1/*** in this repository. All of the code is written in this notebook and the output of each cell can also be seen in the file.

(P.S. Sometimes github fails to render the ipynb notebooks, so I provide an alternative link to view the same code: https://nbviewer.jupyter.org/github/yuhsuan1203/VRDL/blob/master/HW1/VRDL_HW1.ipynb)

Introduction

Image classification is a popular problem in computer vision and the commonly used technique is to build a deep learning model. In this homework, we have 2819 training scene images, 1040 test images and 13 classes. The goal is to train a convolution neural network to classify these images into correct class. The distribution of the training data in each class is shown below.



Methodology

- Data Preprocess

  The data preprocessing technique I used in this problem was directly taking advantage of the *preprocess_input* function provided by Keras. It would scale pixels between -1 and 1 through normalization. After this operation, all input image would be resized to 256 x 256 shape. In addition, I did some data augmentation before resizing the images through the function given by Keras. In this work, only horizontally flipping the images at random was implemented.

- Model Architecture

  Due to the limited number of training data, it is hard to achieve high accuracy if a model is trained from scratch. Therefore, I decided to use the pre-trained model that gives us some useful pre-trained weights.

  After evaluating the performance of a number of existed models, I decided to use model ensemble technique in this work. Three models trained on ImageNet are considered, which are VGG16, VGG19 and ResNet50. Because of the difference of VGGNet and ResNet, different dense layers and softmax layer are added to them.

  For VGG16 and VGG19, both of them are added two dense layers with relu as activation function, one layer with 512 and the other with 128 output neurons. For ResNet50, three dense layers with relu as activation are added. The output neurons are 2048, 512 and 128 respectively. In order to prevent overfitting, I also added a dropout layer after each dense layer with the same dropout rate 0.35. Except the layers added by myself, the weights between each other layer are non-trainable.

  Each model is trained for 25 epochs individually. Finally, when making a prediction on the test data, the final result is determined by voting from the result of each model.

- Hyperparameters

  There are a set of parameters to be tuned in this work, such as epochs, batch size, dropout rate, and so on. After doing a lot of experiments, the following value for each parameters is used. The combination of these values can gain the highest accuracy in my work. The accuracy can also be stable when I trained another set of three models.

  ★ learning rate: 0.001
  ★ epochs: 25
  ★ batch size: 64
  ★ dropout rate: 0.35

Summary & Findings

Some of the images in the training set are too ambiguous and thus hard to predict even by human. For example, mountain, forest and opencountry are similar in some cases. If zooming is implemented when doing the data augmentation, the image may become confusing. Another example is the similarity between livingroom and bedroom. The ambiguity can decrease the classification performance.

As mentioned in Methodology, I have tried several combinations of hyperparameters. The training accuracy can converge very quickly even in 5 to 10 epochs due to the strength of the pre-trained models and the small number of trainable parameters.

In my work, each model has a quite good performance when evaluated independently. Their performances are only slightly lower than that of combining them altogether. The difference is about 1% to 2%. I also found that VGGNet performs well on most of the class in this problem whereas ResNet may be more accurate than VGGNet sometimes.

I have tried two methods to determine the final result when combining three predictions from each model. One is to calculate the sum of probabilities of all models and the other is via voting from them. There is no obvious difference between these methods.

The highest accuracy the model achieves is 0.94807, which is via voting method, while total probabilities method also gains 0.94711 accuracy by using the same set of three models. The improvement by using more than one model proves that model ensemble method can be more powerful.