DSA5203 Assignment 3 Report

Group member: Yao Yuhan (A0236301W), Zhang Youyang (A0251290M)

1. Introduction

Scene categorization is a critical task in computer vision that involves classifying images into specific categories such as mountains, beaches, buildings, and forests. In this technical report, we will describe the process of developing a TensorFlow CNN model for scene categorization using the 15-Scene dataset. Convolutional Neural Networks (CNNs) have shown promising results in scene categorization tasks due to their ability to learn features directly from raw input data.

2. Dataset generating and preprocessing

Dataset review

The dataset contains 15 categories of natural scenes such as forests, mountains, highways, and coastlines. The dataset contains a total of 1500 images, with an average of 100 images per category.

Dataset splitting

For each category, we select 10 images as the validation set, and use the rest as the training set. In total, we have 1350 images for training and 150 images for validation.

Preprocessing

Before training a CNN model, it is important to preprocess the images to ensure they are in a suitable format, which typically involves resizing the images to a common size and normalizing the pixel values. Here we uniformly resize the images into sizes of (224,224) and normalize the pixel values to the range [0, 1].

Data augmentation

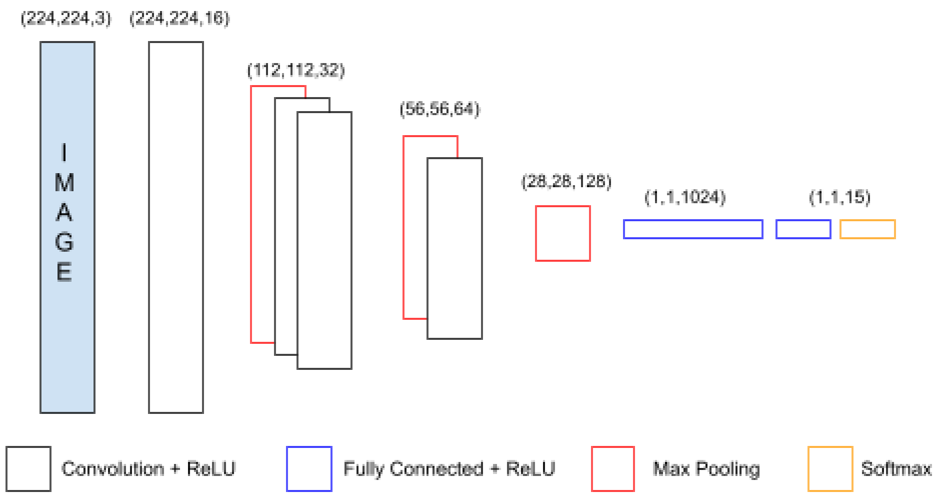
To strengthen the generalization ability of our model, we performed data augmentation while generating the data, which includes horizontal shifting, vertical shifting, zooming and flipping.

Based on these preprocessing, we can get our training and validation dataset by calling the `'get_dataset'` function accessing from different directories.

3. Model architecture

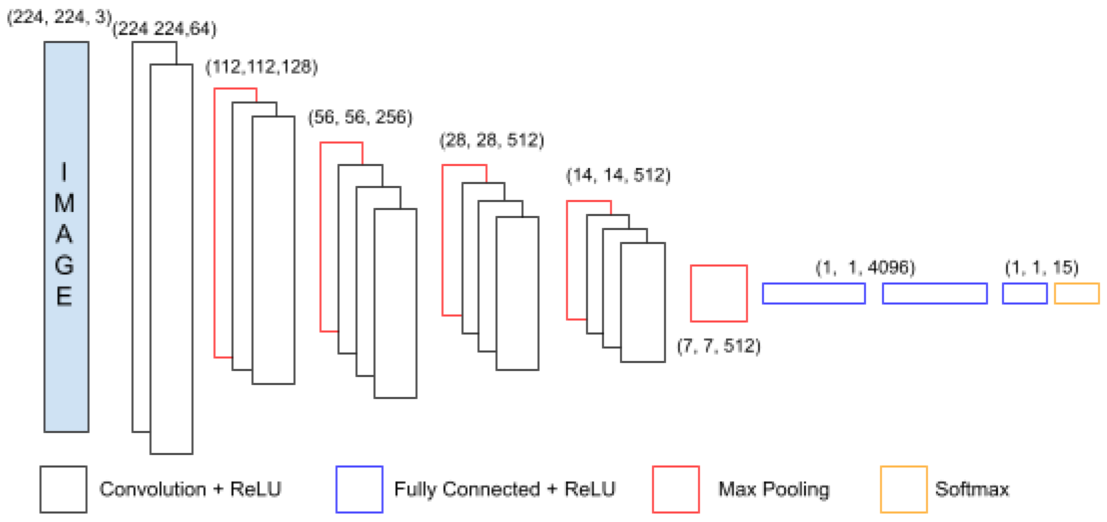
1) Basic model

Our basic model architecture consists of 6 layers, including 4 convolution layers, 3 max pooling layers, and 2 fully-connected layers, where the final fully-connected layer is replaced by a layer with 15 neurons (one for each class) using softmax activation. Our chosen number of filters is 16 and the filter size is (3, 3).



2) VGG16 Model

For the CNN model, we try another popular architecture: VGG16, which is proposed in [1] and described in [2]. It consists of 16 layers, including 13 convolutional layers, 5 max pooling layers, and 3 fully connected layers.
To adapt the VGG-16 model to our dataset, we replaced the final fully connected layer with a new layer with 15 neurons (one for each class) and used softmax activation. We also froze the weights of the first 15 layers to prevent overfitting and fine-tuned the remaining layers during training.



4. Training and evaluation

Based on our two model architectures, we have applied hyperparameter tuning strategies to improve model performances. Validation accuracy was carefully monitored to prevent overfitting. We first tried tuning activation functions, loss functions, and different pooling methods, but they turned out to have little influence on model performance. Our main focuses were on regularization and optimizer learning rate.

For our basic architecture, we set a learning rate scheduler that started learning rate decay after 20 epochs. The initial learning rate was set to 0.001. For our VGG16 architecture, we first tried it with fixed learning rates (0.001 and 0.0001), and then the same scheduler from the basic model was applied to it with initial learning rate as 0.001.

All models were trained for 30 epochs using a batch size of 32 and the Adam optimizer. Additionally, a L2 regularization of strength 0.001 and a dropout layer of rate 0.15 were added to the basic architecture. The table below shows the validation accuracies of all trained models. Both basic and VGG16 architectures reach highest validation accuracies with learning rate decay, so we include learning rate decay in our final models.

To test our model, you could select one model (‘basic’ or ‘vgg’) as one of the `‘train’` function inputs, which will train the corresponding model with all tuned hyperparameters. After training, the training accuracy will be printed and the model will be saved in the directory so that it could be fetched in the `‘test’` function. The test function will return the testing accuracy.

Model	**Validation accuracy**
Basic architecture	0.433
Basic architecture (lr=0.001 + decay)	0.642
VGG16 (lr=0.001)	0.817
VGG16 (lr=0.0001)	0.794
VGG16 (lr=0.001 + decay)	0.833

5. Conclusion

In conclusion, the dataset provides a challenging task for image classification, but with appropriate preprocessing and a suitable CNN architecture, it is possible to achieve high accuracy on this dataset. By fine-tuning a pre-trained VGG-16 model and using techniques such as freezing layers and monitoring validation accuracy, we were able to achieve an accuracy of 80.5% on the validation set.

However, we still have a lot to explore and improve. For example, resizing the images to a common size may result in lower sample quality. More methods can be tried to preprocess the data, such as SSP-Net[3]. Also, more advanced models can be used for this image classification task to further improve the accuracy, such as Inception V3 architecture[4], ResNet[5] and so on.

References

[1] Simonyan, Karen, and Andrew Zisserman. (2014) "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556.
[2] Frossard, Davi. (2017) "VGG in TensorFlow." VGG in TensorFlow· Davi Frossard.
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. (2014) "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." arXiv preprint arXiv:1406.4729.
[4] Xiao, Jianxiong, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. (2010) "Sun database: Large-scale scene recognition from abbey to zoo." In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 3485-3492. IEEE.
[5]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).