DSCI552 Project Report

Kayvan Shah, Zuqi Li, Qianyou Wang

April 2023

Abstract

This project aims to address the difficulty of landmark image classification by predicting both the category and landmark names of given images. The provided dataset is organized into six categories, with five landmarks in each category. Due to the small dataset size, transfer learning utilizing pre-trained models, EfficientNetB0 or VGG16, is explored. To handle the classification problem of landmark categories, a single multi-classifier is proposed. Multi-task learning technology is also considered to incorporate the landmark and category classification into one model. EfficientNetB0 is chosen as the pre-trained model due to its higher accuracy than VGG16.

The data is split into train, validation, and test datasets, and data augmentation is used to expand the data to overcome the small size of the dataset. For the optimizer, we used the Adam optimizer with a learning rate of 0.001. For the loss function, we used categorical cross-entropy, which is suitable for multi-class classification problems like ours. As for the metrics, we monitored f1-score during training. After training and fitting, the model's behavior is analyzed by visualizing the learned features and identifying if the model still performs well on other test datasets.

1 Introduction

Image classification has always been an important topic in machine learning that involves assigning a label to an image based on its content. In this project, we aim to address the difficulty of landmark image classification, where the task is to predict both the category names and landmark names of given images.

The dataset provided is organized into a two-level hierarchy structure. It is categorized into six categories: Gothic, Modern, Mughal, Neoclassical, Pagodas, and Pyramids, and for each category there are 5 landmarks for a total of 30 landmarks, with each landmark having 14 images.The small dataset size make it difficult to avoid overfitting with deep learning models.Therefore it's better for us to explore transfer learning utilizing pre-trained model instead of training conditional neural networks from scratch. However, it is challenging to achieve high accuracy on small datasets, and choosing the right pre-trained model is crucial. Due to the resource usage limitations, we can only apply EfficientNetB0 or VGG16 to complete the classification task.

Under these limitations, there are still several issues to consider.

There are 6 categories that need to be classified in the task, so we need to decide whether to use one 6-classifier or 6 binary classifiers. Based on the research, we decide that for the classification problem of landmark categories, a single multi-classifier is proposed. The reasons are as follows. Firstly, Landmarks usually belong to one of several categories, and there are often correlations between categories. It is better to use a single multi-classifier so that these correlations can be captured more accurately and the accuracy can be improved. Secondly, The categories of landmark buildings are usually relatively balanced. Even though some styles have more landmarks than others, the difference in numbers between them is not too great. Therefore, there is no need to use multiple binary classifiers to deal with the class imbalance problem. Thirdly, the classification of landmark categories usually needs to consider the overall characteristics, shape, structure and other aspects of the building. Using a single multi-classifier can process all features at one time, and can be more efficient in data processing when training and testing. In conclusion, the classification problem of the landmark category is suitable to be handled by a single multi-classifier.

Another issue we need to analyze is whether the landmark classification benefits from knowing the output of category classification. Since the data set has a hierarchical structure and landmarks and categories are strongly associated, if the landmark classification is completed, the category classification has actually been completed. Then the two tasks can be regarded as related tasks, multi-task learning technology can be considered to incorporate two tasks into one model for training and optimization. Multi-task learning (MTL) is a machine learning technique that aims to improve the performance of models by learning multiple related tasks simultaneously. In traditional single-task learning, the model is only trained for a specific task, while in MTL, the model needs to learn multiple tasks at the same time, and there is some association or dependency between tasks.

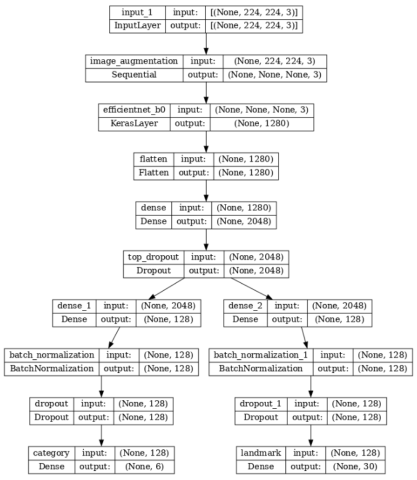
Before constructing the model, we need to choose the pre-trained model between EfficientNetB0 and VGG16. Based on the research, EfficientNetB0 requires larger computing resources than VGG16 in general because it has more network layers and parameters. But the accuracy of EfficientNetB0 is higher than that of VGG16 according to the implementation on the ImageNet dataset. This is because EfficientNetB0 adopts a method based on network scaling, which can improve the accuracy and generalization ability of the model while reducing model parameters and computation. At the same time, EfficientNetB0 also uses some effective techniques to optimize the model, such as using the Swish activation function and adaptive learning rate. In conclusion, EfficientNetB0 has higher accuracy than VGG16 on the ImageNet dataset and we expect this to be similar on our given dataset.

Based on the discussion about the issues and challenges we face, we will develop and evaluate transfer learning models with EfficientNetB0 and investigate the effect of different hyperparameters on model performance, such as learning rate, batch size, and optimizer. And also, analyze the model's behavior by visualizing the learned features and identifying which parts of the images the model focuses on for classification.

2 Method

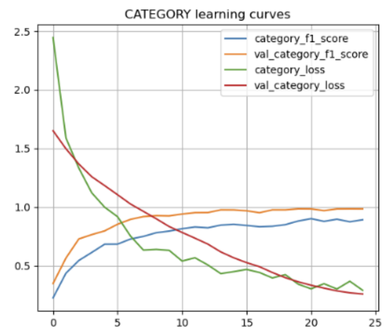
First, we need to process the raw data of 420 landmarks that we have and import it into our dataframe and then create categorical labels for each category and landmark. Then, we use stratified split to split the dataset into train dataset, validation dataset and the actual test dataset with the same number of images per label. We first use 0.1 split for test and then on the resulting dataset we use 0.2 split for validation. It means the train dataset includes 302 samples, the validation dataset includes 76 samples, and the test dataset includes 42 samples. After the split, we convert these dataframes into Tensorflow datasets to be more efficient in preprocessing and easier to integrate into a training pipeline(for lazy loading).

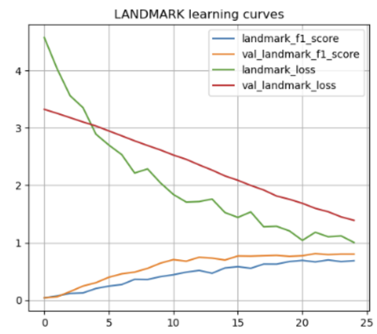
After we have processed all our data, we start building the model. Since the dataset we have is pretty small, there is a large possibility that there exists overfitting in our model. In order to avoid it, we use data augmentation to tweak the images in order to expand the data and overcome the small size of our dataset. After augmenting the data, we now have the input layer and the augmentation layer. Our full model is presented as shown on the right:

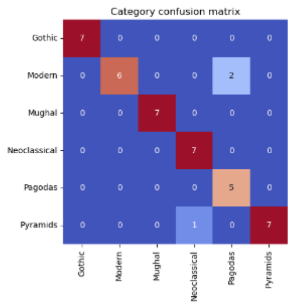

input_1 | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
image_augmentation | input: | (None, 224, 224, 3)
Sequential | output: | (None, None, None, 3)
efficientnet_b0 | input: | (None, None, None, 3)
KerasLayer | output: | (None, 1280)
flatten | input: | (None, 1280)
Flatten | output: | (None, 1280)
dense | input: | (None, 1280)
Dense | output: | (None, 2048)
top_dropout | input: | (None, 2048)
Dropout | output: | (None, 2048)
dense_1 | input: | (None, 2048)
Dense | output: | (None, 128)
dense_2 | input: | (None, 2048)
Dense | output: | (None, 128)
batch_normalization | input: | (None, 128)
BatchNormalization | output: | (None, 128)
batch_normalization_1 | input: | (None, 128)
BatchNormalization | output: | (None, 128)
dropout | input: | (None, 128)
Dropout | output: | (None, 128)
dropout_1 | input: | (None, 128)
Dropout | output: | (None, 128)
category | input: | (None, 128)
Dense | output: | (None, 6)
landmark | input: | (None, 128)
Dense | output: | (None, 30)

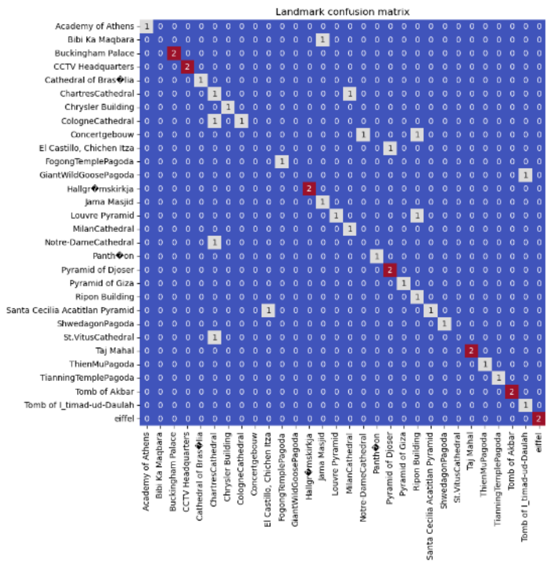
3 Result

With the model ready, the next step is to train and fit the model. We use 25 epochs here to obtain a result of f1-score 0.9848 for category validation, and a result of f1-score 0.8030 for landmark validation. After training, we also obtain learning curves for loss function and f1-score and the confusion matrix for each label.

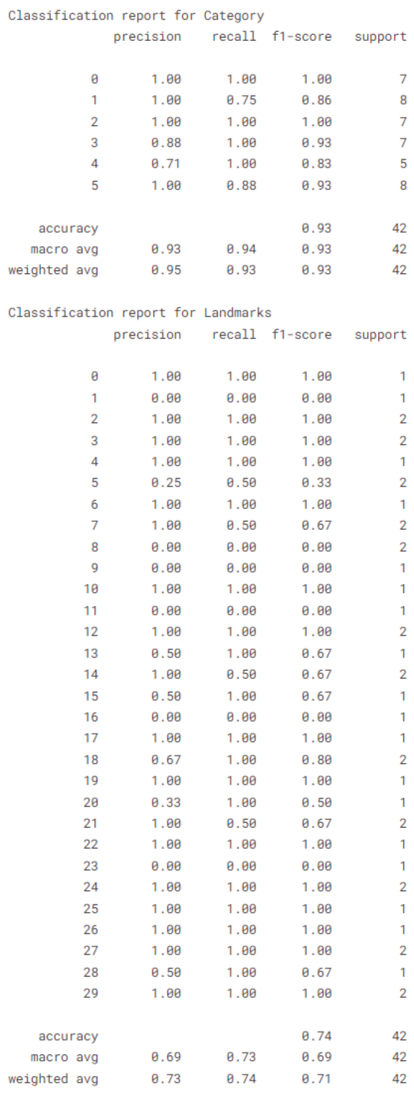








Moreover, we also have our classification reports that contain precision, recall, f1-score, and support for categories and landmarks. As shown on the right, we can see the we have reached an average accuracy of 0.93 for category classification, and an average accuracy of 0.74 for landmark classification.



```
Classification report for Category
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       1.00      0.75      0.86         8
           2       1.00      1.00      1.00         7
           3       0.88      1.00      0.93         7
           4       0.71      1.00      0.83         5
           5       1.00      0.88      0.93         8

    accuracy                           0.93        42
   macro avg       0.93      0.94      0.93        42
weighted avg       0.95      0.93      0.93        42

Classification report for Landmarks
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       0.00      0.00      0.00         1
           2       1.00      1.00      1.00         2
           3       1.00      1.00      1.00         2
           4       1.00      1.00      1.00         1
           5       0.25      0.50      0.33         2
           6       1.00      1.00      1.00         1
           7       1.00      0.50      0.67         2
           8       0.00      0.00      0.00         2
           9       0.00      0.00      0.00         1
          10       1.00      1.00      1.00         1
          11       0.00      0.00      0.00         1
          12       1.00      1.00      1.00         2
          13       0.50      1.00      0.67         1
          14       1.00      0.50      0.67         2
          15       0.50      1.00      0.67         1
          16       0.00      0.00      0.00         1
          17       1.00      1.00      1.00         1
          18       0.67      1.00      0.80         2
          19       1.00      1.00      1.00         1
          20       0.33      1.00      0.50         1
          21       1.00      0.50      0.67         2
          22       1.00      1.00      1.00         1
          23       0.00      0.00      0.00         1
          24       1.00      1.00      1.00         2
          25       1.00      1.00      1.00         1
          26       1.00      1.00      1.00         1
          27       1.00      1.00      1.00         2
          28       0.50      1.00      0.67         1
          29       1.00      1.00      1.00         2

    accuracy                           0.74        42
   macro avg       0.69      0.73      0.69        42
weighted avg       0.73      0.74      0.71        42
```

References