TRANSFER LEARNING FOR CATEGORIES AND LANDMARKS CLASSIFICATION

Group #12
Samarth Saxena saxenasa@usc.edu
Revati Pawar rpawar@usc.edu

I. Abstract

In this project, we explored the use of transfer learning to classify images of monuments from around the world using the EfficientNetB0 and VGG16 architectures. Our primary goal was to develop efficient and accurate models for landmark and category classification. We formulated several hypotheses and experimented with different approaches, including data augmentation and normalization techniques, freezing and unfreezing layers, adjusting learning rates, and incorporating dropout layers. We achieved an F1-Score of 0.98 and 0.91 for categories and landmarks classification, respectively, using the EfficientNetB0 model with a Sequential learning approach. Future work includes exploring different architectures and regularization techniques and investigating the underperformance of some approaches. Overall, our results demonstrate the effectiveness of transfer learning and its potential for improving the accuracy and efficiency of deep learning models.

II. Approaches and Hypotheses

We formulated hypotheses to guide our experimentation and tested various approaches to train our deep learning models for landmark and category classification. We explored different techniques such as data augmentation, normalization, freezing and unfreezing layers, adjusting learning rates, and incorporating dropout layers. We also hypothesized that batch sizes and pre-trained CNN architectures like EfficientNetB0 could impact the accuracy of our models. By keeping these hypotheses in mind, we designed and experimented with five different model architectures, including parallel and sequential learning approaches.

1. Parallel Learning (Multi-output model) : Compiling the model just once with 2 different output layers for each classification task.
2. Sequential Learning : Compiling and training the model for **Categories Task first** and then doing the same for Landmarks Task using the same model (same weights) and swapping out the output layer.
3. Sequential Learning : Compiling and training the model for **Landmarks Task first** and then doing the same for Landmarks Task using the same model (same weights) and swapping out the output layer.

4. Sequential Learning : Compiling and training the model for **Categories Task first** and using the output of the first model as **an additional feature for the Landmarks Task.**
5. Sequential Learning : Compiling and training the model for **Landmarks Task first** and using the output of the first model as **an additional feature for the Categories Task.**

III. Research and Implementation

This section highlights the key decisions made during the development of the model. The research and development stage of the project involved testing and validating the hypotheses, fine-tuning the proposed approaches, and iterating over multiple model iterations.

1. Loading and Splitting Data

We opted to store the pixel values in a 4D tuple in the shape of (X, 255, 255, 3), with the first dimension representing the number of images, and the remaining dimensions representing the height, width, and color channels of each image. To ensure that every class is represented in each of the data splits created, we implemented stratify manually. This was crucial for our multiclass problem as not doing so would mislead the model and affect its performance.

2. Data Preprocessing

a. Data Normalization: We considered whether normalizing image pixels would be beneficial for our model's performance. However, since we were using EfficientNet, we decided not to use normalization in our approach.

b. Data Augmentation: Data was augmented to account for the small dataset to improve performance of the models.

3. Model Architecture and Model Training

a. **Transfer Learning:** We decided to use EfficientNetB0 instead of VGG16 because it has a faster training time and can achieve comparable accuracy on various tasks. On the other hand, VGG16 is an older architecture that has been widely used and tested in various computer vision tasks. It has a more complex architecture and more parameters than EfficientNetB0, making it a better choice for larger datasets and achieving higher accuracy at the cost of longer training times.

b. **Batch Size:** We found that a batch size of 128 performed better than using a batch size of 32 or 64. This seemed to strike a perfect balance between performance and computation time.

c. **Dropout Layers:** We added dropouts to prevent overfitting since these pretrained models can easily overfit on the training data due to their large size and previously learned weights. When training for categories first, we used lots of dropout layers as it was an easier task and easier for the model to overfit. But when training for landmarks first or training them at the same time, we reduced the dropout rate or the number of dropout layers because the model had to learn as many patterns as possible for the landmarks set, and it was not overfitting for a relatively long number of epochs.

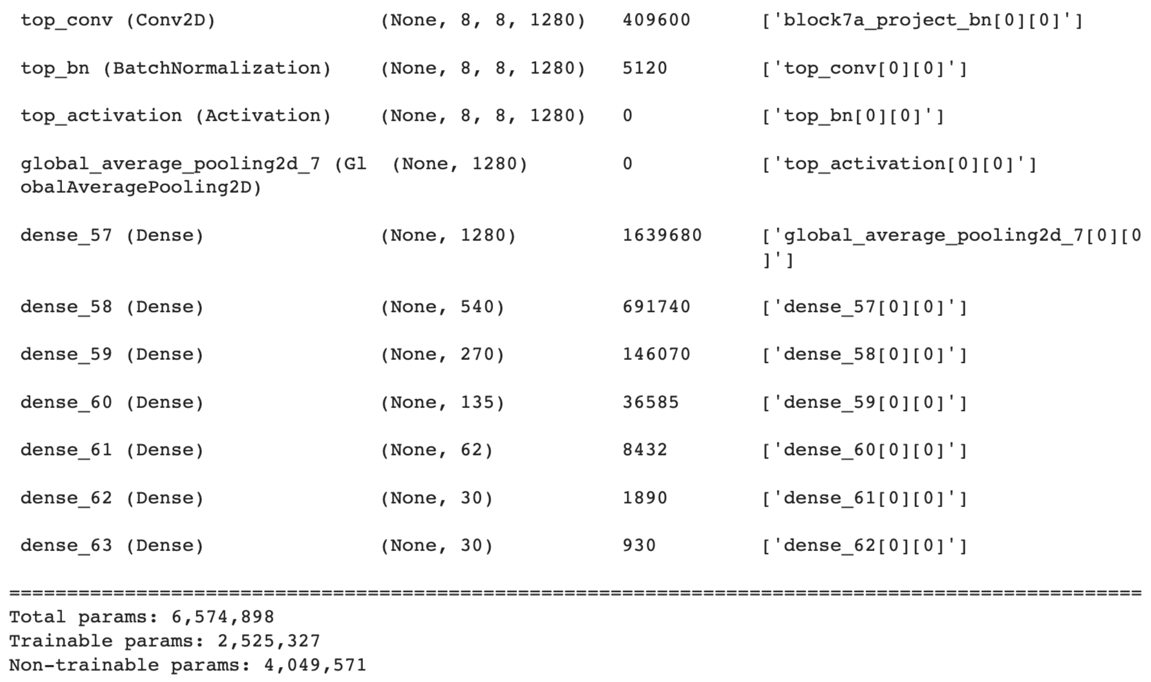
d. **Activation Function:** We used softmax as it produces probabilities relative to the other classes. For example, if there are 3 classes, the relative probabilities might be 0.6, 0.3, and 0.1 (which add up to 1), while sigmoid doesn't do that. It just gives an output between 0 to 1 for each class.

e. **Freezing Layers:** We froze all the layers from EfficientNet and added a few dense layers on top so that the model learns some task-specific patterns from the images.

f. **Learning Rate:** We iterated over multiple learning rate values and found the optimum value to be 0.003. A high learning rate value can give a lot of spikes in the graphs, and a low learning rate value will produce a smooth graph but might never get to the desired results. Choosing the right learning rate can be tricky, and if a high value is chosen, it might never converge on the global minimum, while if a small value is chosen, it might get stuck in the local minimum.

4. Model Selection

It was established that training for landmarks task first was better because there were fewer images per class in the training set. It was better for the model to emphasize more on the landmarks task. Since there were a lot of images per class for the categories task and on top of that, there were fewer numbers of classes, this was an easier task (in terms of predictions) compared to the landmarks task. So upon research, these inferences were made that it is much better to train for the landmarks dataset first, and then we could use the same model with the same weights for the categories task.



```
 top_conv (Conv2D)              (None, 8, 8, 1280)   409600      ['block7a_project_bn[0][0]']

 top_bn (BatchNormalization)    (None, 8, 8, 1280)   5120        ['top_conv[0][0]']

 top_activation (Activation)    (None, 8, 8, 1280)   0           ['top_bn[0][0]']

 global_average_pooling2d_7 (Gl  (None, 1280)        0           ['top_activation[0][0]']
 obalAveragePooling2D)

 dense_57 (Dense)               (None, 1280)         1639680     ['global_average_pooling2d_7[0][0
                                                                  ]']

 dense_58 (Dense)               (None, 540)          691740      ['dense_57[0][0]']

 dense_59 (Dense)               (None, 270)          146070      ['dense_58[0][0]']

 dense_60 (Dense)               (None, 135)          36585       ['dense_59[0][0]']

 dense_61 (Dense)               (None, 62)           8432        ['dense_60[0][0]']

 dense_62 (Dense)               (None, 30)           1890        ['dense_61[0][0]']

 dense_63 (Dense)               (None, 30)           930         ['dense_62[0][0]']

==================================================================================================
Total params: 6,574,898
Trainable params: 2,525,327
Non-trainable params: 4,049,571
```

Figure 1: Model Summary

IV. Results

We implemented transfer learning for multiclass classification of categories and landmarks using the EfficientNetB0 pre-trained model. We experimented with multiple approaches and achieved the best F1-Score of 1.00 and 0.97 respectively on test sets using the Sequential approach of first training the model for Landmarks task and then using the same model (only swapping out the output layer) for the Categories task.

Order of Classification	**F1 Score (Categories)**	**F1 Score (Landmarks)**
Category > Landmarks	0.98	0.48
Landmarks > Category	**1.00**	**0.97**
Category > Landmarks (Using Category as a feature)	0.98	0.46
Landmarks > Category (Using Landmarks as a feature)	0.98	0.79
Multi-Output Model	0.97	0.81

Table 1: Performance comparison among different approaches

To evaluate the performance of our deep learning models, we plotted the loss and accuracy for each epoch during training and validation as seen in Figure 2 and 3.

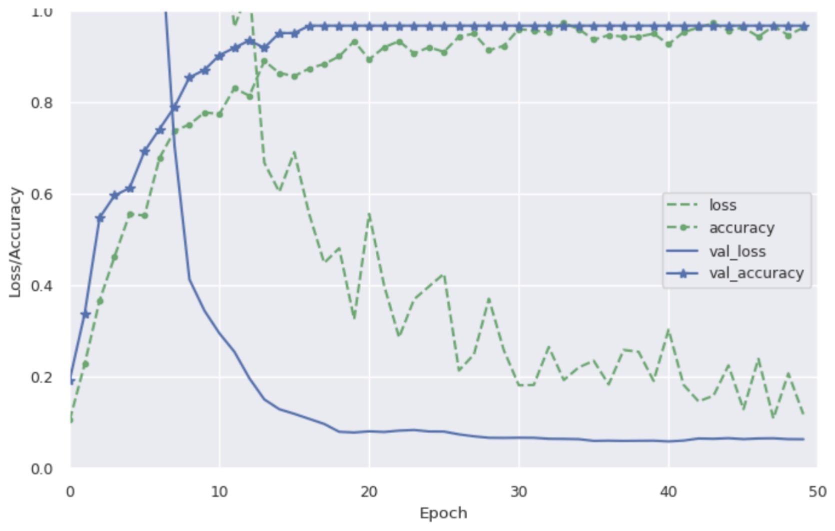


Figure 2: Model Performance - Landmarks

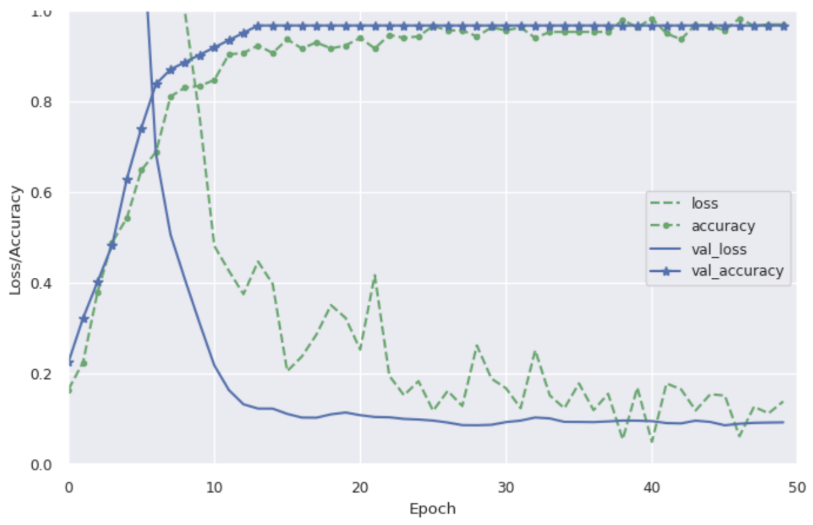


Figure 3: Model Performance - Categories

IV. Conclusion

This project utilized transfer learning with EfficientNetB0 and VGG16 architectures to classify images of Monuments around the World. The EfficientNetB0 model was found to be more suitable, achieving F1-Scores of 1.00 and 0.97 for Categories and Landmarks Classification respectively. Future work includes exploring other architectures, regularization techniques, and unsupervised learning methods to improve performance and data quality. Overall, this project highlights the potential of transfer learning in image classification tasks.