Analysis of Different Transfer Learning Approaches on Natural Scenes Classification

Members: Changchieh Liu, Jiazhou Liu

Abstract:

In this work, we examined the different transfer techniques in transfer learning using various convolutional neural networks (CNN) available in the TensorFlow framework. Using the pre-trained ImageNet models, we performed many experiments using feature extract, finetune, and a hybrid transfer approach and analyzed the results to identify the method that offers the best performance using the minimal amount of training time. Our experiments will be able to offer machine learning practitioners advice on the ideal approach to perform transfer learning that achieves high accuracy within a reasonable amount of time. The results of the experiment indicate that the hybrid transfer approach offers a good balance between model performance and time when trained and evaluated on the Intel Natural Scene Classification dataset. Furthermore, by combining transfer learning with ensemble learning we were able to obtain a classifier with improved performance compared to each individual network at 95% testing accuracy. Last but not least, our experiments illustrated that in the hybrid approach, the transition from feature extraction to finetune generally lowers the accuracy but quickly recovers and improves the performance of the network towards the end of training.

Problem Motivation:

The motivation behind our project is that training a state-of-the-art CNN for image classification tasks has many challenges. The two main challenges are that it requires a significant amount of annotated data and a large amount of time to reach a desirable performance. However, by leveraging transfer learning, it is possible to achieve high accuracy with a smaller dataset within a reasonable amount of training time. Transfer learning is a popular machine learning technique that allows one to reuse a network originally trained for a source task and transfer its knowledge to perform another target task. Nevertheless, there are many different ways to perform transfer learning and identifying the best method can help make the training process more manageable by finding the right balance between performance and training time which may be crucial to business requirements. There are two primary transfer approaches to take advantage of pre-trained models: feature extract and finetune. Feature extraction uses the feature representations learned by another network and replaces the last layer with a new one and only trains that new layer to repurpose the network for the intended target task. On the other hand, finetuning also replaces the last layer, however, it involves unfreezing the hidden layers of the source network and updating the weights among those layers to obtain a new network for the target task. By performing finetuning, we allow the network to make adjustments to the weights across the hidden layers and learn new feature representations that are more relevant to the target dataset. Nevertheless, in this project, we considered a third approach that combines both methods and conducted several experiments and analyzed their results to offer comprehensive comparisons between the different approaches on pre-trained ImageNet CNNs.

Background:

The transfer learning experiments are conducted using the Intel Scene Classification dataset that contains images that fall into six different categories: buildings, forest, glacier, mountain, sea, and street. The dataset is available on Kaggle and was published by Intel as part of its image classification challenge [1]. In the training set, there are around 14,000 images; whereas in the testing set, there are a total of 3,000 images. The six classes are more or less equally distributed with mountain having slightly more samples compared to the other. Since this is a public challenge, we were able to find the first-place solution [2] which allows us to establish a benchmark for how well the models performed in our experiments. Some of the notable characteristics of the first-place solution is that it utilized transfer learning on ResNet50 and achieved test accuracy of approximately 96%.

Technical Challenges:

Hardware:

Throughout the project, there are two main challenges that were encountered. The first is a hardware related challenge where we only have access to a limited number of resources on the cloud platform due to free trial accounts. We were only able to set-up a K80 compute instance on Azure Machine Learning platform. However, the hardware is incapable of training the CNNs within a reasonable amount of time due to large amounts of computation and memory requirements for each iteration. To overcome this hardware resource limitation and performance issue, we decided to change the GPU and use V100 on NYU High Performance Computing (HPC) for training our networks. This significantly improved the training time for each epoch across the different networks allowing us to obtain the results within a reasonable amount of time.

Data:

The second challenge that we faced is data-related. Throughout the dataset, we observed that there are some classes of natural scene images that overlap with one another. This is a task specific problem considering the fact that some natural scenes are a subset of other natural scenes. For example, to differentiate between mountain and glacier proved to be a challenging task and may require a human expert to distinguish between the two as glaciers develop in mountainous regions. Nevertheless, it is an unavoidable issue that may impact the performance of the networks.

Approach:

Selection of Pre-trained Models and Source Task:

The first part of our approach is the selection of the pre-trained models and source weights for transfer learning. To ensure the effects of the different transfer learning approaches are independent of the CNN architectures, we selected five different networks available within the TensorFlow framework: Xception, InceptionV3, InceptionResNetV2, VGG16, and ResNet50. Furthermore, to maintain consistency of the source knowledge transferred to the target networks, we used the ImageNet weights across all the models. This is because ImageNet is a publicly available dataset that is widely used for image classification competitions, so various CNNs are trained on the dataset with weights available for the different networks that we selected for the experiment. In addition, the dataset is general and large enough which we believe is the best candidate for using it as a source for our intended target task to classify natural scenes.

Transfer Approaches:

The second part of our approach is to determine the different approaches for the experiments. For each of the selected CNNs, we trained the models using the following four methods: base model, feature extract, finetune, and hybrid. The base model serves as a baseline benchmark for the different transfer learning approaches where we trained the network from scratch with random weights initialization. The feature extract method is where we freeze all the layers with ImageNet weights in the pre-trained model and train only the last softmax layer. The finetune method initializes the network with ImageNet weights but makes minor updates to all the hidden layers in the network. Last but not least, the hybrid method performs feature extraction for the first half of training, followed by finetune in the remaining half of training.

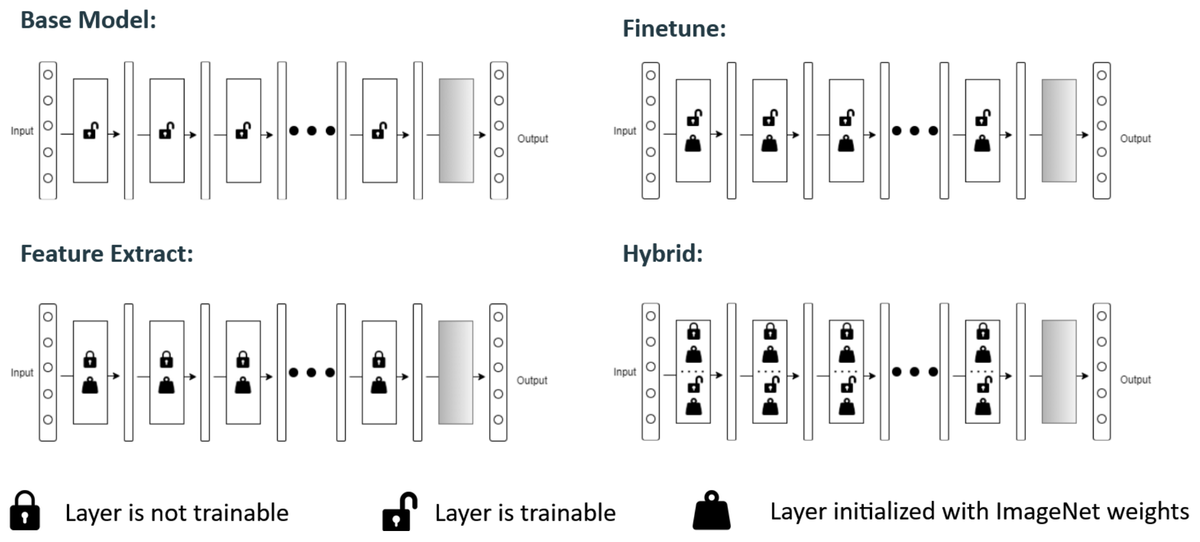


Figure 1: Diagrams of the four different methods in the experiments

Implementation Details:

Our model implementation is based on the Tensorflow machine learning framework. In addition, we also utilized various libraries such as Keras, Matplotlib, numpy and so on. In terms of underlying hardware, we trained and tested our models on the NYU HPC cluster with Nvidia V100 GPUs. Regarding the hyper parameters, we chose 64 as the batch size, Adam as the optimizer, and sparse categorical cross entropy as the loss function; all the models are trained for 100 epochs, and we used 0.001 learning rate for the last dense layer and 0.0001 learning rate for fine tuning the hidden layers.

Experiment Design Flow:

In order to evaluate the results of our experiments, we used line charts to characterize the fluctuation of training and validation accuracy of our models during the training and we also created tables to compare the performances between the different CNN architectures. Performances are measured by both validation accuracy and training time. We decided to add training time as a metric because accuracy itself isn't good enough to evaluate the goodness of a model. By adding training time, we can calculate TTA for each model, which can help us to make decisions when choosing appropriate models to begin with.

Result Observation/Analysis:

In this section, we will reveal the best performing model for each transfer learning approach. For Feature Extraction, the best performing model is ResNet50 and for both Fine Tuning and Hybrid, the best performing model is Xception.

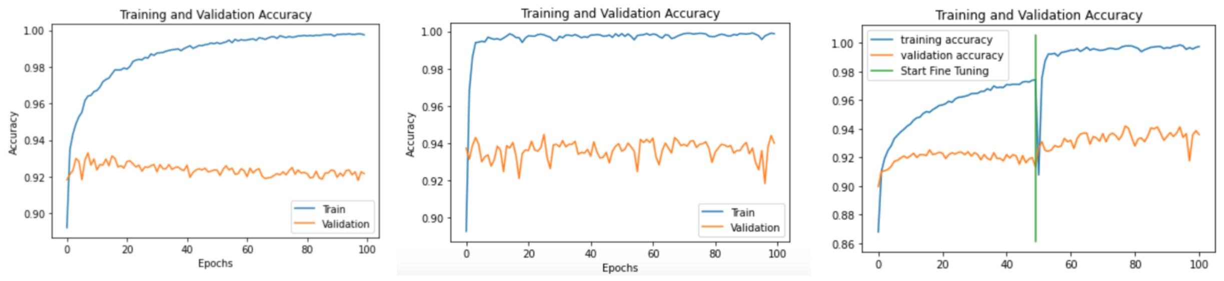
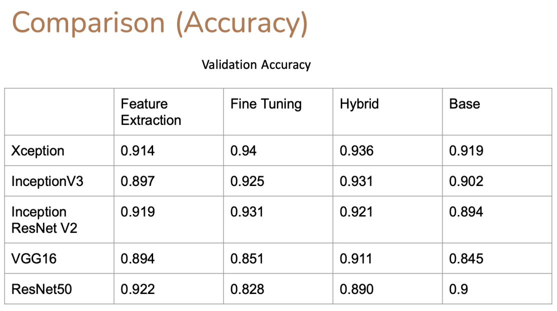


Figure 2: The line charts of all three models (from left to right): ResNet50 - Feature Extraction, Xception - Fine Tuning, Xception - Hybrid

For the Hybrid approach, It's worth noticing that when we switch from Feature Extraction to Fine Tuning, the accuracy drops significantly at first, but recovers immediately after. Our intuition is that when switching from feature extraction to fine tuning, since the inner layers are suddenly unfrozen, the models tend to forget what they learned initially, but as the training continues, they can relearn those knowledges and even achieve a better performance.

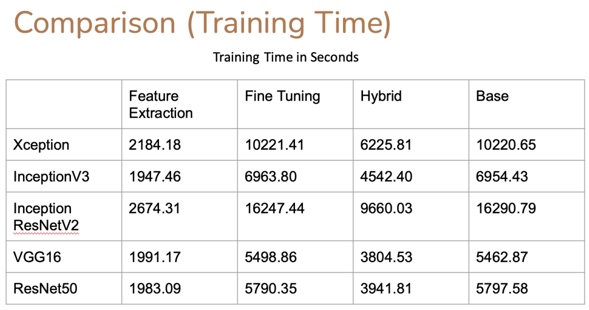
In order to compare the validation accuracy as well as training time for all the models, we organize the data from our experiments into tables below:



Comparison (Accuracy)

Validation Accuracy

	Feature Extraction	Fine Tuning	Hybrid	Base
Xception	0.914	0.94	0.936	0.919
InceptionV3	0.897	0.925	0.931	0.902
Inception ResNet V2	0.919	0.931	0.921	0.894
VGG16	0.894	0.851	0.911	0.845
ResNet50	0.922	0.828	0.890	0.9



Comparison (Training Time)

Training Time in Seconds

	Feature Extraction	Fine Tuning	Hybrid	Base
Xception	2184.18	10221.41	6225.81	10220.65
InceptionV3	1947.46	6963.80	4542.40	6954.43
Inception ResNetV2	2674.31	16247.44	9660.03	16290.79
VGG16	1991.17	5498.86	3804.53	5462.87
ResNet50	1983.09	5790.35	3941.81	5797.58

Figure 3: Comparisons of accuracy and training time of different CNN architectures

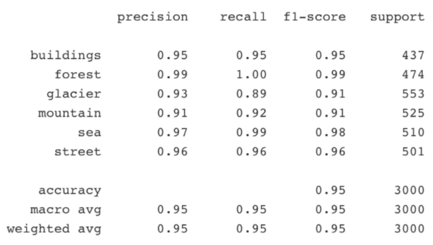
From the first table, it's obvious that for each architecture, there exists at least one transfer learning approach that provides better performance than training from scratch, which demonstrates the benefit of using transfer learning. Usually speaking, Fine tuning and Hybrid provide best performance rather than Feature Extraction, except for ResNet50 in our case; which one is strictly better can depend on different CNNs. From the second table, we can observe that Feature Extraction requires the least training time since we only train the last dense layer; meanwhile, Fine Tuning and Base approaches require the most training time as all the layers need to be trained for the entire training stage. Hybrid's training time is in the middle as it uses both Feature Extraction and Fine Tuning during training.

Conclusion:

After observing the experiment results, we found that both Fine Tuning and Hybrid methods can provide outstanding performance in terms of prediction accuracies. In general, hybrid is considered as the overall best approach, as it requires significantly shorter training time, to achieve a prediction accuracy similar to Fine Tuning.

Further Improvement:

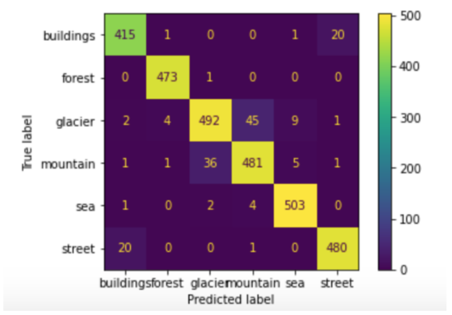
In order to further improve the prediction accuracy, we decide to create an ensemble model, which makes predictions by considering the predictions from all the models above, and following the majority decisions. With this added model, we further improve the validation accuracy to 95%. We provide its performance metrics below as reference:



```
              precision    recall  f1-score   support

   buildings       0.95      0.95      0.95       437
      forest       0.99      1.00      0.99       474
     glacier       0.93      0.89      0.91       553
    mountain       0.91      0.92      0.91       525
         sea       0.97      0.99      0.98       510
      street       0.96      0.96      0.96       501

    accuracy                           0.95      3000
   macro avg       0.95      0.95      0.95      3000
weighted avg       0.95      0.95      0.95      3000
```



Reference:

[1]: https://www.kaggle.com/puneet6060/intel-image-classification

[2]:https://towardsdatascience.com/1st-place-solution-for-intel-scene-classification-challenge-c95cf941f8ed