Features Transferability in VGG11 with CIFAR10

By: Tey Shiwei A0112101M and Chua Chin Siang A0112089J

Table of Content

Introduction

Deep Neural Network (DNN) is the state-of-the-art technology for image classification tasks and promising results have been achieved. With the advancement in hardware, image classification can now be easily done with all the well-built DNN models published online. However, training DNN from scratch is extremely time consuming and very often, the databases readily available for the training of such models are not extensive enough to create a model that meets the stringent accuracy and robustness requirements for practical applications. As a result, the concept of Transfer Learning (TL) was introduced. TL provides users with the ability to leverage on existing pre-trained models and extend it to the specific tasks at hand. Typically, these pre-trained models have been fed with large datasets of test examples from another source, allowing basic recognition abilities to be established in such models. The pre-trained models can then be utilized and applied on another set of related problems with minimal fine tunings to be done.

Similar to human cognitive and brain development, these models employ a hierarchy of layers whereby each layer considers information from previous layers and passes on its output to the subsequent layer during its learning process. As such, by freezing different layers, one can control the level of learnability and data transferability of the pre-trained model.

Literature review

Based on a current work by Andrew [1], a ResNet50 model trained from Imagenet architecture correctly predicted the classes of CIFAR10 with only minimal modifications done on the final layer. His model managed to obtain a classification accuracy of 98% with only 5 epochs of re-training. This example shows that the features of CIFAR10 have already been learnt by the network from Imagenet due to a certain level of similarity between the training and test datasets. The key message here is that from a model trained with a huge dataset (14mil), TL to a 60k dataset works pretty well.

Paper review

This project is inspired by Yosinki's paper: **How Transferable are features in deep neural networks**, where TL was performed from a model trained with a selected pool of 500 classes from Imagenet to the remaining 500 classes from Imagenet. The objective of his work was to investigate the transferability of a DNN based on the number of frozen layers and the similarity between the base and target tasks. The author highlighted that the first few layers from the inputs of image classification DNN constitute generic representations while layers nearer to the output constitute task-specific representations. It turns out that DNN has a transition phase in between the layers which has yet to be determined with a consistent method.

Project Contributions

Based on available online sources, top performing TL networks [4] such as VGG, Inception and ResNet for performing image recognition tasks on ImageNet are conveniently accessible. For this project, we will create a DNN model with randomized weights and train the Base model with a portion of the dataset from CIFAR10. TL onto the Target model will be done on the remaining subset of CIFAR10 dataset which were not included in the training process. Details of the model setup will be discussed in the next section.

Firstly, we are going to investigate the generic to specific features detailed by Yosinki by replicating the environment onto CIFAR10. Our project also aims to monitor the training and validation accuracy trends during TL to further expand findings that were not mentioned by the author. Lastly, custom experiments will be carried out to investigate the effects of data similarity on classification accuracy.

Experimental Setup

Coding Framework

The author used the Caffe framework to carry out his experiments while we use Pytorch to carry out our own experiments.

Dataset selection

ImageNet is a huge collection of human annotated images and it has been commonly used for various TL tasks. We have chosen CIFAR10 and CIFAR100 as our base training dataset and TL target dataset respectively. Most of the experimental tasks were on CIFAR10 because it has more training data (6k per class) than CIFAR100 (600 per class). In addition, CIFAR10 consists of only 10 classes which is relatively easier for us to categorize the data by degree of similarity.

In this project, CIFAR10 is clustered into five different groups:

Group Names	**CIFAR10 classes***
Class A	cat, deer, dog, horse, ship
Class B	airplane, automobile, bird, frog, truck
Animal 1 (A1)	bird, cat, deer
Animal 2 (A2)	frog, horse, dog
Vehicles (V)	airplane, automobile, truck, ship

Table 1: CIFAR10 classes by different group names

**It is to be noted that classes used for Base model training will not be used in Target model training.*

Data Pre-processing

- Random Cropping, Flipping and Rotating of the training images
- 0.2 for train-to-test ratio

Model Selection and Setup

VGG11 with batch normalization was chosen based on 3 criteria:

1. Good classification accuracy on CIFAR100 [5]
2. Optimal number of hidden layers. With 11 hidden layers in series, it is sufficient to generate defined generic and specific layers, yet not too many such that the investigation on the effects of layer discretization can still be easily performed

3. 28.5mil parameters, short training time with GPU

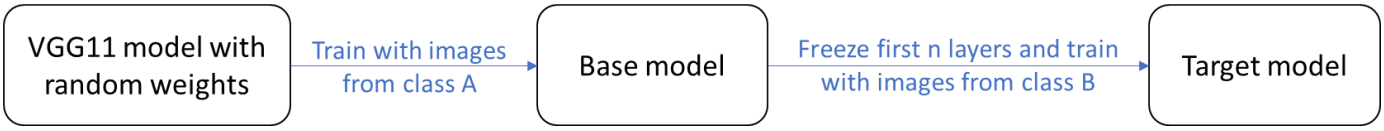

Figure 1: Transfer learning (TL) process flow for our project

VGG11 Base model setup:

- Replacing the last fully connected layer with a 10 classes layer
- Initial weights of the whole network are randomized
- 200 epochs will be run on each training phase
- Model with the best validation score during each training phase will be picked

VGG11 Target model setup:

- Frozen layer: Initialize the weights from VGG11 Base model
- Non-frozen layers: Initialize the weights randomly
- 200 epochs will be run on each training phase with a different set of training data
- Model with the best validation score during each re-training phase will be picked

Evaluation Metric

The evaluation of our Target model's performance was carried out based on the classification accuracy.

Results and Discussion

To validate the model's performance, we tested VGG11 using CIFAR10 with different output channels. Figure 1 shows that the performance between the original VGG (100 output channels) and modified VGG (10 output channels) are comparable. Hence, VGG11 with 10 output channels will be used for all the tests in this project.

The number of frozen layers (n), is counted starting from the input layer. For example, when n=5, the first 5 hidden layers from the input are frozen.

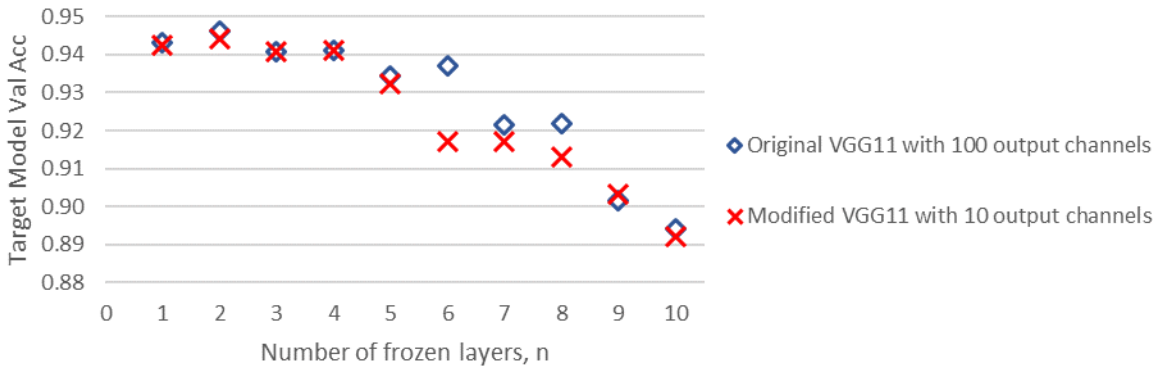

Figure 1. A to B Transfer Learning on VGG11 architecture with different output channels

Our findings on Transfer Learnings

1. As shown in Figure 2 below, the Base model requires at least 120 epochs to achieve 0.88 accuracy, whilst the validation accuracy during Target model training reaches 0.88 with only 5 epochs of training and attains a high 0.92 accuracy at the end of training. This shows that transfer learning is effective and lesser training is required for a new classification task.

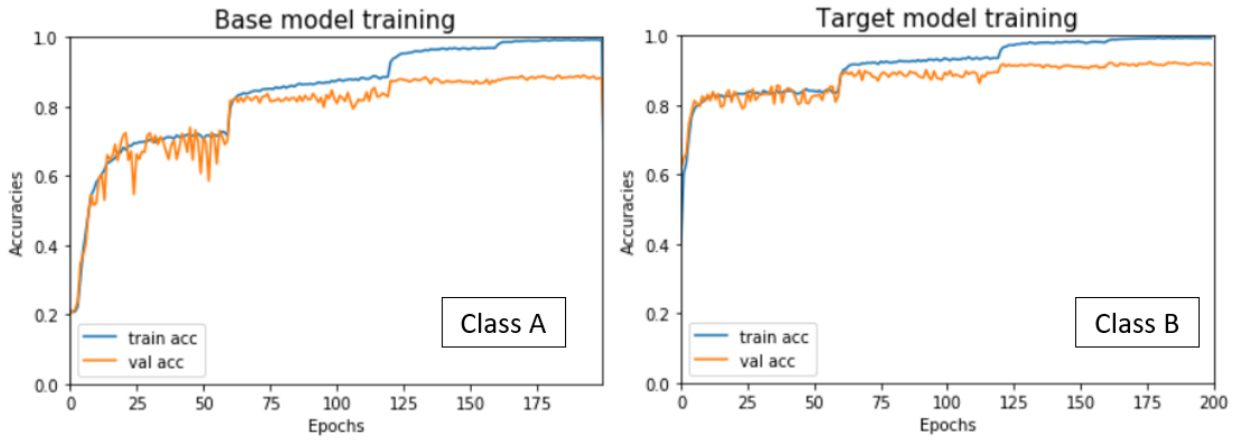

Figure 2. Left: training accuracy (train acc) and validation accuracy (val acc) during Base model training, Right: training accuracy and validation accuracy during Target model training.

2. In this section, we investigated the correlation between n and both training and validation accuracies of the Target model.

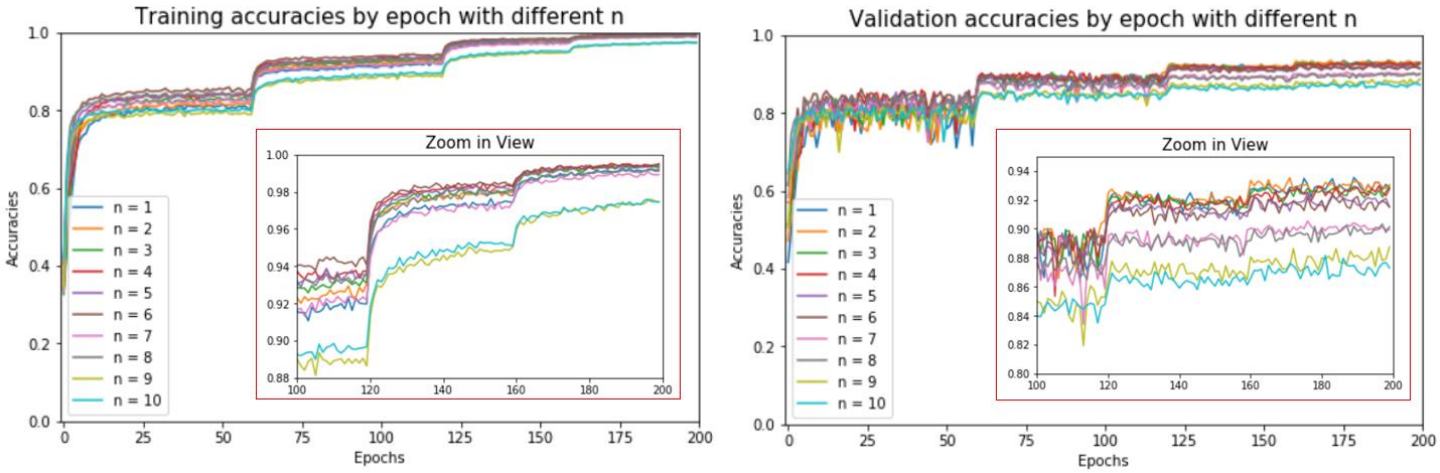

Figure 3. Left: training accuracies by epoch with different n, Right: validation accuracies by epoch with different n

According to Figure 3 (Left), the training accuracies are relatively lower for n=9 and n=10, For n=1 to n=8, there are no correlations between n and training accuracies. Results in Figure 3 (Right) shows that a lower n yields higher validation accuracies. Lower n implies that most of the hidden layers in the Target model are re-trained. From here, we could segregate the hidden layers of VGG11 into 3 categories based on their level of accuracies:

- All layers before the 6^{th} layer are generic layers (Val acc >0.91 even after freezing)
- 7^{th} and 8^{th} layer are transition layers (Val acc = 0.90 after freezing)
- 9^{th} and 10^{th} layer are specific layers (Freezing them lowers the Val acc)

Results Comparison with Yosinki's work

In order to duplicate the author's work, CIFAR10 was grouped into two equal major classes: Class A and Class B as shown in Table 1. With a baseline performance of 94.76% on Class B, two different sets of experiments were carried out, A to B and B to B. Figure 4 illustrates our results on CIFAR10, in which the validation accuracy drops as n increases for both A to B experiment. As mentioned in the previous section, this is due to the specific layers (n>6) in VGG11 being frozen.

Whilst our A to B results were in good agreement with literature, the results of our B to B TL were not consistent with the author's findings. From Figure 5, the Val Acc of the author's B to B initially dropped till n=5 due to fragile co-adapted features [6], but recovered as n further increased until the last layer. However, our Val Acc has a sudden drop when n=10. This discrepancy might be due to the high learning rates employed during re-training while the model parameters' weights are already close to the optimal.

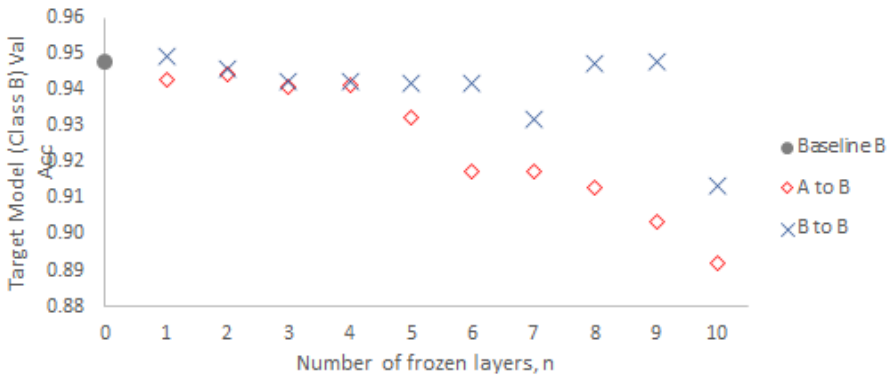

Figure 4. Our results on CIFAR10

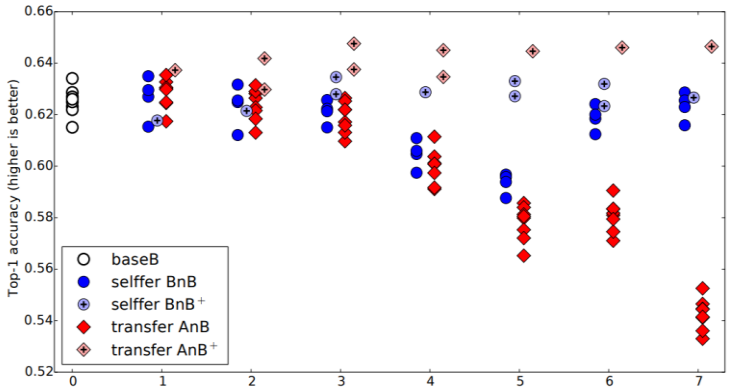

Figure 5. Yosinki's results on ImageNet

Data Similarity effects

In this section, we investigated the effects of data similarity on the validation accuracies of the Target model. CIFAR10 data were split into Animal1 (A1), Animal2 (A2) and Vehicle (V) as shown in Table 1.

First, we trained the Base model using A1 images, then we performed a A1 to A2 TL. The result holds the same A to B assumptions. When n=1, this model is able to achieve the same accuracy as the Base model. This could be due to the similarity of the datasets as animals. Comparing the respective results of V to A2, A1 to V and V to A1 as shown in Figure 6, it was observed that TL in similar contexts helps reduce the drop in performance when n increases.

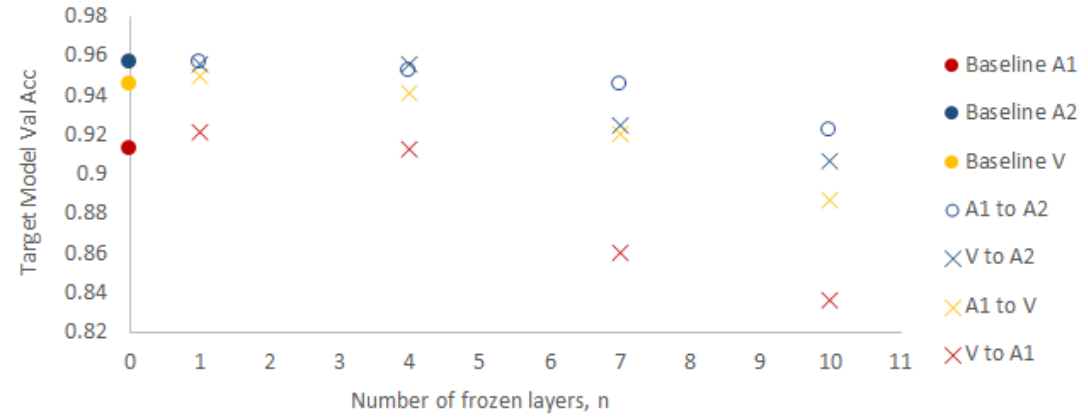

Figure 6. Target model accuracy with the changes of frozen layers count

Conclusion

The number of classes in this experiment might be small but the general trend was observed in our results. Firstly, TL's performance is comparable even after changing the final FC layer. By freezing the generic layers of VGG11, TL works very well even on a different dataset, with the exception when the specific layers (n=9,10) are frozen. The last experiment was carried out to show that TL performs optimally when the Base and Target models were trained with similar datasets.

The biggest limitation of TL is that tuning might take quite some time as there are no auto tuning methods available. We can foresee that TL will be very commonly used in the future and that there would be a robust model created, capable of classifying most of the available images with minimal fine tuning done.

References

[1] Transfer Learning Using ResNet50 and CIFAR-10: https://medium.com/@andrew.dabydeen/transfer-learning-using-resnet50-and-cifar-10-6242ed4b4245

[2] Transfer Learning in Tensorflow (VGG19 on CIFAR-10)

a. Part 1: https://towardsdatascience.com/transfer-learning-in-tensorflow-9e4f7eae3bb4

b. Part 2: https://towardsdatascience.com/transfer-learning-in-tensorflow-5d2b6ad495cb
[3] Data similarity and sample size effects: https://kharshit.github.io/blog/2018/08/10/transfer-learning
[4] Transfer Learning in Keras with Computer Vision Models: https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/
[5] Model selection based on the performance table from https://github.com/weiaicunzai/pytorch-cifar100
[6] How transferable are features in deep neural networks? http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf?fbclid=IwAR2clOYITEdWHBWT-iqvgIDYpKet3VH9MYWZXBnp3G3NmgAIv0C6ay-9iyo
[7] Transfer learning source code: https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
[8] CIFAR datasets: https://www.cs.toronto.edu/~kriz/cifar.html

Appendix

network	params	top1 err	top5 err	memory
mobilenet	3.3M	34.02	10.56	0.69GB
mobilenetv2	2.36M	31.92	9.02	0.84GB
squeezenet	0.78M	30.59	8.36	0.73GB
shufflenet	1.0M	29.94	8.35	0.84GB
shufflenetv2	1.3M	30.49	8.49	0.78GB
vgg11_bn	28.5M	31.36	11.85	1.98GB
vgg13_bn	28.7M	28	9.71	1.98GB
vgg16_bn	34.0M	27.07	8.84	2.03GB
vgg19_bn	39.0M	27.77	8.84	2.08GB
resnet18	11.2M	24.39	6.95	3.02GB
resnet34	21.3M	23.24	6.63	3.22GB
resnet50	23.7M	22.61	6.04	3.40GB
resnet101	42.7M	22.22	5.61	3.72GB
resnet152	58.3M	22.31	5.81	4.36GB
preactresnet18	11.3M	27.08	8.53	3.09GB

preactresnet34	21.5M	24.79	7.68	3.23GB
preactresnet50	23.9M	25.73	8.15	3.42GB
preactresnet101	42.9M	24.84	7.83	3.81GB
preactresnet152	58.6M	22.71	6.62	4.20GB
resnext50	14.8M	22.23	6	1.91GB
resnext101	25.3M	22.22	5.99	2.63GB
resnext152	33.3M	22.4	5.58	3.18GB
attention59	55.7M	33.75	12.9	3.47GB
attention92	102.5M	36.52	11.47	3.88GB
densenet121	7.0M	22.99	6.45	1.28GB
densenet161	26M	21.56	6.04	2.10GB
densenet201	18M	21.46	5.9	2.10GB
googlenet	6.2M	21.97	5.94	2.05GB
inceptionv3	22.3M	22.81	6.39	2.26GB
inceptionv4	41.3M	24.14	6.9	4.11GB
inceptionresnetv2	65.4M	27.51	9.11	4.14GB
xception	21.0M	25.07	7.32	1.67GB
seresnet18	11.4M	23.56	6.68	3.12GB
seresnet34	21.6M	22.07	6.12	3.29GB
seresnet50	26.5M	21.42	5.58	3.70GB
seresnet101	47.7M	20.98	5.41	4.39GB
seresnet152	66.2M	20.66	5.19	5.95GB
nasnet	5.2M	22.71	5.91	3.69GB

Table 2: Performances and memory requirements for different DNN models on CIFAR100