Image Classification using Transfer Learning

Bhanu Renukuntla, Saket Sharma, Swaroop Gadiyaram, Venkatesh Elango, Vikas Sakaray
Department of Electrical and Computer Engineering
University of California, San Diego

Abstract

This work deals with application of *Transfer Learning* for classification of images from Caltech-256 - a dataset similar to ImageNet dataset, using a pre-trained Convolutional Neural Network (CNN) model *VGG16*, using *Keras* library in Python. We also evaluate the performance of *Transfer Learning* in classifying images from the Urban Tribes dataset which not similar to ImageNet dataset. We study the performance of the CNN as a function of number of training examples used on both datasets by replacing the output softmax layer of the *VGG16*, pre-trained on 1000 classes of ImageNet, by a new Softmax layer to predict the classes of our dataset. This softmax layer is trained using much fewer training examples and its performance is observed. For Caltech-256, classification accuracies ranging from 39% (training with 2 images per class) to 76.74% (training with 60 images per class) were achieved. On the Urban Tribes dataset, the accuracy varied from 30% (training with 2 images per class) to 56% (training with 64 images per class). We finally try classifying Caltech-256 dataset by adding a softmax layer for our classes over the softmax of *VGG16* by employing a *temperature* parameter. We observe that this approach prevents overfitting and yields better results than vanilla transfer learning with fewer examples. We get an accuracy of 46% on test set using 2 images per class for training (an 18% increase) and comparable results with more images per class for training.

1 Introduction

Convolutional Neural Networks (CNNs) are very similar to ordinary Neural Networks; they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other and they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks in Homework 2 still apply. The differentiating feature of CNNs is that ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the feedforward more efficient to implement and vastly reduce the amount of parameters in the network.

1.1 Transfer Learning

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size and is computationally expensive. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the trained ConvNet either as an initialization or a fixed feature extractor for the task of interest.

In our problem, we take *VGG16* (Section 1.2), a CNN trained on ImageNet dataset with

1000 classes, replace its final softmax layer with a new softmax layer with output nodes equal to the number of classes in our datasets. And then just train this final layer in Keras using a few number of training examples.

1.1.1 Temperature based softmax regression

We also try a different type of transfer learning wherein we do not remove the output softmax layer from the *VGG16* Convnet and instead insert temperature into the calculation of the last softmax layer of *VGG16*. We add another softmax layer on top of it, with appropriate number of outputs, and then train that layer. The slightly modified softmax function incorporating a *temperature* term as follows:

$$y_i = \frac{exp(a_i/T)}{\sum_j exp(a_j/T)} \tag{1}$$

where T is the *temperature* parameter. Using a higher value for T produces a softer probability distribution over classes.

1.2 The Network - VGG-16

During training, the input to *VGG16* ConvNet is a fixed-size 224x224 RGB image. The image is passed through a stack of convolutional (conv.) layers, that use filters with a very small receptive field: 3 by 3 (which is the smallest size to capture the notion of left/right, up/down, center). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 by 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2x2 pixel window, with stride 2.

A stack of convolutional layers of varying depth is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity. The network doesn't contain Local Response Normalisation (LRN) normalisation (Krizhevsky et al., 2012): such normalisation does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. Where applicable, the parameters for the LRN layer are those of (Krizhevsky et al., 2012). Such a pre-trained network (on ImageNet dataset) has been imported from *Keras* library for our transfer learning task.

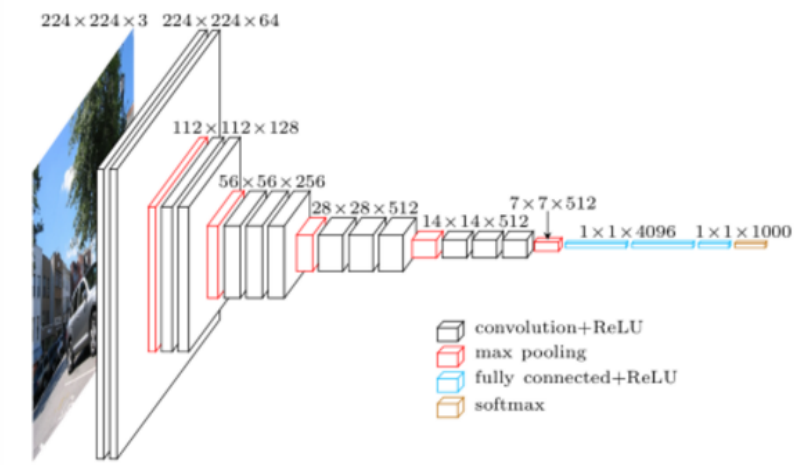


Figure 1: Macro-architecture of VGG16

2 Caltech 256 Classification

2.1 Dataset

The Caltech-256 dataset consists of images divided into 257 categories. For the purpose of our experiments we use the first 256 categories, leaving out the images from the 'clutter' category.

2.2 Methods

The images in the dataset are all of different sizes. These need to be resized and preprocessed appropriately before feeding into the network. Each image is reshaped to a size of 224x224 pixels and preprocessed in Keras using the `preprocess_input`. The corresponding labels are one-hot encoded as categorical outputs of dimension 256. The images in the dataset are then shuffled. Since we are using Transfer Learning, we train only the last layer of the network but the output layer in VGG-16 has 1000 nodes and needs to be modified. We replace this with a layer consisting of 256 nodes with softmax activation.

We study the network performance for varying size of training set used. The training set is composed by picking K examples from each class at random and shuffling. The model is trained and it's performance on the test set is observed. We use about 25 images from each class for the test set. We repeat for K = 2, 4, 6, 8, 16, 32, 60. We also visualize the activations of the filters in the first and last layers of the trained model. Next, we experiment with the network by using the intermediate layers as input to the softmax (output) layer and training just the last layer of the network. For this part, we pick a training set in a similar fashion as above, but with 8 images per class. Note that we restrict our experiments to only a few layers since this requires extremely high computational resources for training. Finally, to visualize filter activations (first and last convolutional layers) we pick a random sample from the test set and use the VGG-16 network.

2.3 Results and Discussion

2.3.1 Model Performance

The training loss decreases monotonically with number of epochs. However, the test loss doesn't follow a similar trend and we observe overfitting phenomenon after a few epochs (Figure 2). Classification accuracy increases as the size of the training set is increased (see Figure 3). Classification accuracies ranging from 39% (training with 2 images per class) to 76.74% (training with 60 images per class) were achieved.

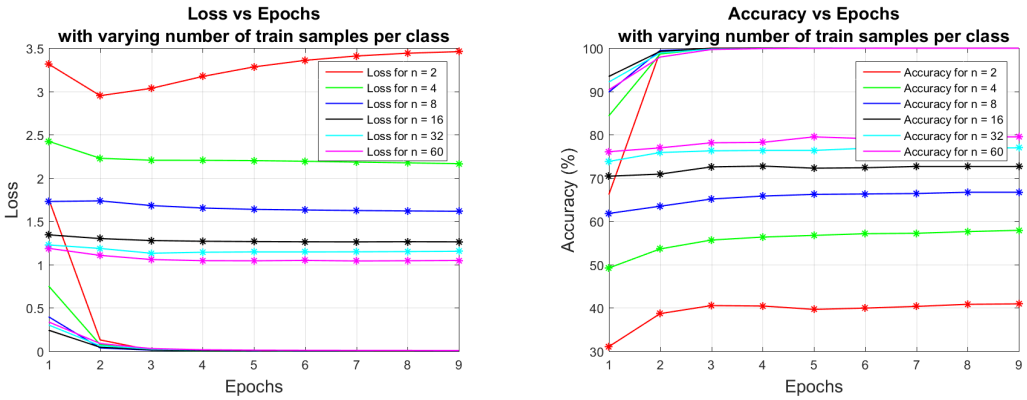


Figure 2: Transfer learning on Caltech256: (a) Loss vs epochs and (b) Accuracy vs epochs on the train (solid line) and test sets (solid line with $*$)

With Transfer Learning VGG-16 performs better than AlexNet (which is comparatively older) on Caltech-256. The VGG-16 network we are using is pre-trained on the ImageNet dataset which is

a much larger dataset than Caltech-256 (about 1.2 million images from 1000 categories vs approx. 30k images in the latter). We believe that with the same set of weights, the network is able to learn relevant high level representative features from the images that we could use to train just one layer and perform the classification task with great accuracy. In other words, Transfer Learning is successful here and this is because Caltech-256 is similar to ImageNet and is of a much smaller size.

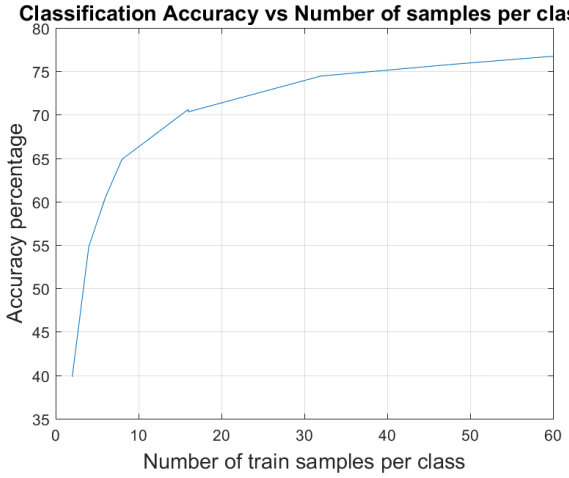
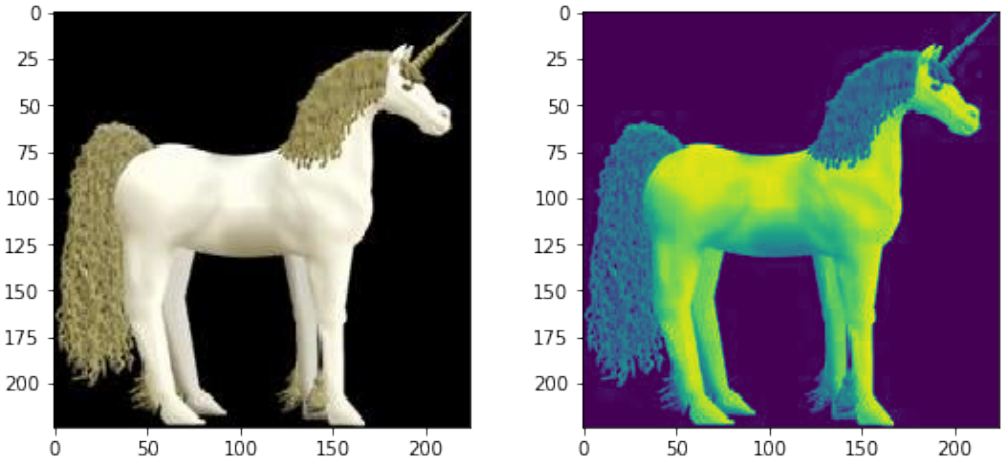


Figure 3: Transfer learning on Catech256: Accuracy on test set vs. number of samples used per class



a b

Figure 4: (a) Sample image from Caltech256 and (b) after preprocessing

We also visualize the filter activations of the first and last convolutional layers of VGG-16 network. We pass an image from the Test set after preprocessing it appropriately (Figure 4). From Figure 5, we believe that the first layer basically encodes color and edges present in the images. These are then used by the latter layers in the network to produce more complex representations. The activations for the final convolutional layer are rather sparse and not so easy to interpret (Figure 6).

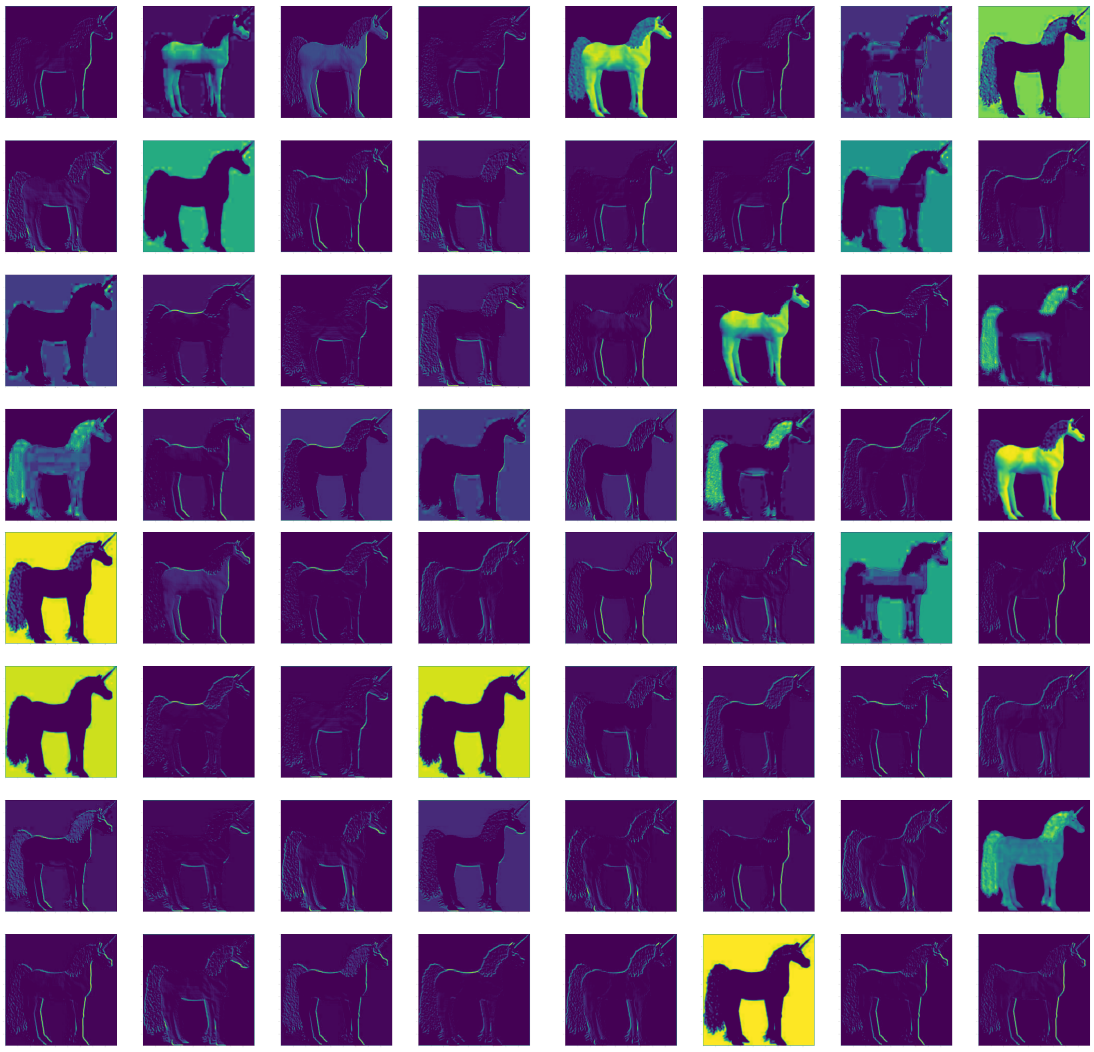


Figure 5: Filter activations after passing the sample image through the first layer

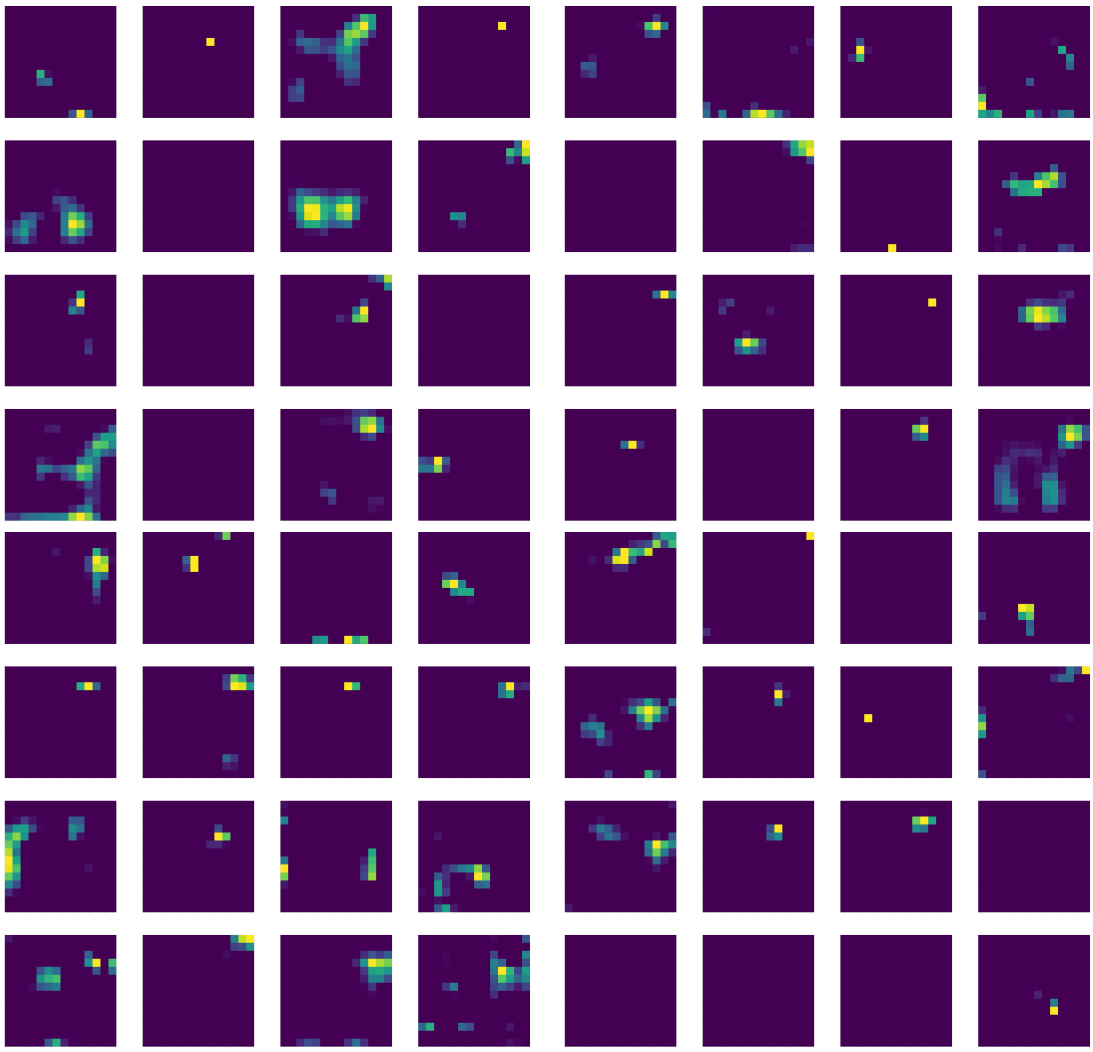
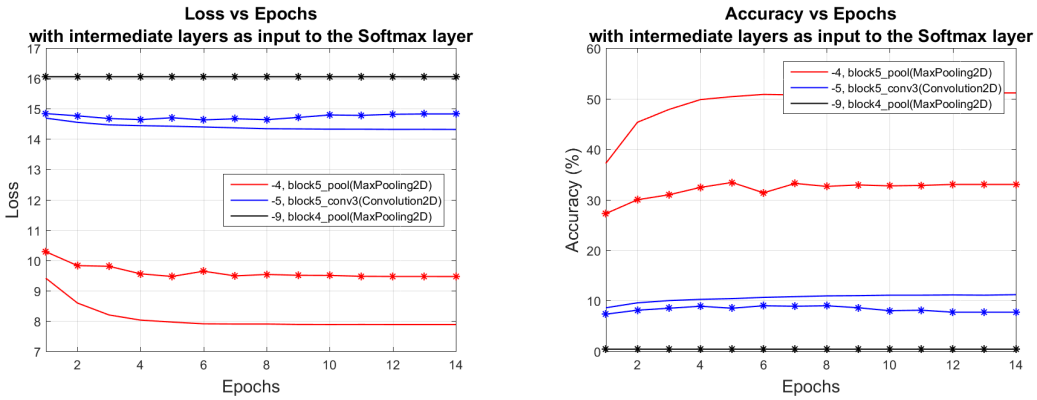


Figure 6: Filter activations after passing the sample image through the last layer

2.3.2 Feature Extraction



a b

Figure 7: Using intermediate layers: (a) Loss vs epochs and (b) Accuracy vs epochs on the train (solid line) and test sets (solid line with $*$)

Finally, we experiment using the intermediate layers of the network as input to the Softmax Layer. We observe that the classification accuracy decreases as we move the softmax layer closer to the input layer. (Figure 7).

We believe that this is because the representations learnt by the layers further up in the network are comparatively better to those learnt by the initial layers and this makes a difference in the network's prediction performance.

3 Urban Tribes Classification

3.1 Dataset

In this section we are looking at Urban Tribes dataset in the context of recognition of social styles of people. We look at classification of people into social groups such as bikers, hiphop, hipsters, formal etc. This dataset consists of 11 classes in total.

3.2 Methods

For this task we pre-process the images in same way as described in section 2.2. The difference being that the labels are one-hot encoded as categorical outputs of dimension 11. Similarly the last layer of modified VGG-16 network will now have 11 nodes with softmax activation.

We study the model performance for different size of training set. The size of test set is fixed to 25 images per class for the analysis. We the train the model and observe it's performance on the test set. Model is trained using training sets with 2, 4, 6, 8, 16, 32, 64 samples per class.

For visualizing filter activations, again we use the VGG-16 network. A sample (figure 10), selected at random, is used as input to the above model. Activations of first and last convolutional layers are recorded and visualized.

Next, we experiment with the network by using the intermediate layers as input to the softmax (output) layer and training just the last layer of the network. For this part, we pick a training set in a similar fashion as above, but with 8 images per class. Note that we restrict our experiments to only a few layers since this requires extremely high computational resources for training.

3.3 Results and Discussion

3.3.1 Model Performance

We observe a similar trend in training and test loss as we did for the previous case (Figure 8). Classification accuracy increases as the size of the training set is increased (see Figure 9). The classification accuracies varied from 30% (training with 2 images per class) to 56% (training with 64 images per class). The highest accuracy achieved was 56.0% percent for a training set with 64 images from each class. We note that although the dataset used is relatively small (about 1.2k images) compared to Caltech-256, the model's accuracy is comparatively lower for the same size of training and test sets. We believe this is because the Urban Tribes dataset is not similar to ImageNet dataset unlike Caltech-256, and so transfer learning isn't as successful as it was Caltech-256.

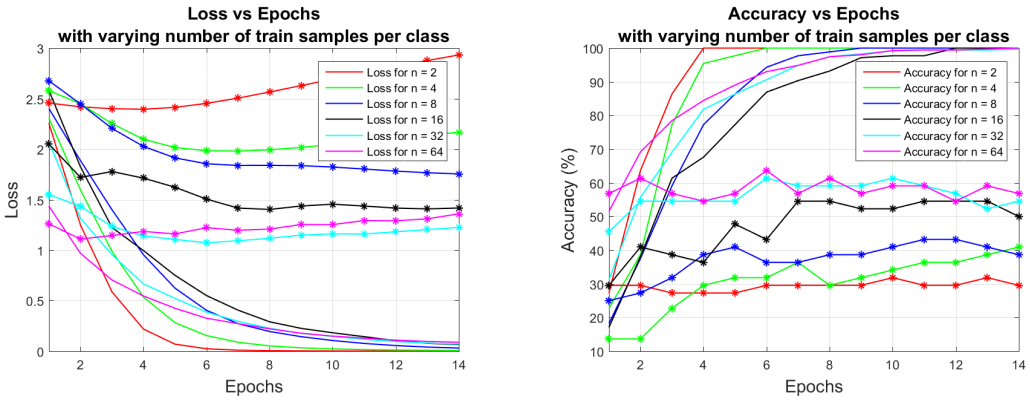


Figure 8: Transfer learning on Urban Tribes: (a) Loss vs epochs and (b) Accuracy vs epochs on the train (solid line) and test sets (solid line with $*$)

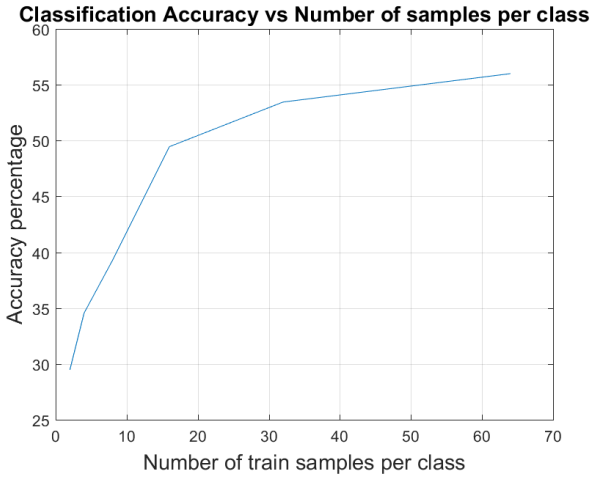
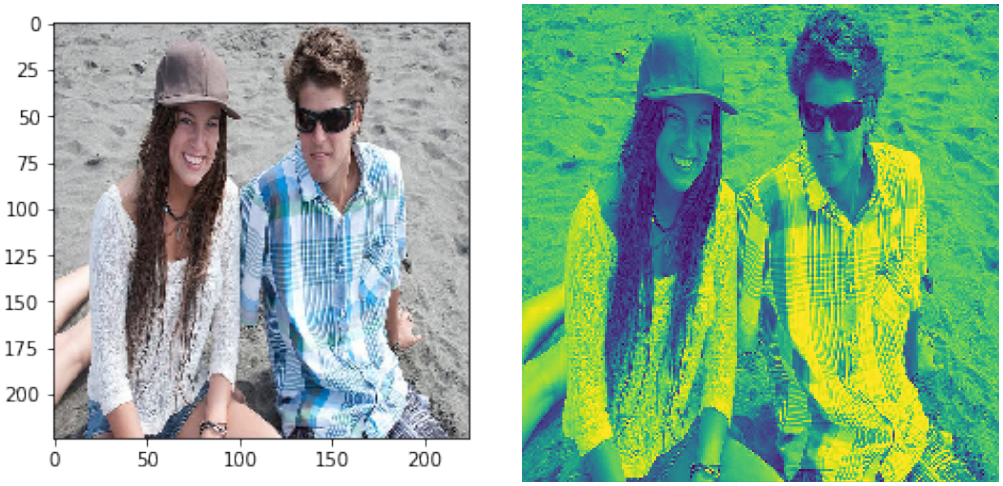


Figure 9: Transfer learning on Urban Tribes: Accuracy on test set vs. number of samples used per class



a b

Figure 10: (a) Sample image from UrbanTribes and (b) after preprocessing

As before, we visualize the filter activations of the first and last convolutional layers of the VGG-16 network. We pass an image from the Test set after preprocessing it appropriately (Figure 10). From Figure 11, we believe that the first layer basically encodes color and edges present in the images. These are then used by the latter layers in the network to produce more complex representations. The activations for the final convolutional layer are rather sparse and not so easy to interpret (Figure 12).

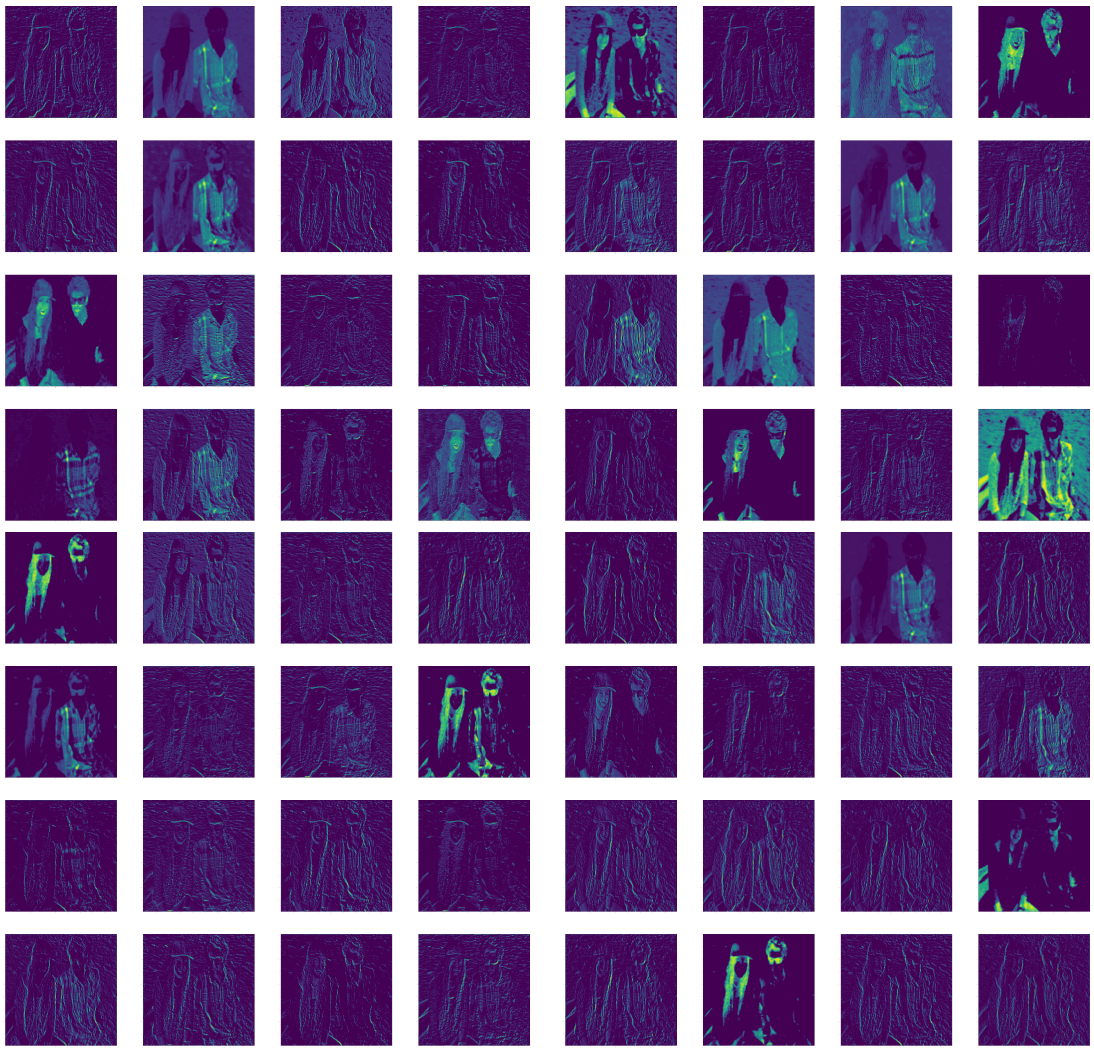


Figure 11: Filter activations after passing the sample image through the first layer

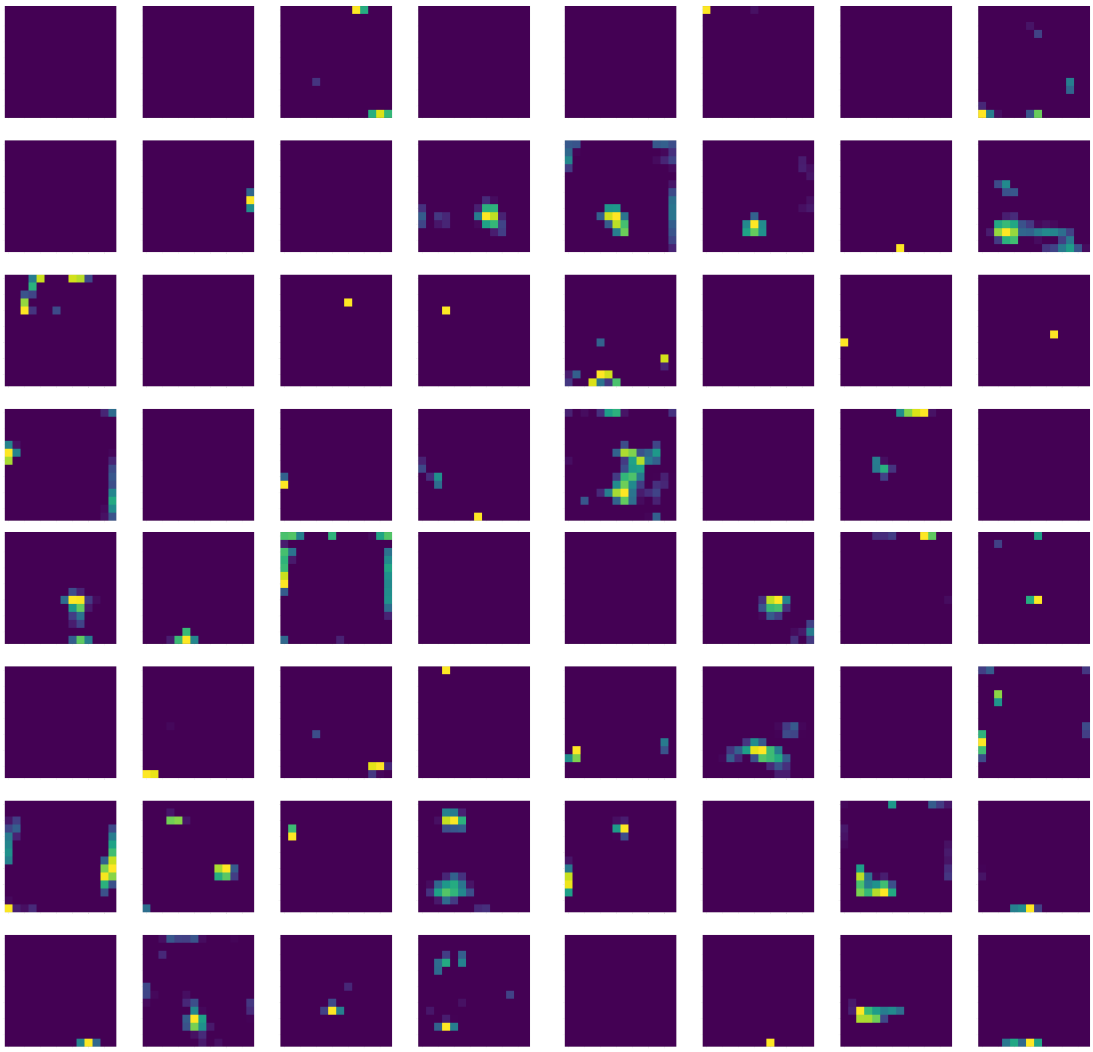


Figure 12: Filter activations after passing the sample image through the last layer

3.3.2 Feature Extraction

Again, we experiment using the intermediate layers of the network as input to the Softmax Layer. Like in the previous case, the classification accuracy decreases as we move the softmax layer closer to the input layer. (Figure 13). We believe that this is because the representations learnt by the layers further up in the network are comparatively better to those learnt by the initial layers and this makes a difference in the network's prediction performance.

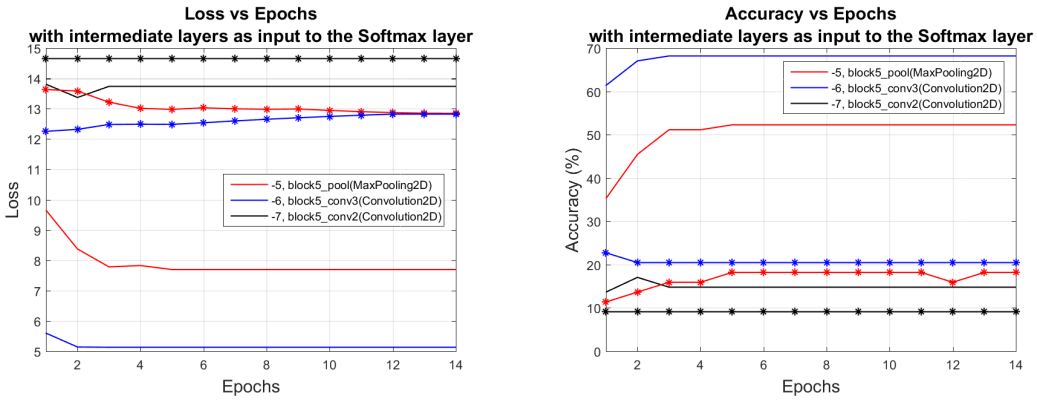


Figure 13: Using intermediate layers: (a) Loss vs epochs and (b) Accuracy vs epochs on the train (solid line) and test sets (solid line with $*$)

4 Temperature-based Softmax Regression

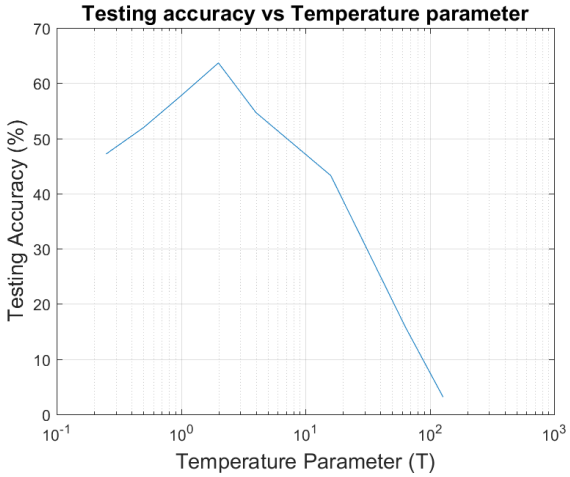


Figure 14: Parameter sweep for finding the optimal temperature

4.1 Methods

The *temperature* based softmax regression is employed as described in Section 1.1.1 on the Caltech 256 dataset.

4.2 Results and Discussion

We observed that using a temperature parameter in general leads to slow learning but the learnt network doesn't overfit the training data. Using a training set of 8 images per class and a validation set of 4 images per class. We found that using a value of 2 for the temperature gave the best performance (Figure 14). Decreasing the temperature below this lead to a sub-optimal performance and increasing the temperature above 2 lead to rapid decrease in performance. This is because the increasing the value of temperature maxes the distribution at the softmax more flat and thus losing information. We note that the accuracy on the test set is *63%*, which is comparable to the result obtained from using vanilla transfer learning. Next, we increased the number of training images to

32 images per class to validate that our chosen value for the temperature parameter would still work. We obtained an accuracy of 72% which is also comparable to the result obtained from using vanilla transfer learning. We hypothesized that if we decreased the number of training images per class then the temperature based softmax would perform better than the vanilla transfer learning from Section 1. The reason behind the hypothesis is that using the temperature based softmax would prevent overfitting. Repeating the experiment with 2 training images per class we obtain an accuracy of 46%, which is an 18% increase in the accuracy obtained from vanilla transfer learning (39%). The temperature based softmax does seem to prevent overfitting.

5 Summary

In this report we have demostrated that large pretrained models can be effectively used in related machine learning tasks using relatively insignificant resources. Achieved an accuracy of 76.7% on Caltech256 dataset and 56.0% on Urban Tribes dataset. Visualising the filter activations helped to improve our understanding of the effectiveness of transfer learning method and also the working of Convolutional Networks. The lower convolutional layers of the network have a small effective receptive field and tend to extract low level features such as edges. But the top layer convolutional layers have huge effective receptive field (here almost half the width of the image) and extract high level features which are very difficult to comprehend. Experimenting with different intermediate convolutional layers as input to softmax also revealed an important and surprising result. Owing to the fact that almost 90% of weight parameters are in the top FC dense layers, most of the predictive power of the network is from those layers. This is demostrated by the fact that test accuracy drops rapidly as we go deeper to connect our softmax layer. We also explore another Temperature based softmax regression for Transfer learning. T=2 was found to be most suitable parameter which resulted in 46% accuracy (2 training images per class). This is an 18% increase in accuracy compared to using vanilla Transfer learning.

6 References

1. *Efficient Backpropagation* by Yann LeCun, Leon Bottou, Genevieve B. Orr and Klaus-Robert Miller.
2. *Visualizing and Understanding Convolutional Networks* by Matthew D Zeiler and Rob Fergus. eprint arXiv:1311.2901
3. *Very Deep Convolutional Networks for Large Scale Image Recogntion.* by Karen Simonyan and Andrew Zisserman.
4. *Distilling the Knowledge in a Neural Network* by Geoffrey Hinton, Oriol Vinyals and Jeff Dean. eprint arXiv:1503.02531
5. *ImageNet Classification with Deep Convolutional Neural Networks* by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton.