CSE 253: Neural Networks
Winter 2017
Transfer Learning with VGG16

Swapnil Taneja (PID: A53219137)
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
swtaneja@eng.ucsd.edu

Saksham Sharma (PID: A53220021)
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
sas111@eng.ucsd.edu

Kriti Aggarwal (PID: A53214465)
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
kriti@eng.ucsd.edu

Prahal Arora (PID: A53219500)
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
prarora@eng.ucsd.edu

Saicharan Duppati (PID: A53221873)
Department of Computer Science and Engineering
University of California, San Diego
San Diego, CA 92093
sduppati@eng.ucsd.edu

Abstract

In this assignment we explored the use of **Transfer Learning** to solve new tasks using the knowledge gained from solving related tasks.The task at hand was to classify images in our datasets (**Caltech256** and **Urban Tribes**) similar to the ImageNet dataset, using a pre-trained CNN model (VGG16 ConvNet model). VGG16 consists of a set of Convolution Layers (including Maxpooling) followed by fully-connected layers and finally a Softmax Layer which is used to predict 1000 classes of ImageNet. We replaced the Softmax Layer of the VGG16 pre-trained model by our own Softmax Layer which predicted classes of our datasets. We attempted to train only the Softmax Layer of the new model on just a few samples of our datasets to observe if it performs well. The model achieved an accuracy of ap-

prox. **65%** on **Caltech256** dataset when 16 training examples per category were used to train the model. The model was also tested on **Urban Tribes** dataset and achieved an accuracy of approx. **51%** when 16 training examples per category were used to train the model.

1 Caltech256 Classification (25 points)

1.1 Introduction

The goal in this problem is to explore the application of **Transfer Learning** to the task of recognizing a set of object categories from images present in Caltech256 dataset. In this section we explore various techniques to train our new model on the Caltech 256 dataset. We use the Keras library to get the VGG16 model up to the second last layer and add an additional Dense layer with softmax activation on top of it. We leverage the methods of Model Class for reading in the images and training on the dataset. In the first approach we use the methods $ImageGenerator()$, $flow\_from\_directory()$ (to read images) and $fit\_generator()$ (to train images). In the second approach, we introduce optimizations for faster execution of code. The optimization introduced lets us reuse the activations and allows to run the code for many epochs. We give more insight into the Methods in the next section.

1.2 Methods

In the first approach, we exploit the directory structure of the Caltech 256 Dataset and the Keras library to read in the data and tag labels for each image. We then train the model using the standard $fit\_generator()$ method. We ran python scripts to segregate images into Training and Validation Set and then read the images by leveraging the directory structure. We test our data on images that has 8 images per class and train on the data that has 2, 4 , 8, or 16 images per class. We ran the code on the comet cluster on XSEDE. The standard compute nodes consist of Intel Xeon E5-2680v3 (formerly codenamed Haswell) processors, 128 GB DDR4 DRAM (64 GB per socket), and 320 GB of SSD local scratch memory. The GPU nodes contain four NVIDIA GPUs each. The large memory nodes contain 1.5 TB of DRAM and four Haswell processors each. The network topology is 56 Gbps FDR InfiniBand with rack-level full bisection bandwidth and 4:1 oversubscription cross-rack bandwidth. Comet has 7 petabytes of 200 GB/second performance storage and 6 petabytes of 100 GB/second durable storage. It also has dedicated gateway hosting nodes and a Virtual Machine repository. External connectivity to Internet2 and ESNet is 100 Gbps. For the purpose of our execution we use gpu-shared partition on the comet cluster. We use SLURM workload manager for running jobs on the machine . In the appendix we mention the scripts. In the second approach, in an attempt for faster execution , we exploit the redundancy in the activations generated at the second last layer. We keep our train and test data consistent and we observe that the weights up to the second last layer are not trainable and hence will remain same. Since, the inference on the same data for each training option $\{2, 4, 6, 8\}$ until the second last layer is same, we persist the activations of the second last layer and reuse them by loading them back for subsequent training and testing. This approach introduces an initial cost for running inference on the train and test data which is one time only. We were able to run for 200 epochs in a reasonable time on Macbook Air.

1.3 Results and Discussion

Using the first approach, we trained on $\{2, 4, 6, 8\}$ and tested on 8 images per class for 11 epochs. Following were the execution times on the comet cluster :
For training on 2 images per class, it ran for 134.538 minutes. For 4 images per class, it ran for 161.518 minutes. For 8 images per class, it ran for 257.408 minutes and for 16 images per class, it ran for 366.140 minutes. In the second approach, we get 50 , 20 , 40 and 120 minutes for the similar configuration and for 200 epochs on Macbook air. This variation in time results from our approach of one time persistence on inference generated for the test data on 8 samples per class.

The answers to the questions given in the assignment are as follows:
Question 1: Training (4 points)
Train the ConvNet with a small number of the training images per class. The idea is to

replicate the Figure in the last lecture, which is Figure 9 from this paper. Try repeating the same, a few times, with larger subsets. As this will get more expensive computationally, the more examples you use, you dont have to replicate that whole figure. E.g., try 2, 4, 8, and 16 training examples per category (more if you have time). Report the performance of the model.
Answer 1:
We analyze the performance of the trained model on different configurations and plot the results in Figures in the following sections . We observe 34.8 , 43.2, 51.6 and 58.75 percent after 11 epochs on 2, 4, 8 , 16 images per class . For our second approach, we achieve a significant performance. We get 40.9 , 43.38 , 61.9 and 63.18 percent after 200 epochs.

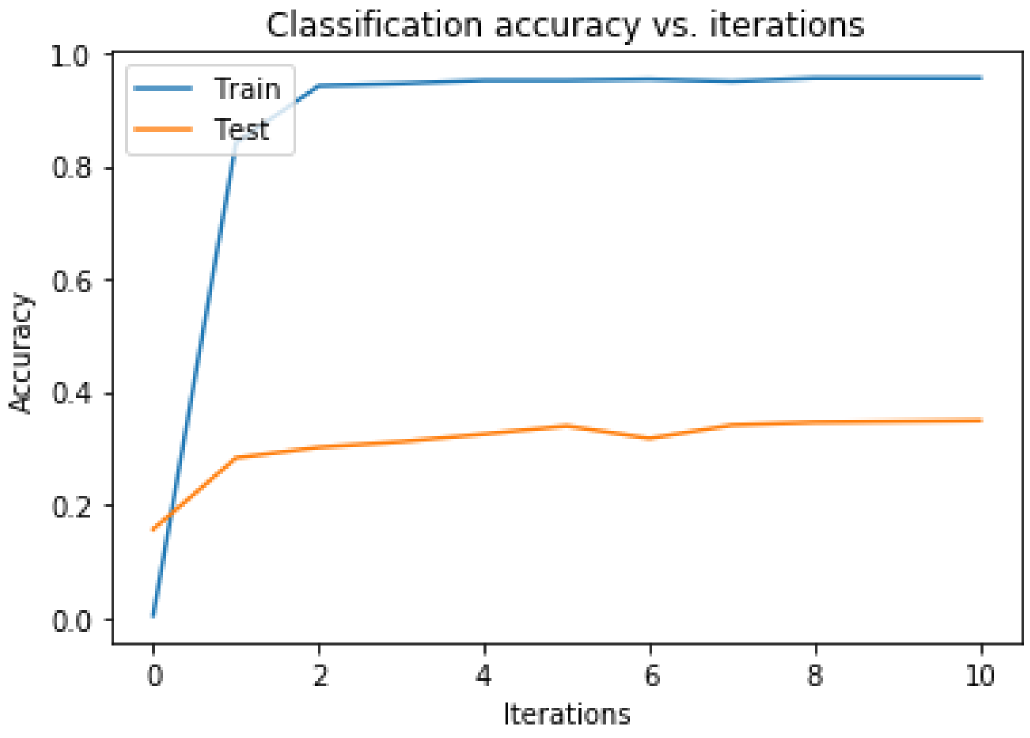


Figure 1: Comet (Caltech256): Accuracy plots when trained on 2 images per class

Question 2: Inference
(a) Plot train and test loss vs. iterations. (2 points)
Answer 2(a):
The train and test loss vs. iterations for the four cases, i.e. $2, 4, 8$ and 16 training images per category for the first approach are shown in Figure 5, Figure 6, Figure 7 and Figure 8 respectively.
The train and test loss vs. iterations for the four cases, i.e. $2, 4, 8$ and 16 training images per category for the second approach are shown in Figure 9, Figure 10, Figure 11 and Figure 12 respectively.

(b) Plot classification accuracy vs. iterations. (2 points)
Answer 2(b):
The test and train classification accuracy for the four options $2, 4, 8, 16$ training images per class for the first approach is shown in Figures 1 2 3 4
The test and train classification accuracy for the four options $2, 4, 8, 16$ training images per class for the second approach is shown in Figures 13 14 15 16

(c) Plot classification accuracy vs. number of samples used per class. (2 points)
Answer 2(c):

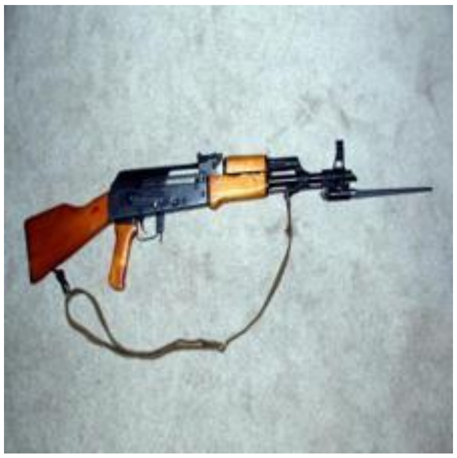
The classification accuracy vs number of samples is shown in Figure 17 for the first approach. And the classification accuracy vs number of samples is shown in Figure 18 for the second approach.

(d) Discuss the plots and report your inference from them. (6 points)
Answer 2(d):
The plots of the accuracy and loss confirm our hypothesis that the model trained on Imagenet dataset provides filters relevant for image classification on Caltech 256 dataset. For the loss and accuracy plots, we observe that with training the last layer, the accuracy on the test data increases (loss decreases) as expected. The resultant trained model provides a accuracy that beats the state of the art results from 30 images per class ($Boet.al$2013) by 2.2 percent at approx 6 images per class from second approach and 12 images per class from the first approach. We believe this result is different due to the number of epochs the models are trained on , in different approaches.
(e) Visualize filters from layers first and last Convolution Layers of the trained model. (4 points)
Notation
Convolution set: It is representative two (or three) convolution layers and a maxpool layer. In VGG16, there are five such Convolution sets For all the below representation, when we say convolution layer $x - y$ what follows are the first six convolved outputs of y^{th} convolutional layer in x^{th} convolutional set

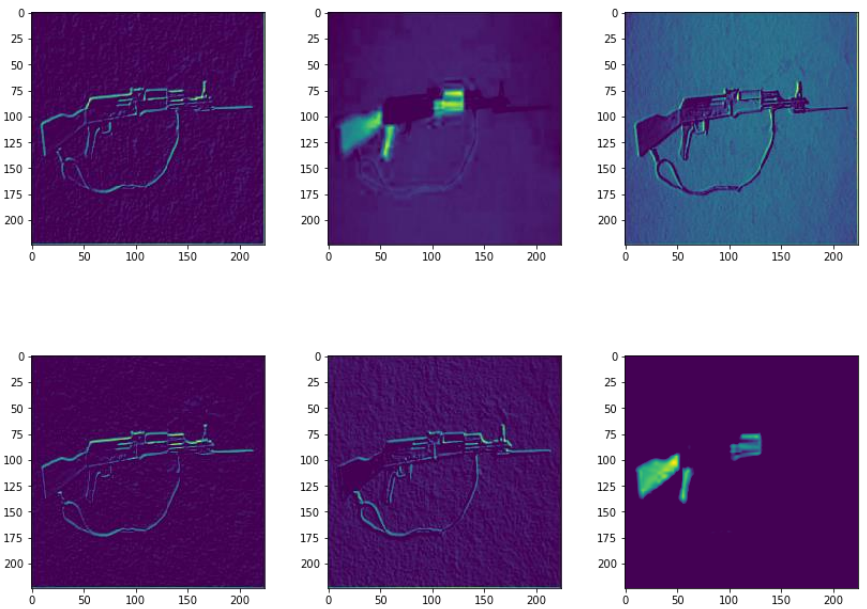
Answer 2(e):

Visualization for Caltech-256 data

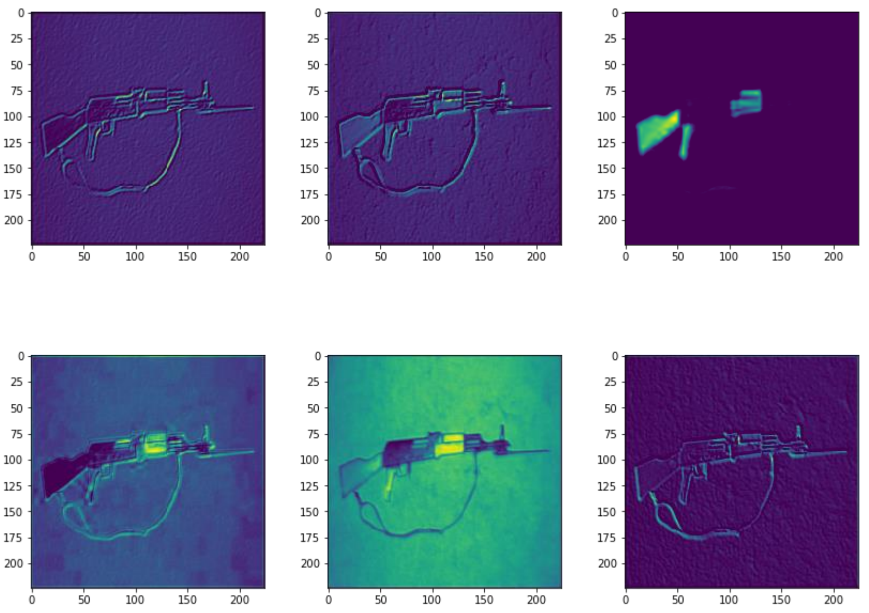
Input Image



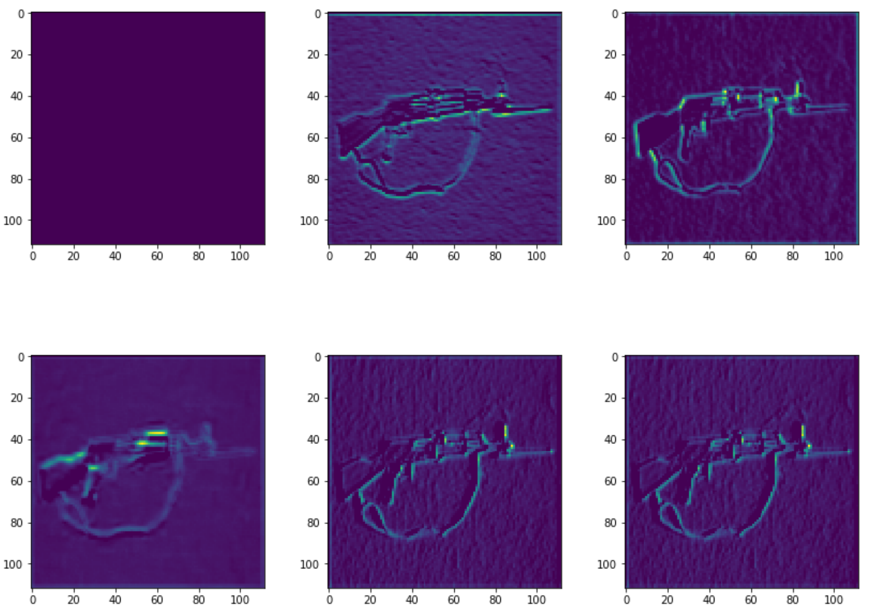
Convolution Layer 1-1



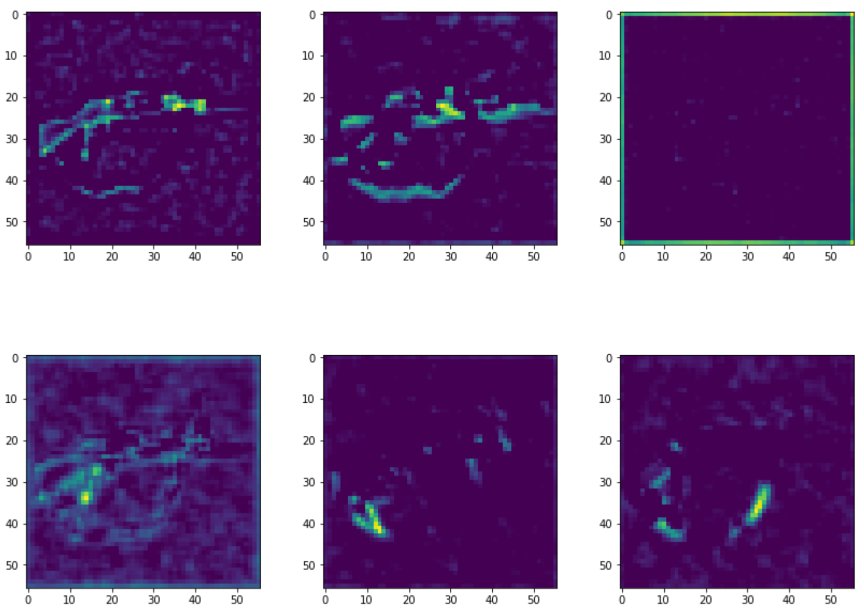
Convolution Layer 1-2



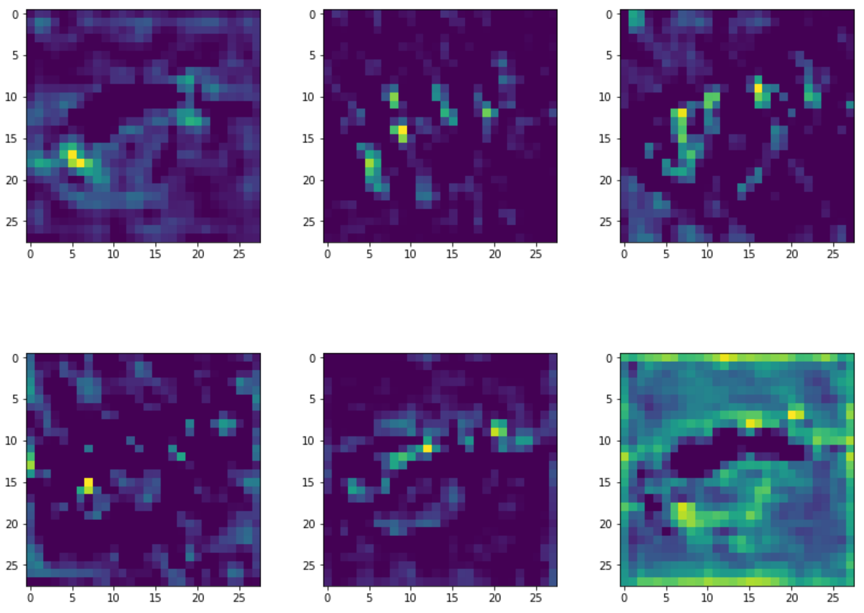
Convolution Layer 2-1



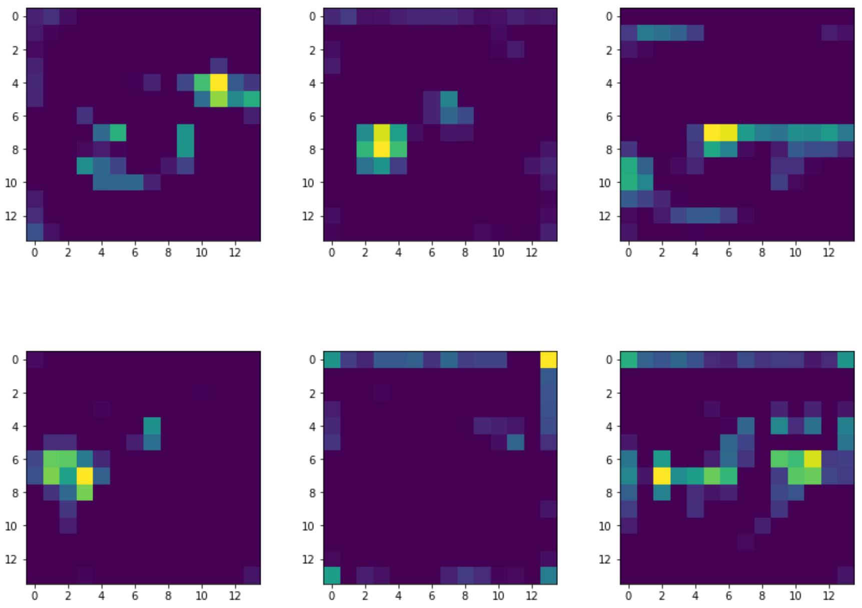
Convolution Layer 3-1



Convolution Layer 4-1



Convolution Layer 5-1



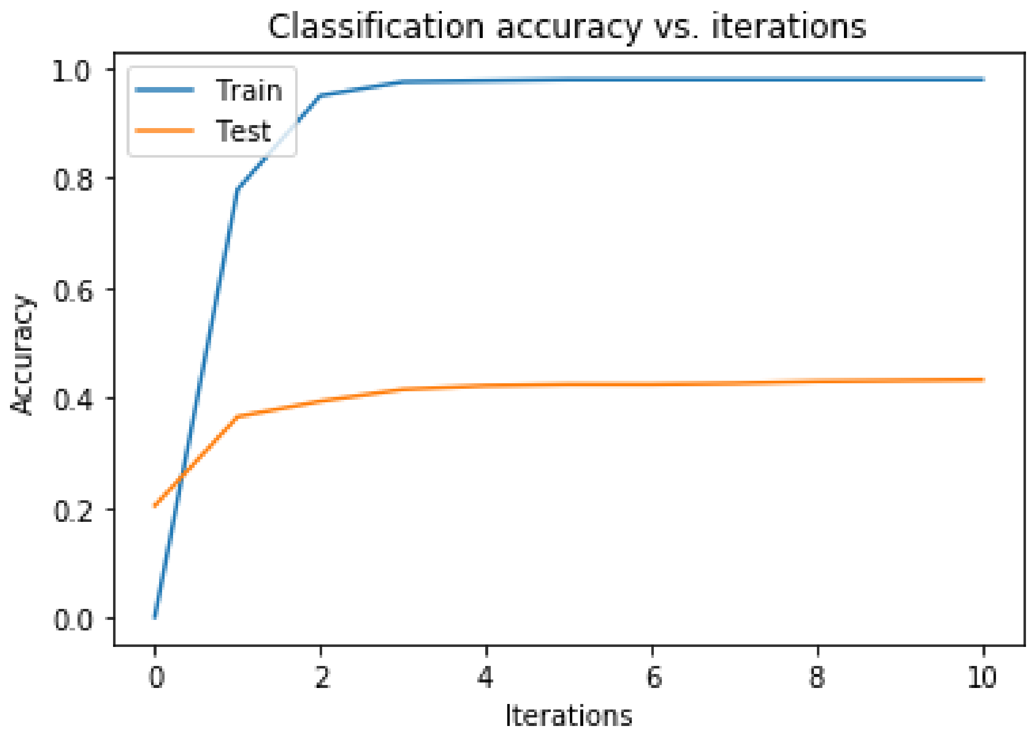


Figure 2: Comet (Caltech256): Accuracy plots when trained on 4 images per class

Question 3: Feature Extraction (5 points)
(a) Experiment using the intermediate Convolutional Layers as input to the Softmax Layer.
(b) Train the network again on a subset of the data.
(c) Discuss what you observe and provide empirical results to support it.
Answer 3:
Analysis of the discriminative information contained from intermediate convolution layer of feature maps within our ImageNet pretrained convnet is shown in Figure 19 20 21 22. We trained a softmax on features from different layers (from end of 4th convolutional set and 5th convolutional set following notation from previous question) from the convnet. Higher layers generally produce more discriminative features, as we can see that the accuracy on the subset of Caltech 256 is increased, when we plug the softmax layer after the convolution layer at a deeper level. This matches out intuition and theoretical estimations that the more rich features are learned at higher layers, which can discriminate an image in a more profound manner.

ω	average diff	max diff
ω_{ij}	0.0000002575	0.000001599
ω_{jk}	0.000000009493	0.0000001594

2 Urban Tribes Classification

2.1 Introduction

The goal in this problem is to explore the application of another variant of **Transfer Learning** to the task of recognition of social styles of people. In particular, we have to classify the Urban Tribes dataset into groups such as bikers, hipshop, formal, goth and many others. In this section we explore various techniques to train our new model on the urban tribes dataset. We use the Keras library to get the VGG16 model up to the second last layer and add an additional Dense layer with softmax activation on top of it. We leverage the methods of Model Class for reading in the images and

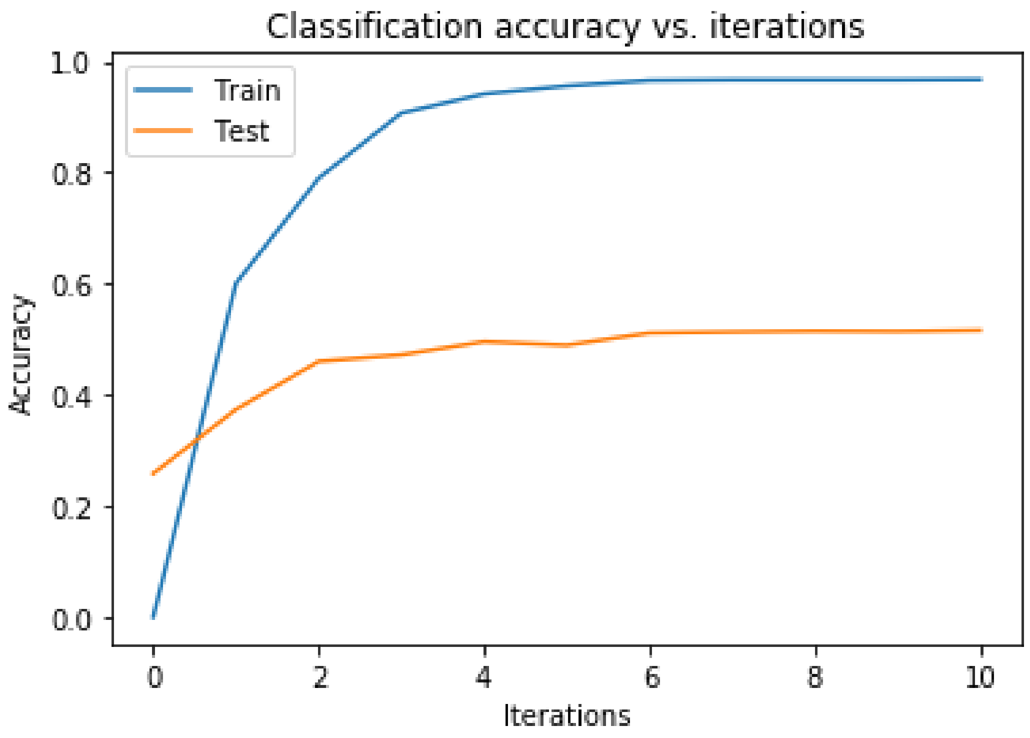


Figure 3: Comet (Caltech256): Accuracy plots when trained on 8 images per class

training on the dataset. We used the methods $ImageGenerator()$, $flow\_from\_directory()$ (to read images) and $fit\_generator()$ (to train images).

2.2 Methods

Similar to how we proceeded with Question 1, we ran python scripts to segregate images into Training and Validation Set and then read the images by leveraging the directory structure. We test our model on test data that has 8 images per class and train on the data that has 2, 4 , 8, and 16 images per class respectively. In the appendix we mention the scripts. In an attempt for faster execution , we exploit the redundancy in the activations generated at the second last layer. We keep our train and test data consistent and we observe that the weights up to the second last layer are not trainable and hence will remain same. Since, the output/forward-propagation of the input data for each training option $\{2, 4, 6, 8\}$ until the second last layer of VCG16 is same, we persist the activations of the second last layer and use them as the input to a new model which only contains one dense layer and a softmax activation. This approach introduces an initial cost for running forward-propagation for VCG16 model(-1 layer) on the train and test data which is a one time job only. We were able to run for 200 epochs in a reasonable time on Macbook Air because we are only training this one-layered model now.

2.3 Results and Discussion

The answers to the questions given in the assignment are as follows:

Question 1: Training (4 points)
Train the ConvNet with a small number of the training images per class. The idea is to replicate the Figure in the last lecture, which is Figure 9 from this paper. Try repeating the same, a few times, with larger subsets. As this will get more expensive computationally, the more examples you use, you dont have to replicate that whole figure. E.g., try 2, 4, 8, and 16

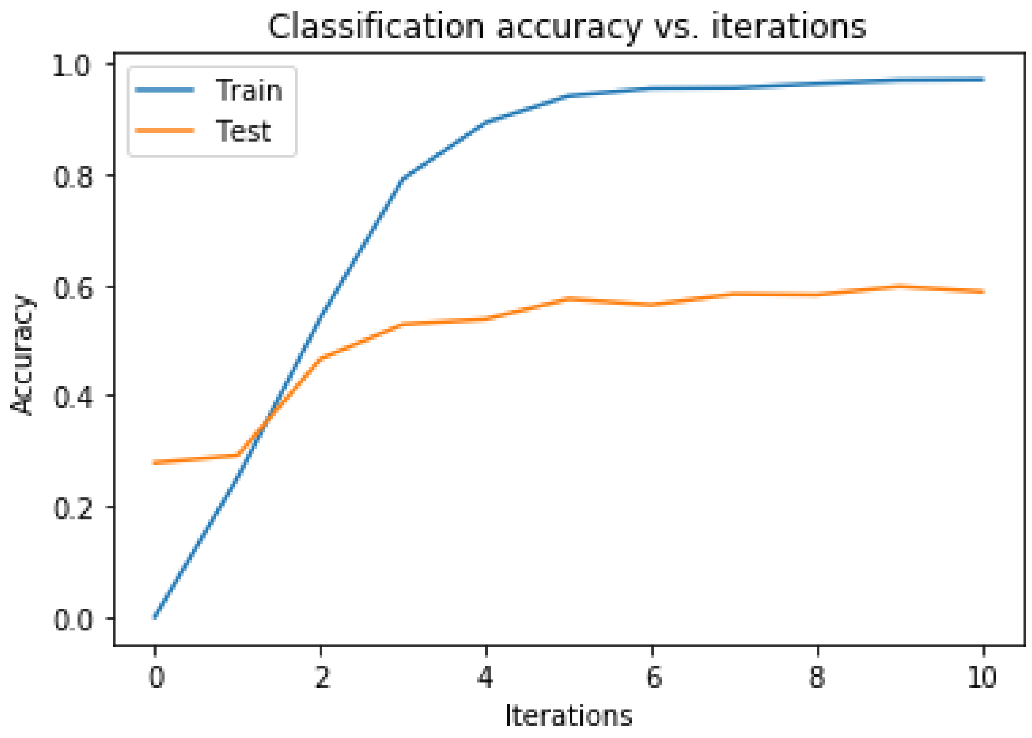


Figure 4: Comet (Caltech256): Accuracy plots when trained on 16 images per class

training examples per category (more if you have time). Report the performance of the model.
Answer 1:
We found that we were able to replicate the results of the paper as hinted by the question. As and when the number of images per class are increased, the accuracy on the validation set also increases. For 2, 4, 6 and 8 images per class, we achieve the accuracy of 22%, 31%, 38% and 47% Please refer the Figures 27, 28, 29 and 30

Question 2: Inference
(a) Plot train and test loss vs. iterations. (2 points)
Answer 2(a):

The train and test loss vs. iterations for the four cases, i.e. $2, 4, 8$ and 16 training images per category for the second approach (the only approach we used for this dataset) are shown in Figure 23, Figure 24, Figure 25 and Figure 26 respectively.

(b) Plot classification accuracy vs. iterations. (2 points)
Answer 2(b):
The train and test accuracy vs. iterations for the four cases, i.e. $2, 4, 8$ and 16 training images per category for the second approach (the only approach we used for this dataset) are shown in Figure 27, Figure 28, Figure 29 and Figure 30 respectively.

(c) Plot classification accuracy vs. number of samples used per class. (2 points)
Answer 2(c):
The classification test accuracy vs number of samples = [2,4,8,16] is shown in Figure 31.

(d) Discuss the plots and report your inference from them. (6 points)
Answer 2(d): Discussion
Observations/Inference for (a)

1. The train loss decreases with number of iterations in each of the cases. This implies that the network is learning

2. The increase in the test loss with number of iterations is maximum in case of 2 followed by 4,8,16 in that order. This is because the neural network is over fitting on few number of training samples which cannot be generalized to test cases

Observations/Inference for (b)

1. The train accuracy quickly rises to 1 in each of the cases, which is an indicative of learning

2. The test accuracy saturates after a certain number of epochs (approximately after 20) and stops increasing. This shows that the network is saturated and cannot learn anything new given the training data

Observations/Inference for (c)

1. The test accuracy increases with number of training samples. This is expected as the network is shown more number of images as part of training, It tends to learn more variations in the data and therefore performs better on test data

(e) Visualize filters from layers first and last Convolution Layers of the trained model. (4 points)
Convolution set: It is representative two (or three) convolution layers and a maxpool layer. In VGG16, there are five such Convolution sets For all the below representation, when we say convolution layer $x - y$ what follows are the first six convolved outputs of y^{th} convolutional layer

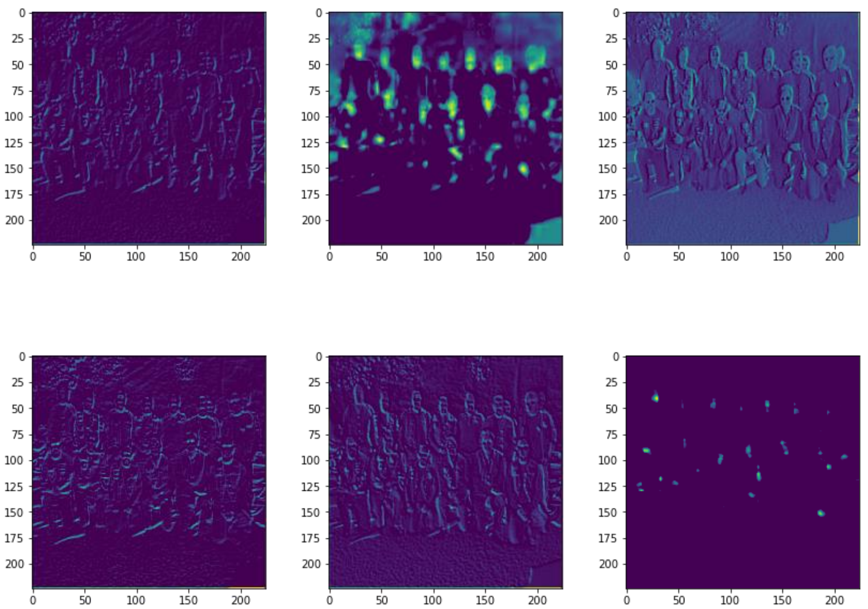
in x^{th} convolutional set

Answer 2(e):

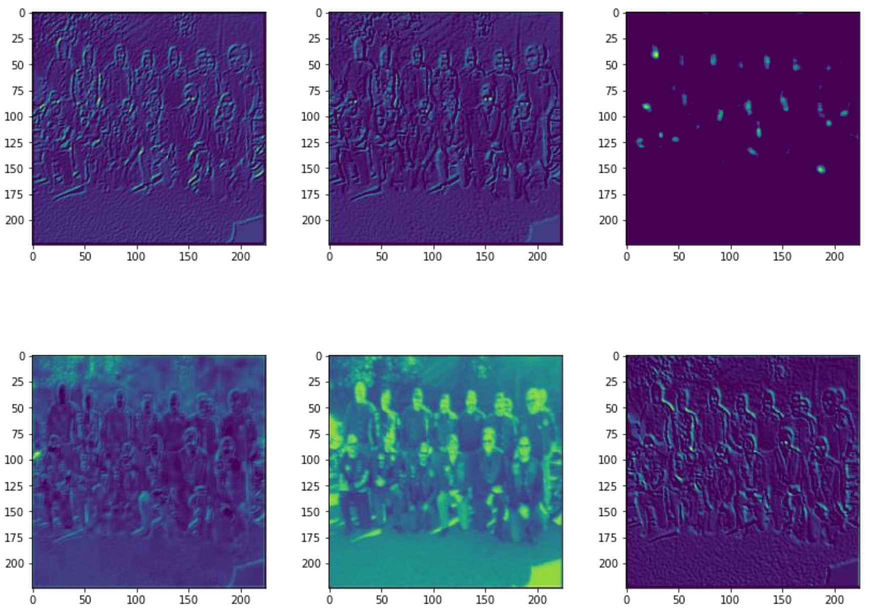
Visualization for Urban Tribes

Input Image

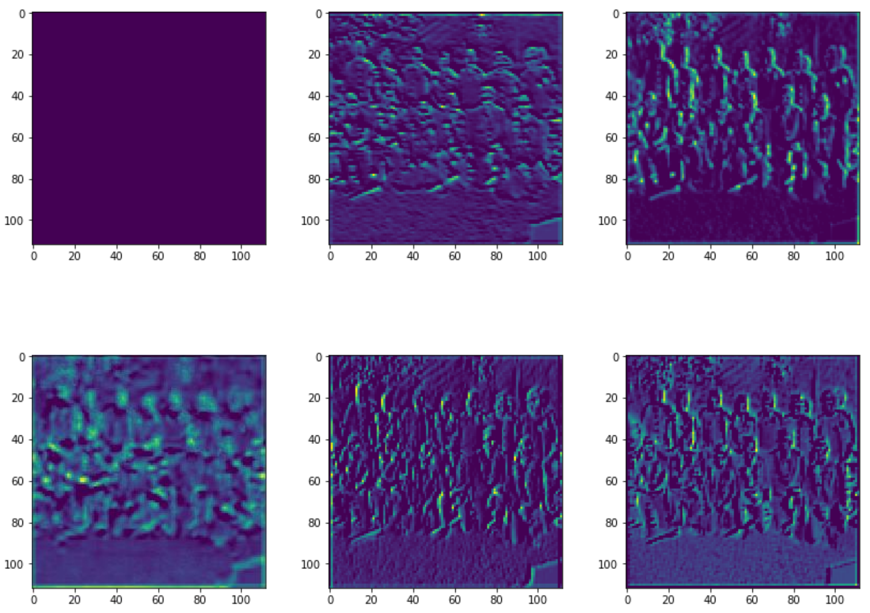
Convolution Layer 1-1



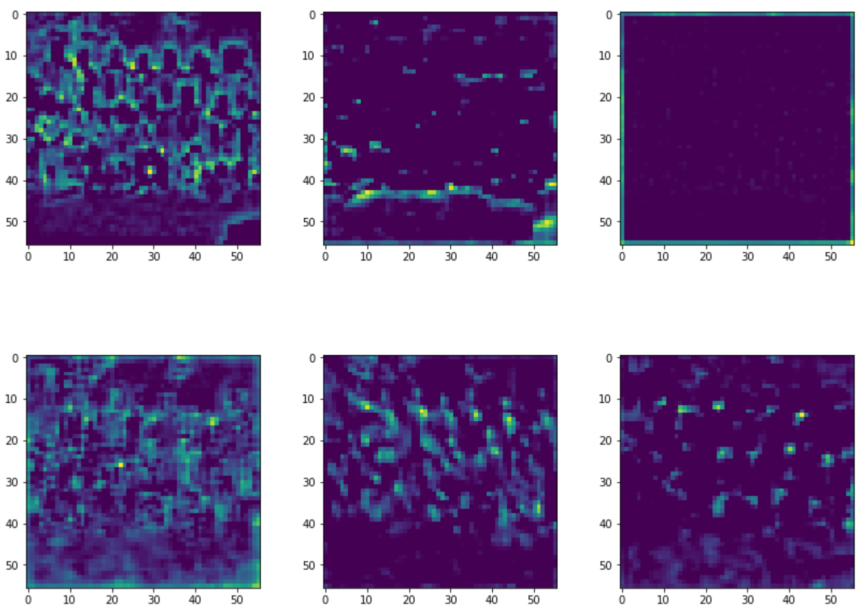
Convolution Layer 1-2



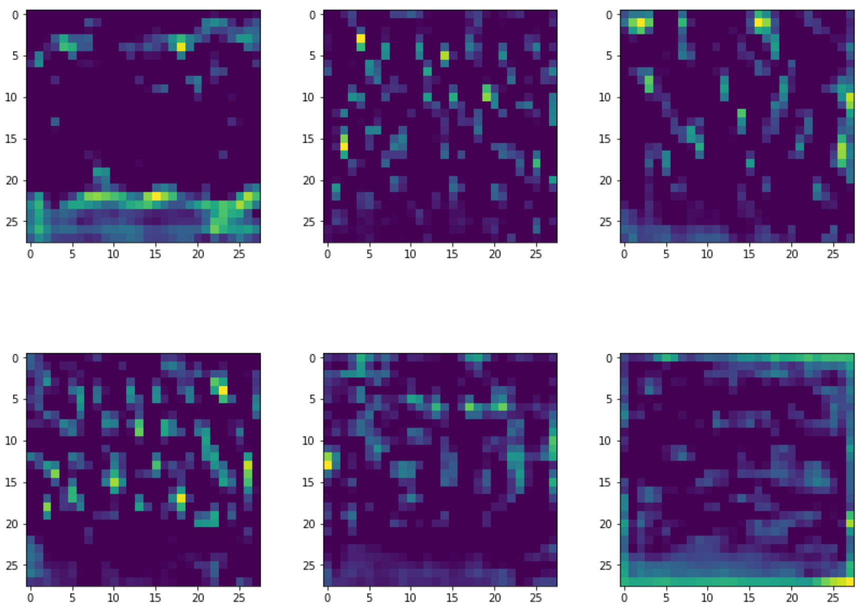
Convolution Layer 2-1



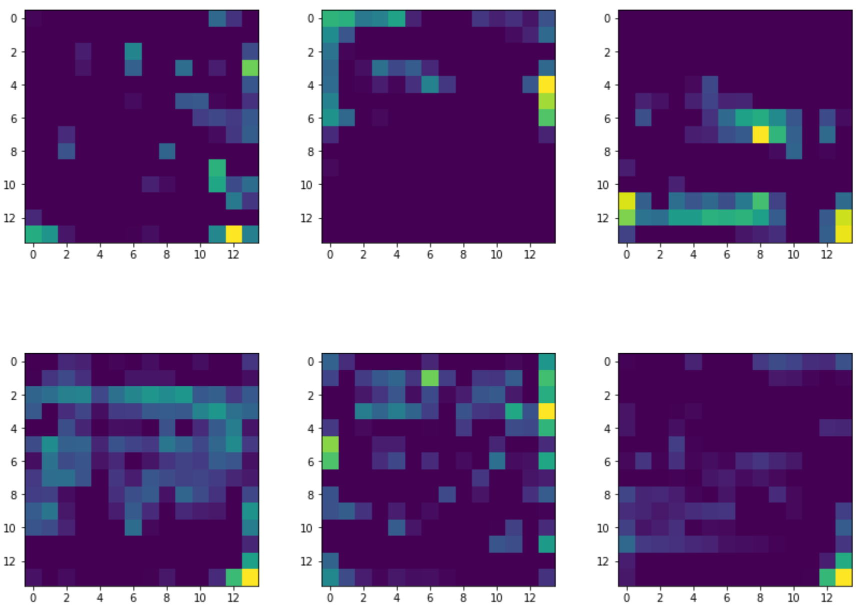
Convolution Layer 3-1



Convolution Layer 4-1



Convolution Layer 5-1



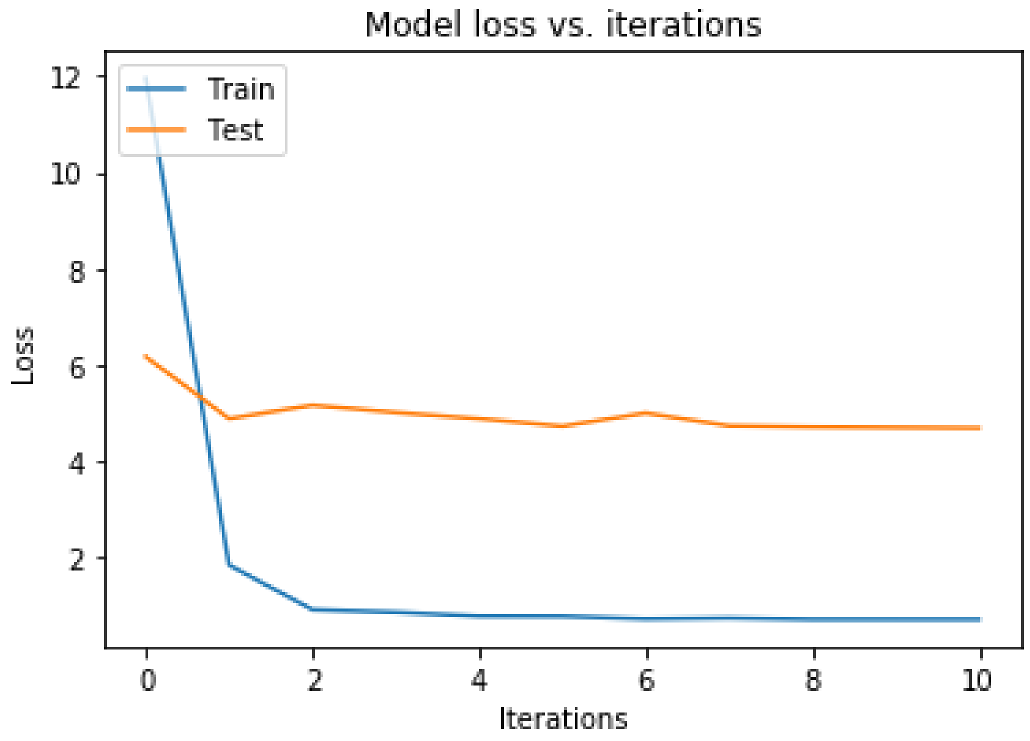


Figure 5: Comet (Caltech256): Loss plots when trained on 2 images per class.

Question 3: Feature Extraction (5 points)
(a) Experiment using the intermediate Convolutional Layers as input to the Softmax Layer.
(b) Train the network again on a subset of the data.
(c) Discuss what you observe and provide empirical results to support it.
Answer 3:

Analysis of the discriminative information contained from intermediate convolution layer of feature maps within our ImageNet pretrained convnet is shown in Figure 32 33 34 35. We trained a softmax on features from different layers (from end of 3rd convolutional set, 4th convolutional set and 5th convolutional set following notation from previous question) from the convnet. Higher layers generally produce more discriminative features, as we can see that the accuracy on the subset of Urban Tribe is increased, when we plug the softmax layer after the convolution layer at a deeper level. This matches out intuition and theoretical estimations that the more rich features are learned at higher layers, which can discriminate an image in a more profound manner. The results we observe are similar to the ones on Caltech 256.

3 Bonus Question (5%): Temperature-based Softmax Regression

3.1 Introduction

Neural networks typically produce class probabilities by using a softmax output layer that converts the logit, z_i, computed for each class into a probability, q_i, by comparing z_i with the other logits.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

where T is a temperature that is normally set to 1. Using a higher value for T produces a softer probability distribution over classes. In the simplest form of distillation, knowledge is transferred to

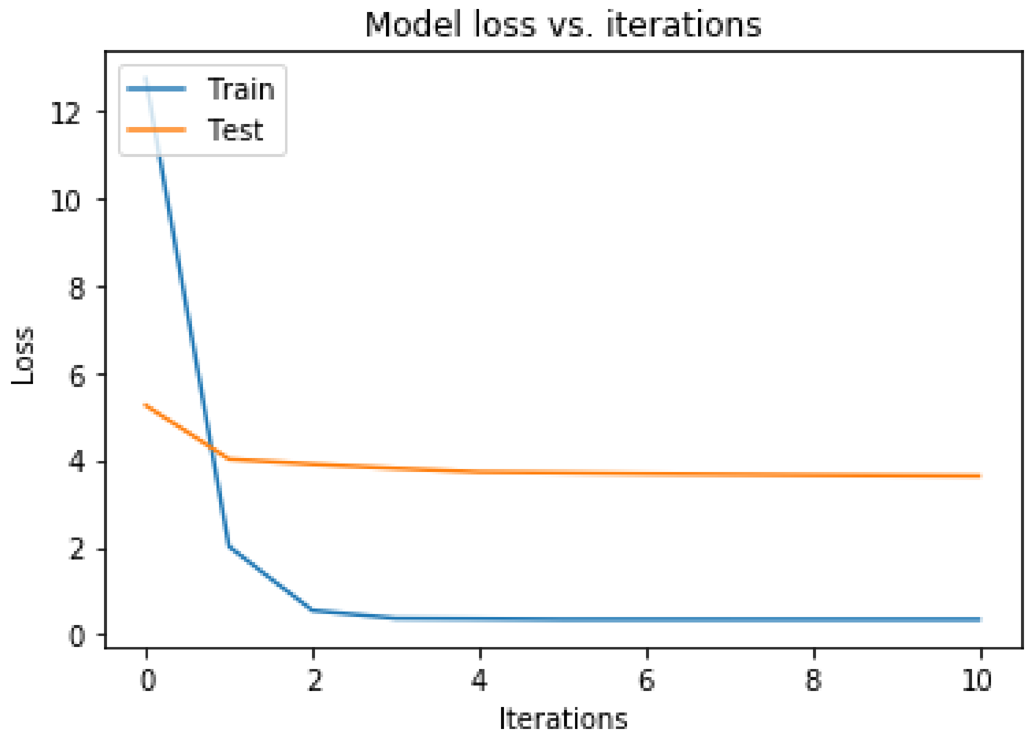


Figure 6: Comet (Caltech256): Loss plots when trained on 4 images per class

the distilled model by training it on a transfer set and using a soft target distribution for each case in the transfer set that is produced by using the cumbersome model with a high temperature in its softmax.
The same high temperature is used when training the distilled model, but after it has been trained it uses a temperature of 1. When the correct labels are known for all or some of the transfer set, this method can be significantly improved by also training the distilled model to produce the correct labels. One way to do this is to use the correct labels to modify the soft targets, but we found that a better way is to simply use a weighted average of two different objective functions. The first objective function is the cross entropy with the soft targets and this cross entropy is computed using the same high temperature in the softmax of the distilled model as was used for generating the soft targets from the cumbersome model. The second objective function is the cross entropy with the correct labels. This is computed using exactly the same logits in softmax of the distilled model but at a temperature of 1. We found that the best results were generally obtained by using a considerably lower weight on the second objective function. Since the magnitudes of the gradients produced by the soft targets scale as $1/T^2$ it is important to multiply them by T^2 when using both hard and soft targets. This ensures that the relative contributions of the hard and soft targets remain roughly unchanged if the temperature used for distillation is changed while experimenting with meta-parameters.

3.2 Methods

In this section, we read the Caltech256 images and prepared the dataset of image array and target labels. We then loaded the VGG16 Net and created a new model which took as its input, the input of the VGG16 model and its output as the output of the second last layer of the VGG16 Net. We then divided all the weights of the last softmax layer by T.
We created another model which took the output of the last model as its input and had softmax activation. After that we did a feed forward through both of these models and stored the weights.

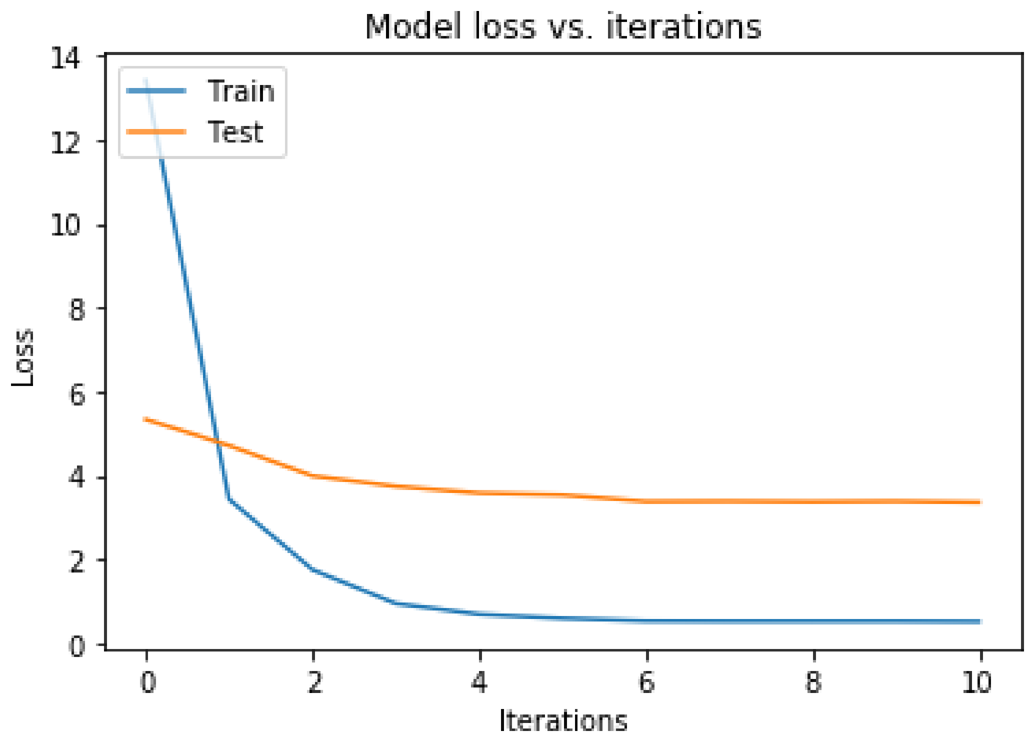


Figure 7: Comet (Caltech256): Loss plots when trained on 8 images per class

Add a Softmax output layer that takes input from the VGG softmax layer to predict Caltech256 classes.

We then trained only the last Softmax layer on 8 images in each category of Caltech256 images. To find the best value of T for this purpose, we tried different values of T : 1, 2, 4, 8, 16.

We tested the trained model on 8 images in each of the CalTech256 categories.

3.3 Results and Discussion

36 37 38 39

We found that the accuracy on the test set first increases when the temperature parameter increases from 1 to 2. It then reaches the maximum value when the Temperature parameter is 2. The accuracy then consistently falls with the increase in the value of the temperature parameter T. The accuracy was 49% with temperature parameter as 1, 53.4% with temperature parameter T as 2. Then the accuracy falls to 44.6% at T = 4, 41.6% at T=8 and 29.7% at Temperature T = 16.

We found that the distillation model as mentioned in the paper of Geoff Hinston is quite suitable for transferring the knowledge from a more cumbersome model to the model more relevant for the task at hand. As in this case the VGG16 model, the output classes were 1000, but in the CalTech image set the classes is a small subset of 256 categories. Hence, by adding the temperature parameter, we are able to transfer better knowledge to the second model. But as the temperature is enhanced, all the classes probability is more and more similar and hence the accuracy subsides.

4 Learning and Results

In this assignment, we worked on practical implementation of Image classification with Convolution Neural Networks. We learned how a deep CNN - VGG16 is modeled, coded and designed in Keras

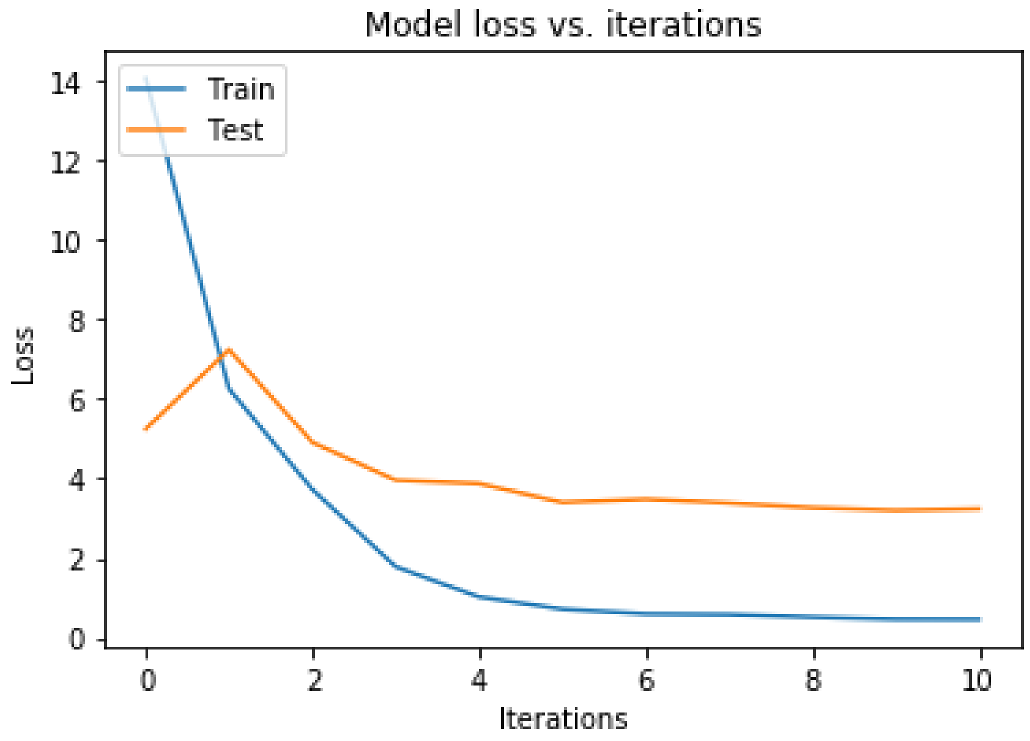


Figure 8: Comet (Caltech256): Loss plots when trained on 16 images per class

library. We also experimented with Keras libraries and the various utilities to read data, predict output and train a custom made user model. We found this assignment to be extremely helpful as we got acquainted to CNNs and layered structure of the Keras environment.

We also studied the concept of 'Transfer learning', which uses the powerful pre-trained weights of ImageNet to train a new model with a new dataset. We learned how efficiently this concept can be used and applied on a new dataset to achieve fairly good accuracies for image classification on a new dataset. We also learned how the intermediate filters could be visualized and used to study the learned features of the image.

Following this, we tried taping the value of convolution layer at different levels on VCG16 model to see how the accuracy varies at different layers of a pre-trained model. Finally, using a temperature parameter in the penultimate layer, we mapped the 1000 categories of VGG16 trained on ImageNet to 257 classes of Caltech and found how temperature parameter can be used to tune the softness of a classification method.

Results of this assignment matched our intuitions and theoretical estimation. With increasing the number of images for each class during the model training, we found that the accuracy on a fixed validation set increases. Thus, indicating that the model becoming more powerful. We were able to beat state of the art models of results obtained by Bo et al. (Bo et al., 2013) by a significant margin. This also tells how powerful transfer learning can be, if we have an already trained model. Also, filter visualization of an image at each layer is a fairly good representation of the image, and it distorts as we go deeper into the network. As expected, the feature extraction at different levels of a deep convolutional network gives different accuracies and learning abilities. Closer the layer is to the input layer, lesser is the accuracy predicted using the ouput of this layer. Temperature parameter also provided and interesting insight. While increasing the temperature parameter (T), we could tune the softmax activation and as this T increases, the accuracies go down indicating an increasing uniform(or random) classification

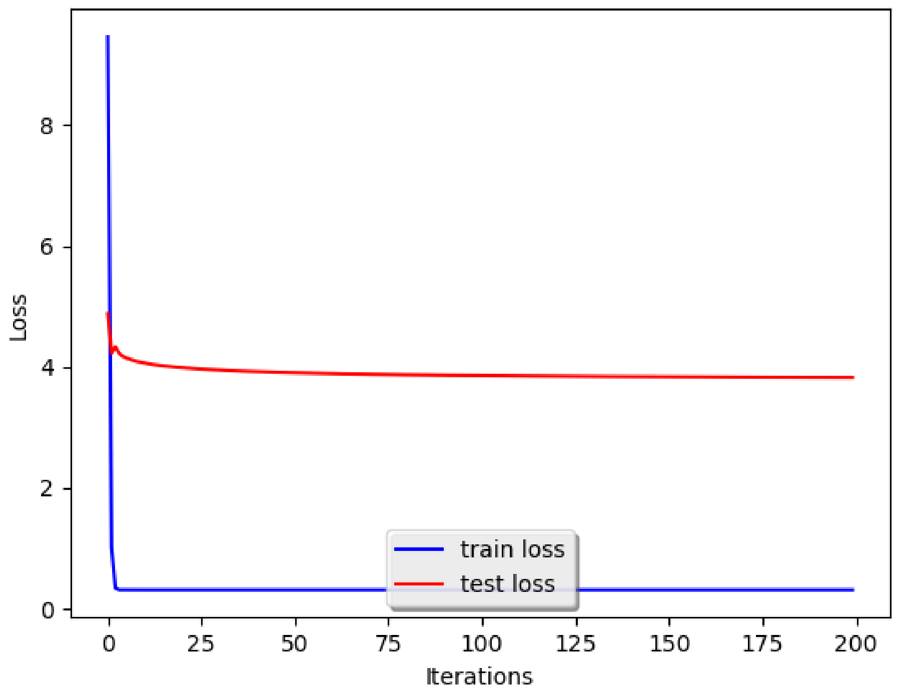


Figure 9: Approach 2 (Caltech256): Loss vs iterations for 2 training samples per class

Table 1: Contribution by each team member

S.no.	Name	Contribution
1	Swapnil Taneja	Question 1 implementation and code design and python scripts for image read and segregation
2	Saksham Sharma	Question 2 implementation, plotting graphs and visualization of filters
3	Kriti Aggarwal	Question 1 implementation, feature extraction, and code optimization
4	Prahal Arora	Question 2 implementation, feature extraction, code optimization
5	Saicharan Duppati	Question 3 implementation , plotting graphs and visualization of filters

5 Contributions

We made sure that most of the work is equally distributed and shared our individual contributions for the team assignment, so that each of us understood the latters work. Following are roughly the contributions made (see Table 1.):

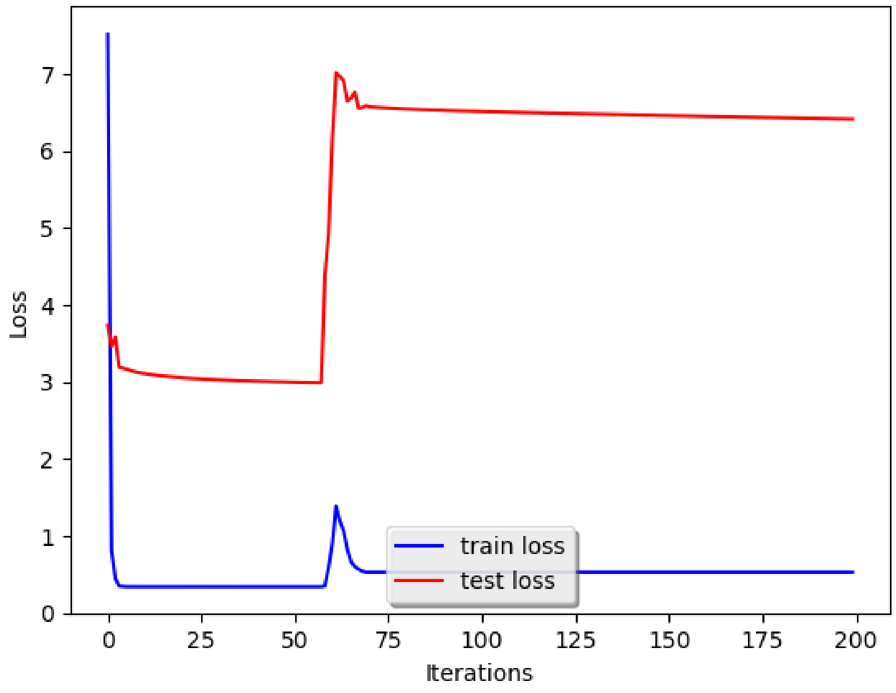


Figure 10: Approach 2 (Caltech256): Loss vs iterations for 4 training samples per class

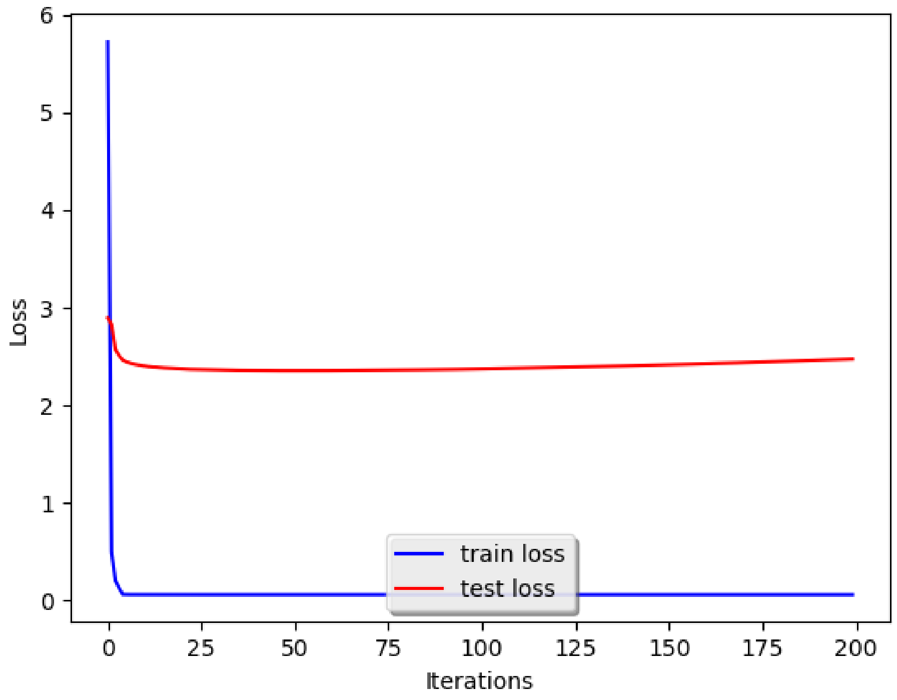


Figure 11: Approach 2 (Caltech256): Loss vs iterations for 8 training samples per class

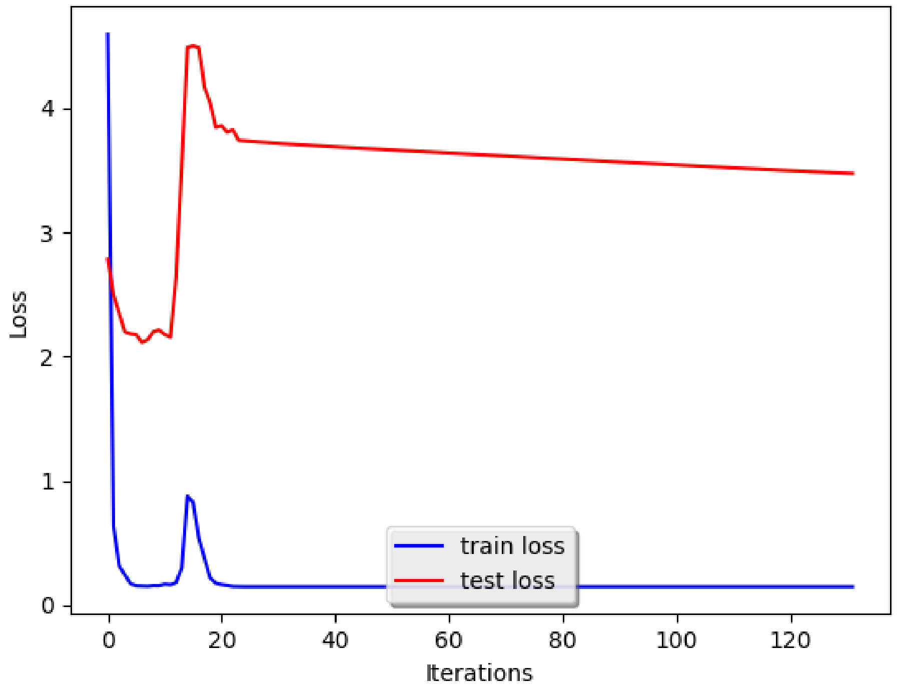


Figure 12: Approach 2 (Caltech256): Loss vs iterations for 16 training samples per class

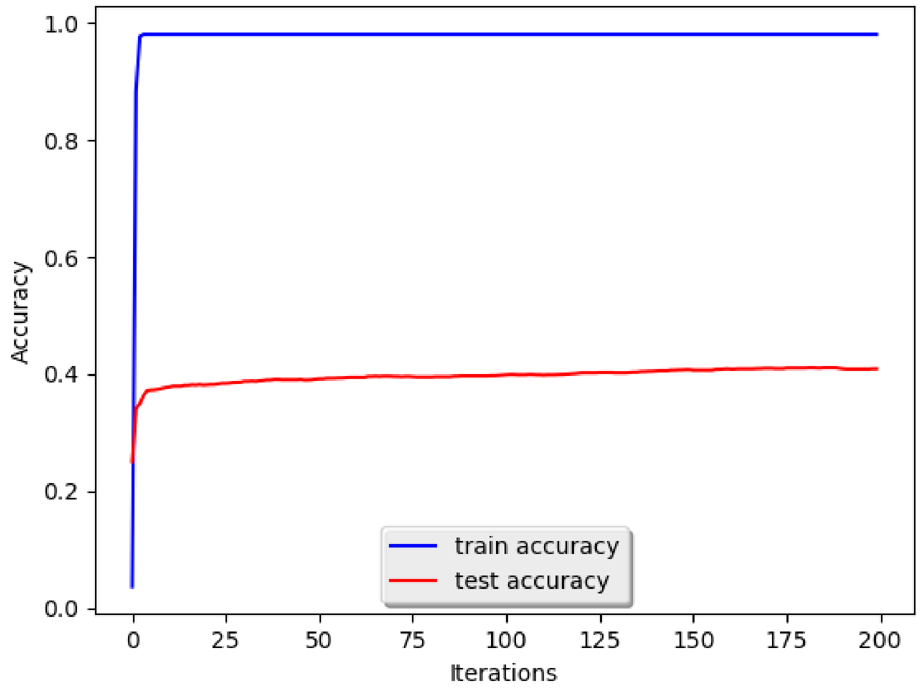


Figure 13: Approach 2 (Caltech256): Accuracy vs iterations for 2 training samples per class

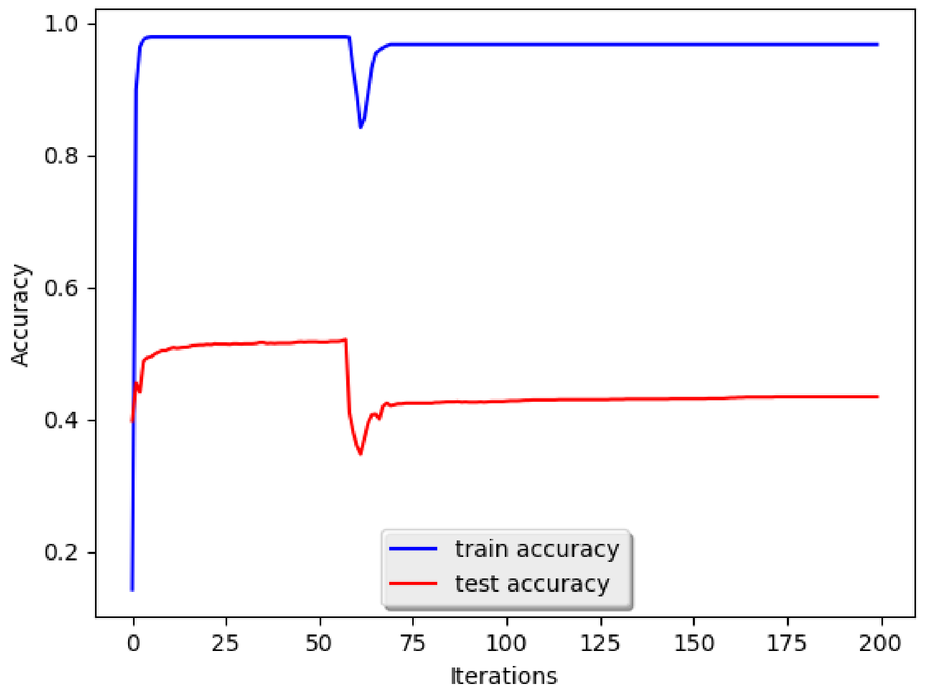


Figure 14: Approach 2 (Caltech256): Accuracy vs iterations for 4 training samples per class

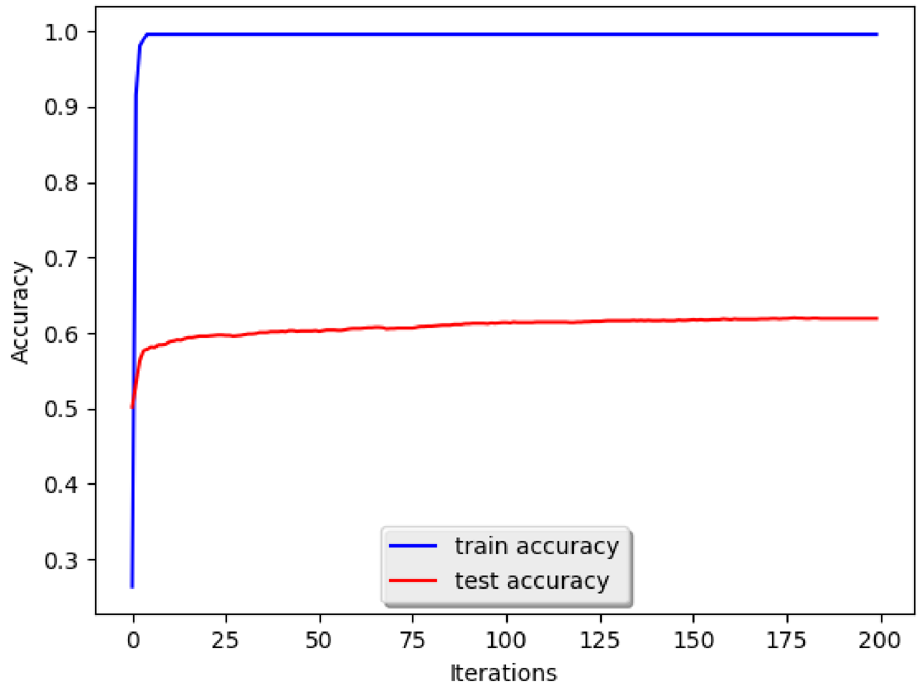


Figure 15: Approach 2 (Caltech256): Accuracy vs iterations for 8 training samples per class

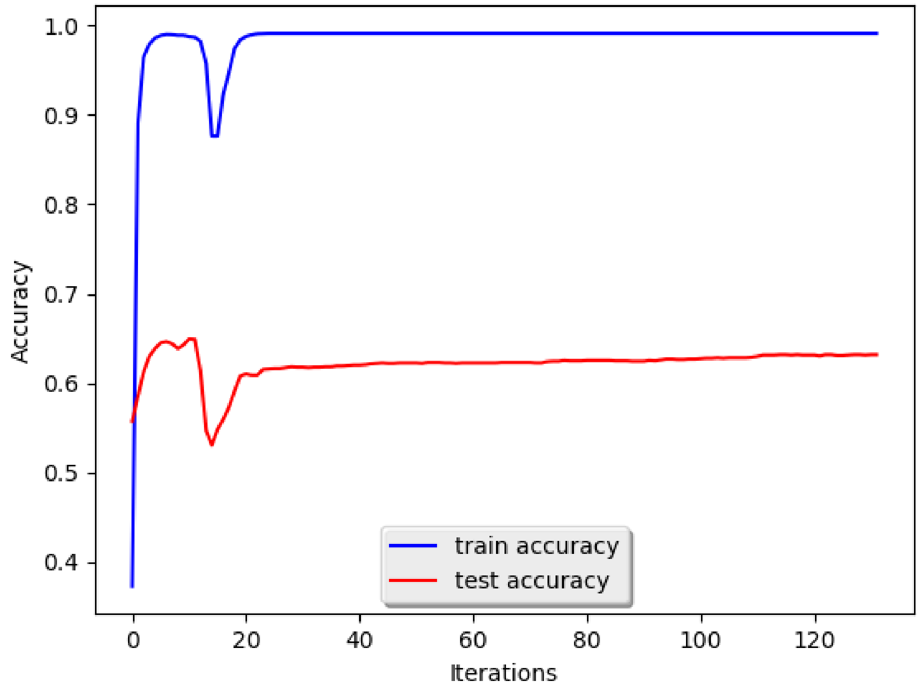


Figure 16: Approach 2 (Caltech256): Accuracy vs iterations for 16 training samples per class

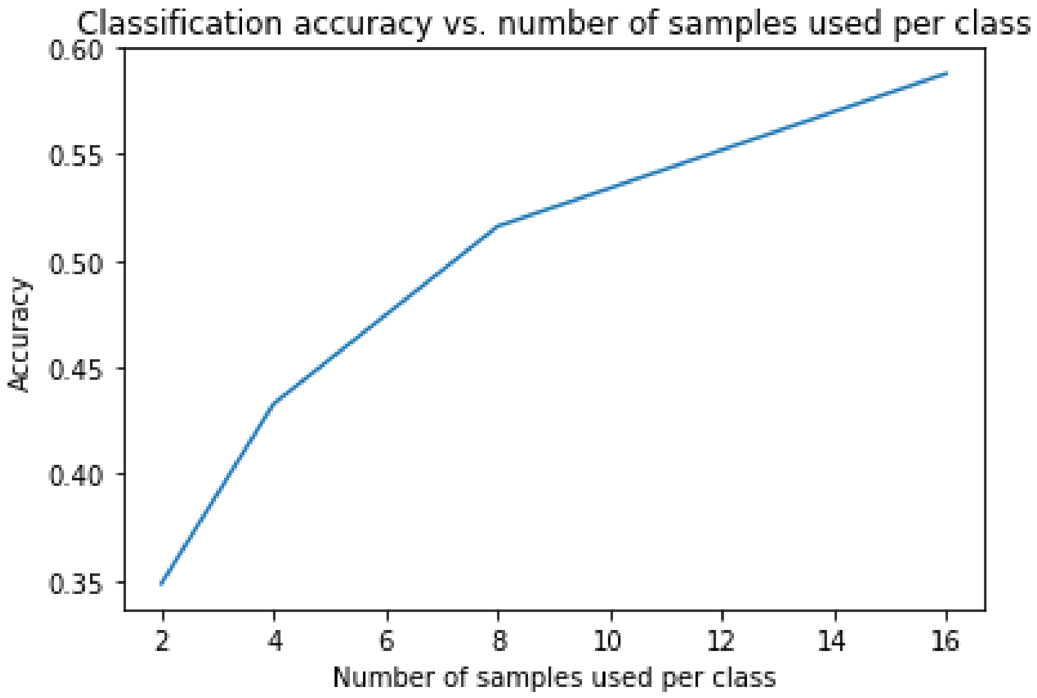


Figure 17: Comet (Caltech256): Accuracy vs Training Samples per class.

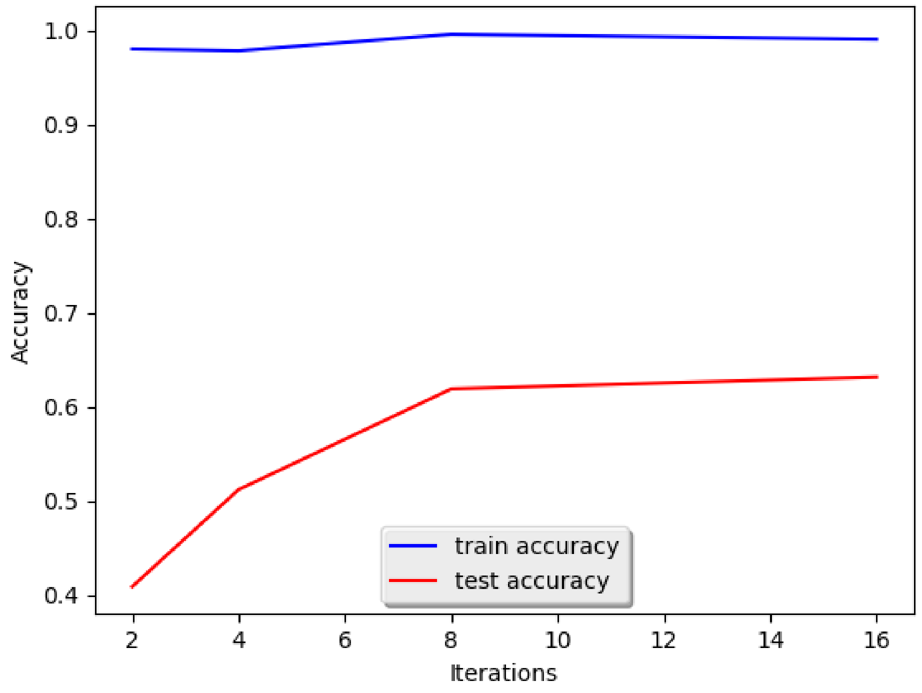


Figure 18: Approach 2 (Caltech256): Accuracy vs Training Samples per class.

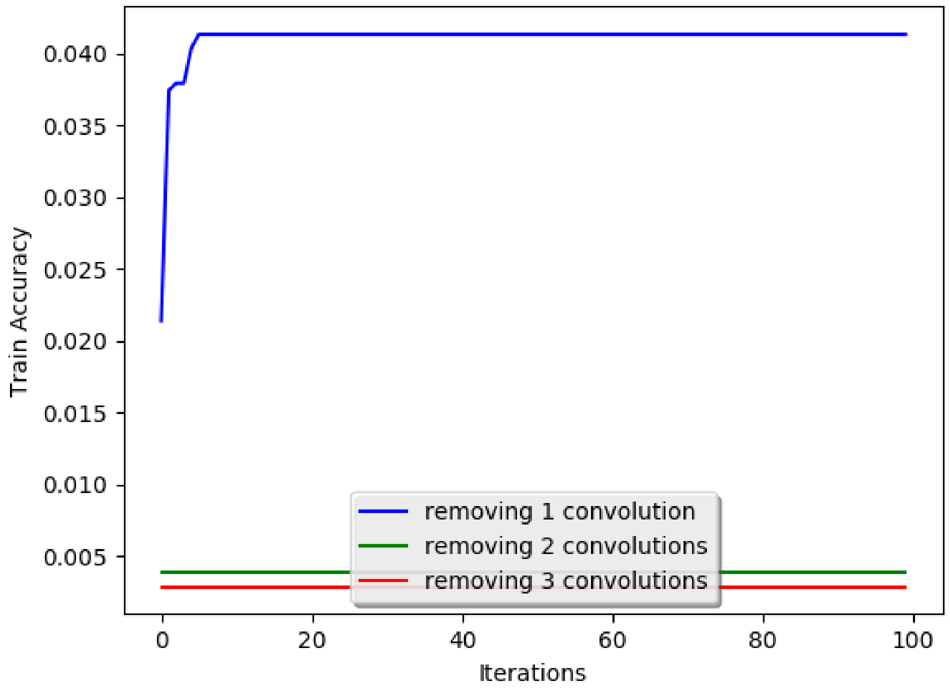


Figure 19: (Caltech256): Intermediate convolution Layers : Train Accuracy vs Iterations

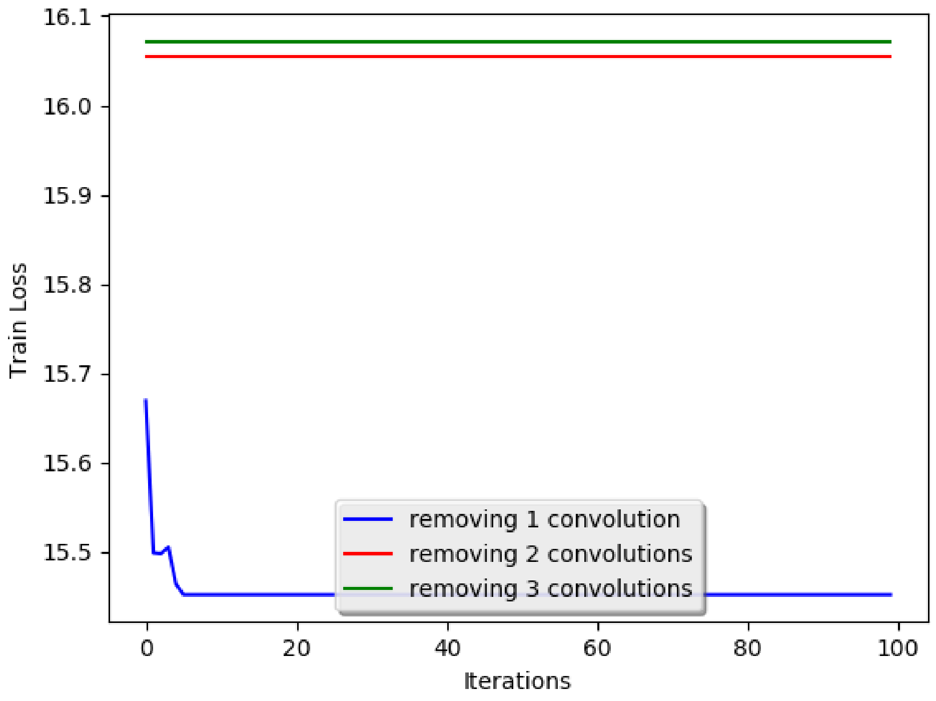


Figure 20: (Caltech256):Intermediate convolution Layers : Train Loss vs Iterations.

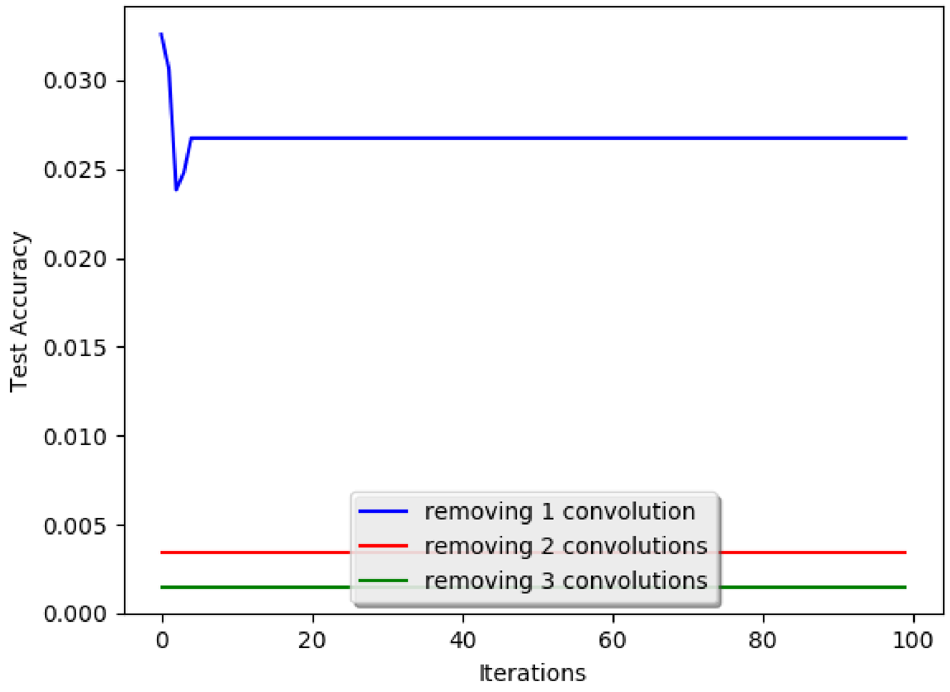


Figure 21: (Caltech256): Intermediate convolution Layers : Test Accuracy vs Iterations.

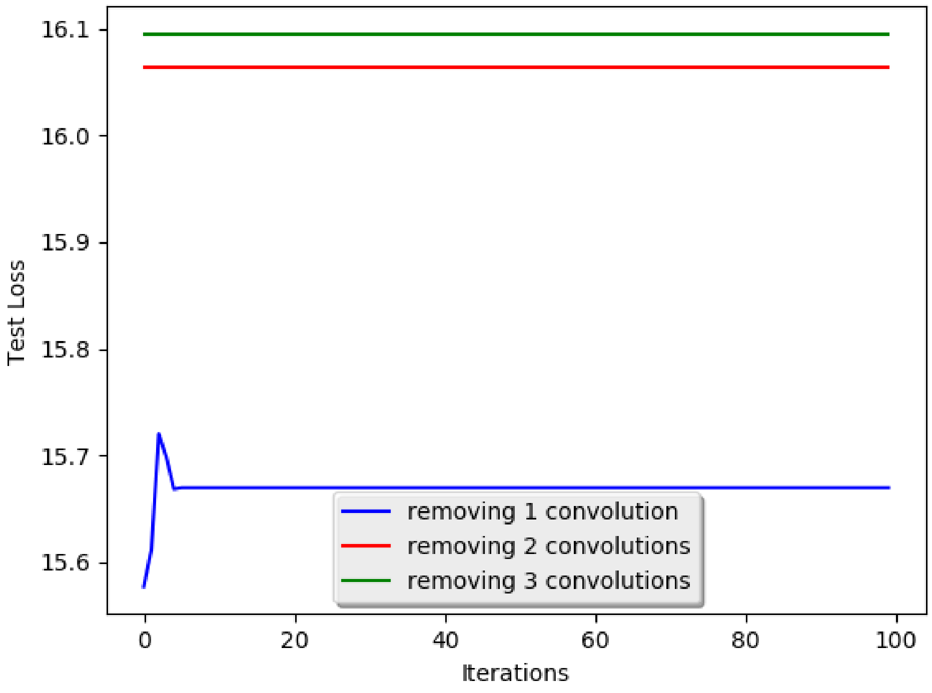


Figure 22: (Caltech256): Intermediate convolution Layers : Test Loss vs Iterations.

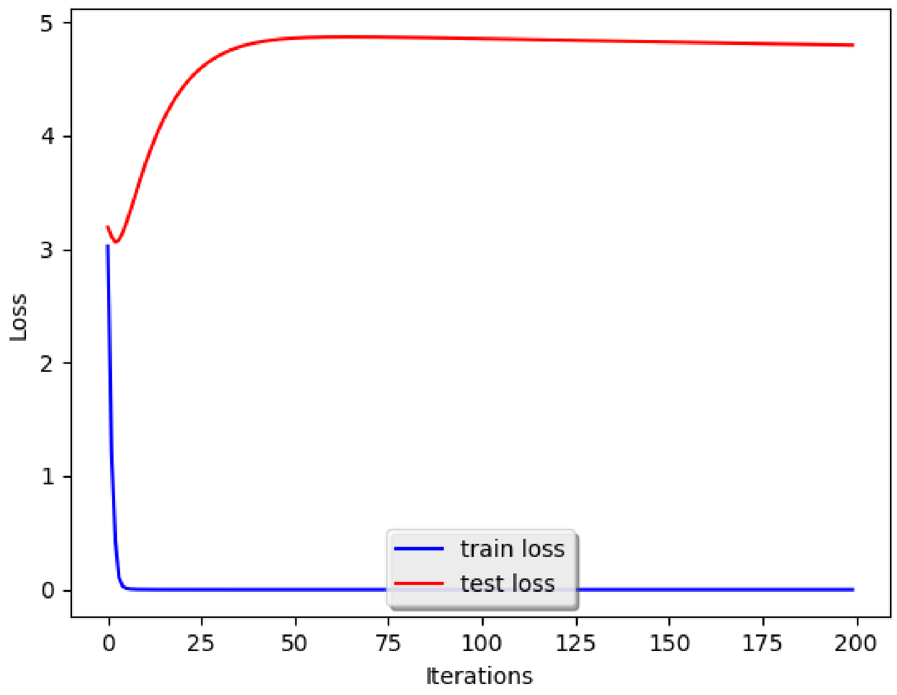


Figure 23: Train and test loss vs iterations for 2 training samples per class

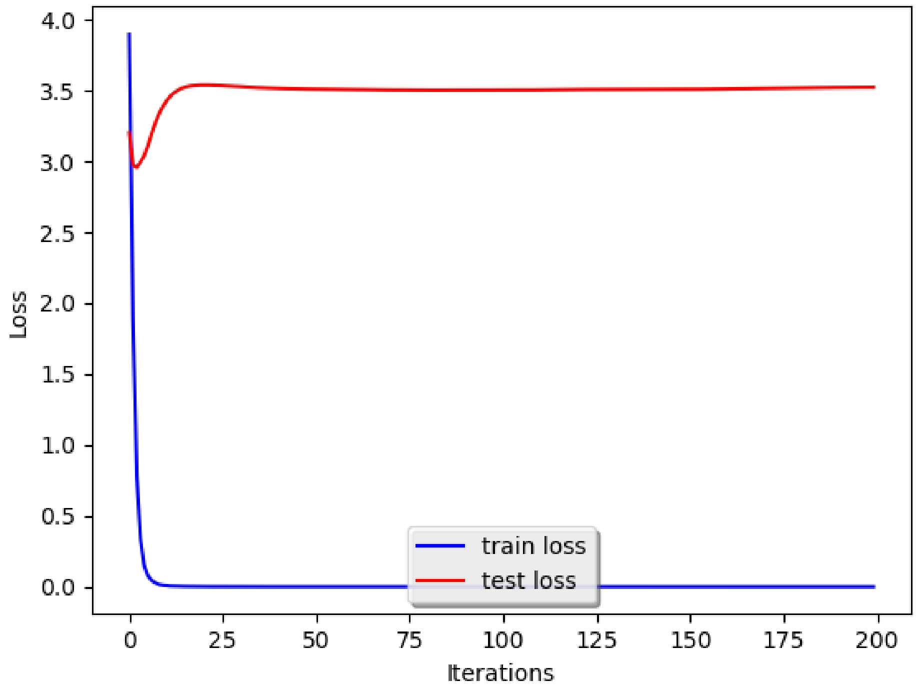


Figure 24: Train and test loss vs iterations for 4 training samples per class

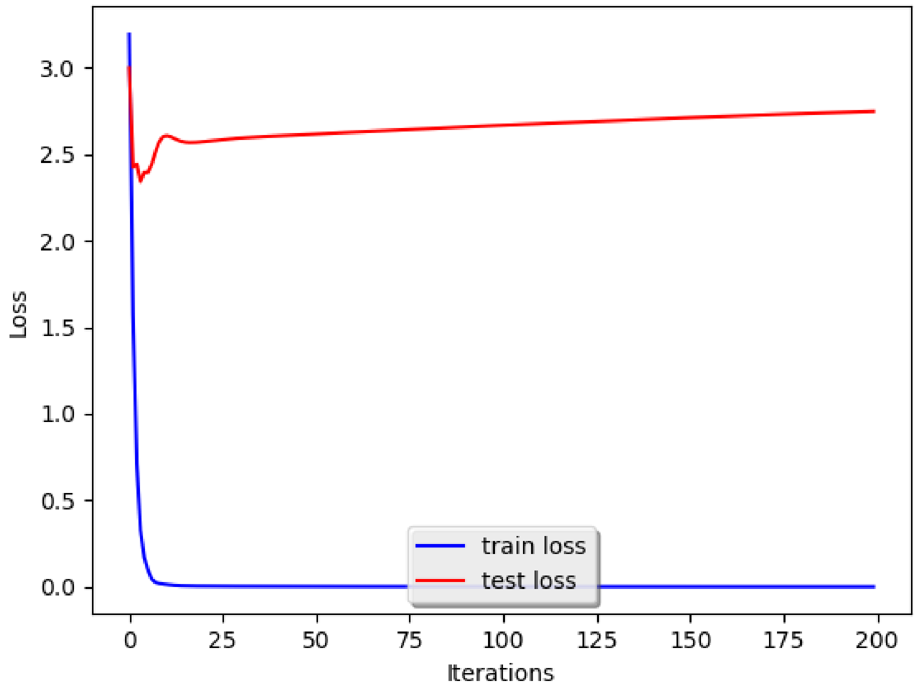


Figure 25: Train and test loss vs iterations for 8 training samples per class

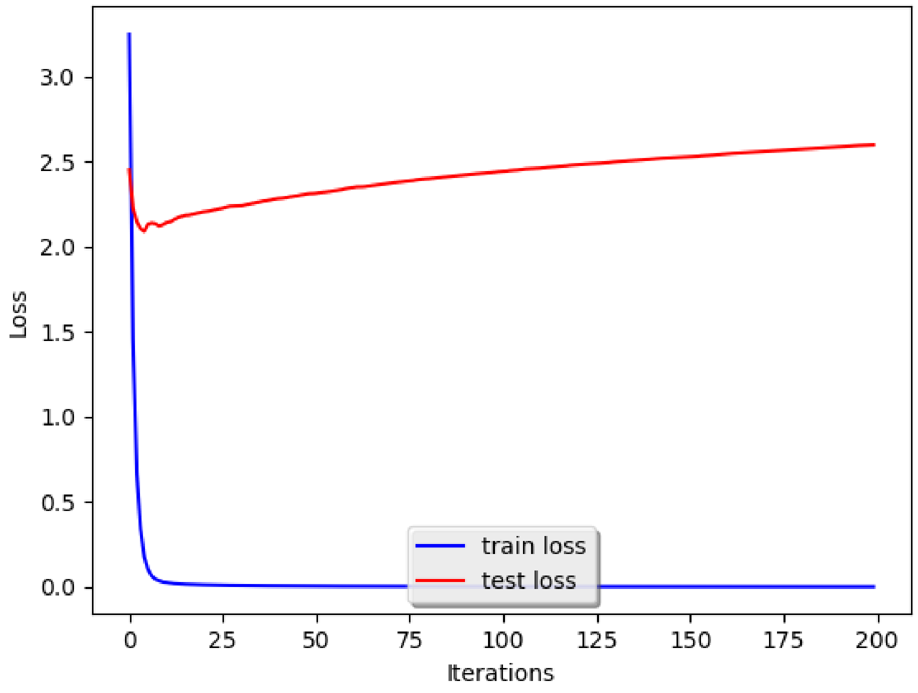


Figure 26: Train and test loss vs iterations for 16 training samples per class

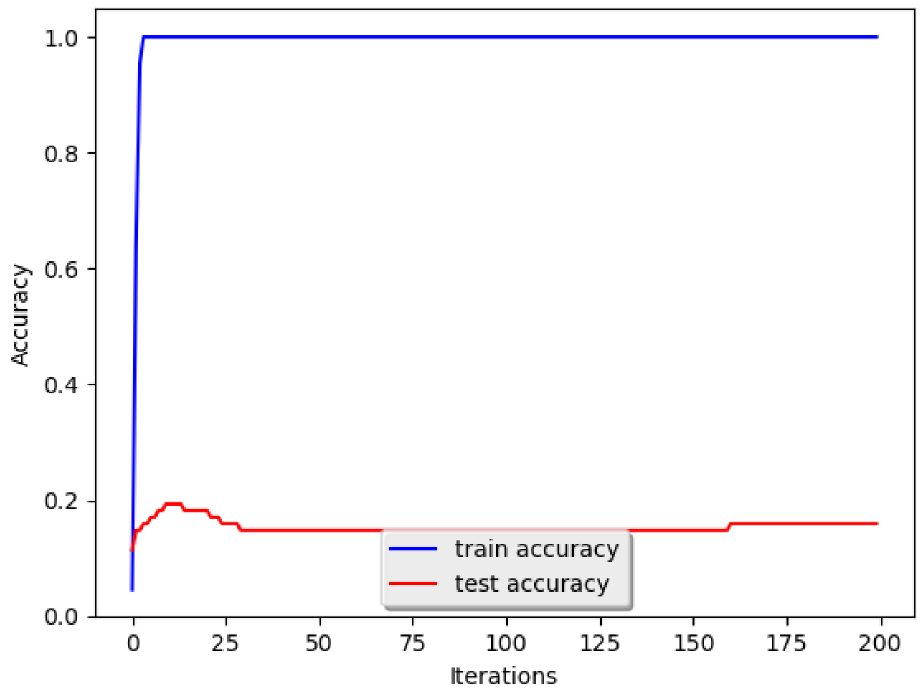


Figure 27: Accuracy vs iterations for 2 training samples per class

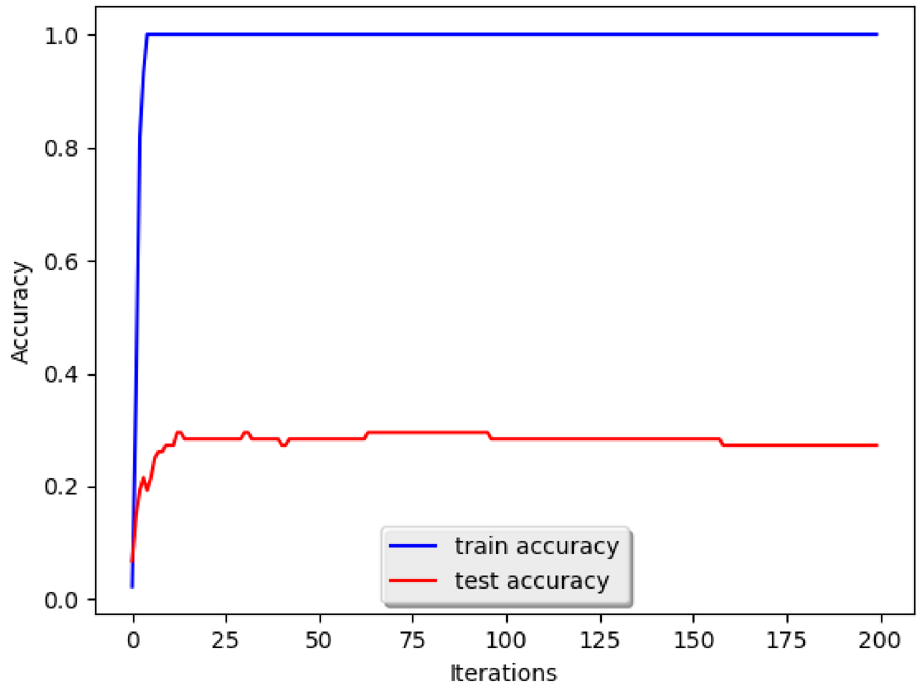


Figure 28: Accuracy vs iterations for 4 training samples per class

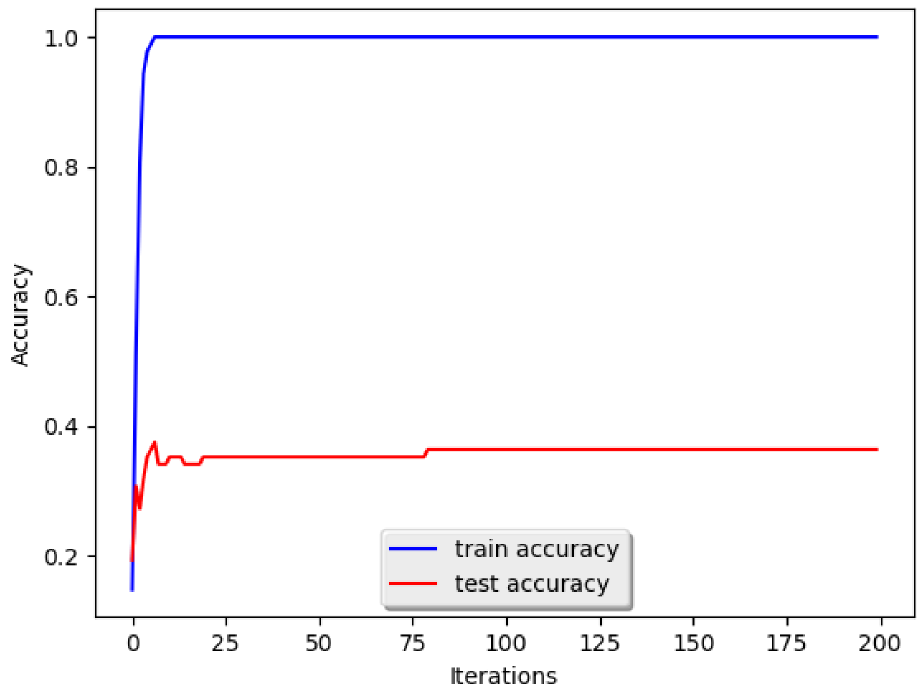


Figure 29: Accuracy vs iterations for 8 training samples per class

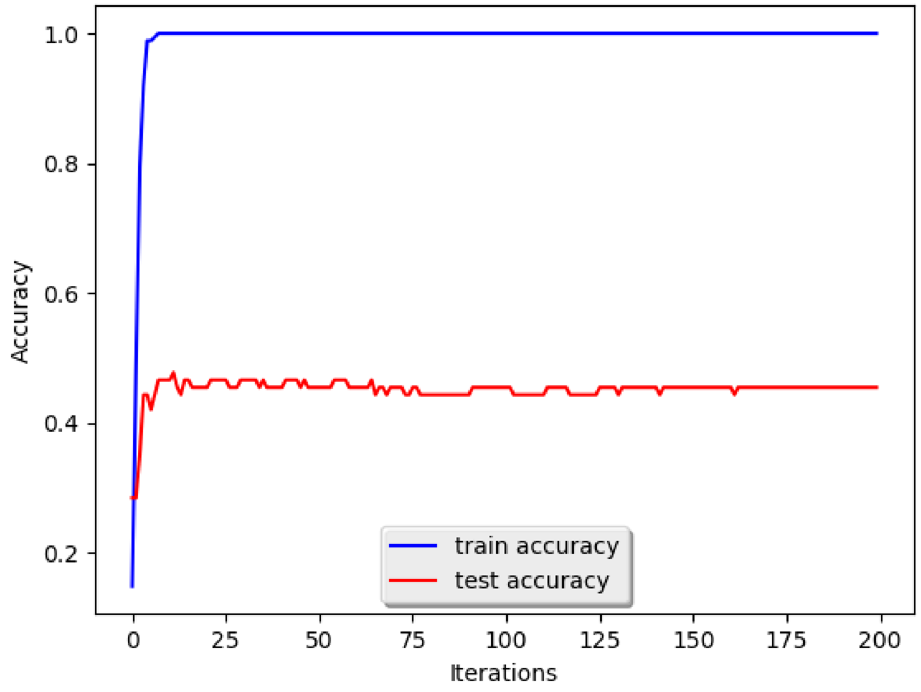


Figure 30: Accuracy vs iterations for 16 training samples per class

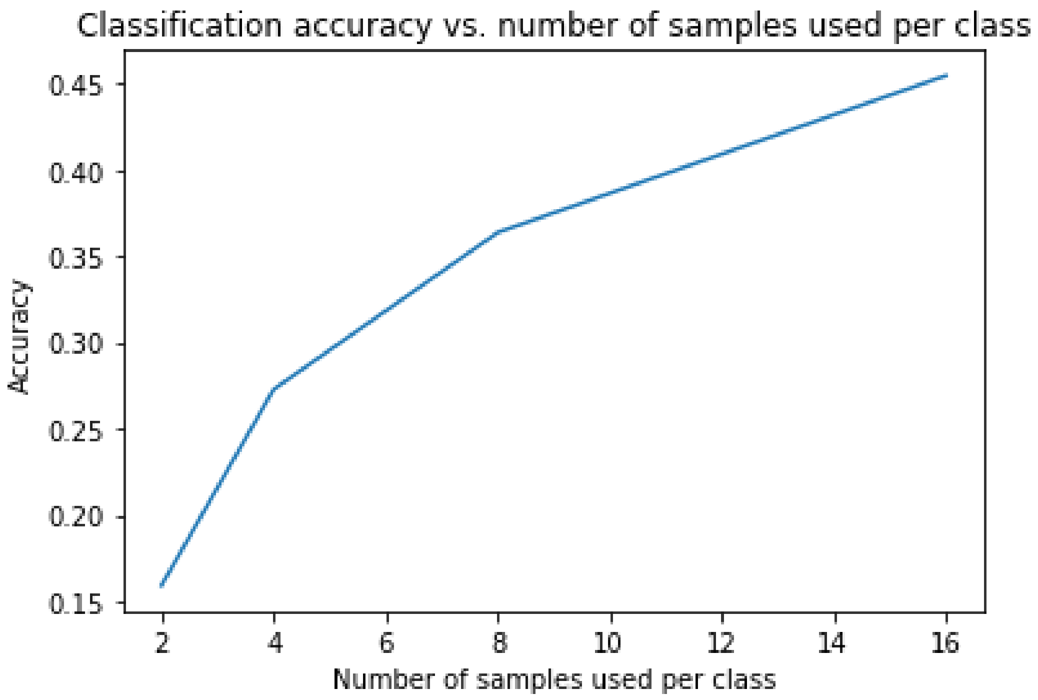


Figure 31: Classification accuracy vs. number of samples used per class.

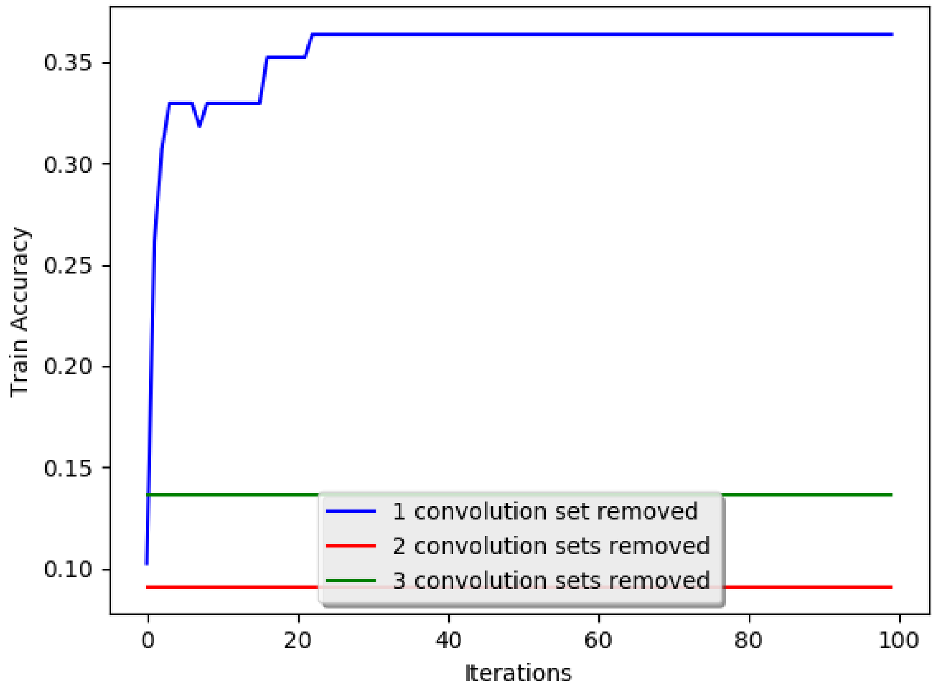


Figure 32: (Urban Tribes): Intermediate convolution Layers : Train Accuracy vs Iterations

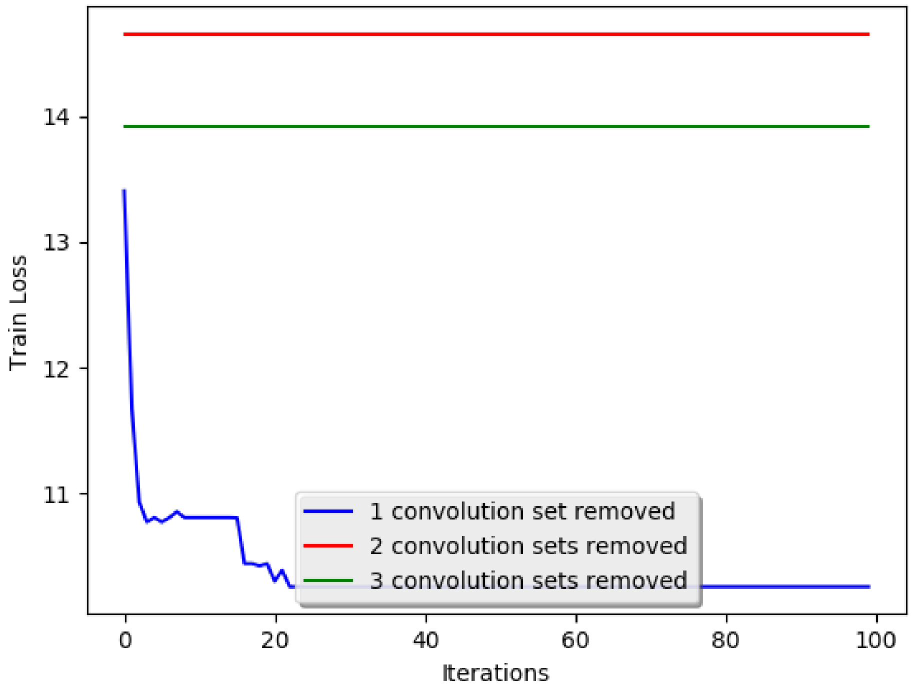


Figure 33: (Urban Tribes):Intermediate convolution Layers : Train Loss vs Iterations.

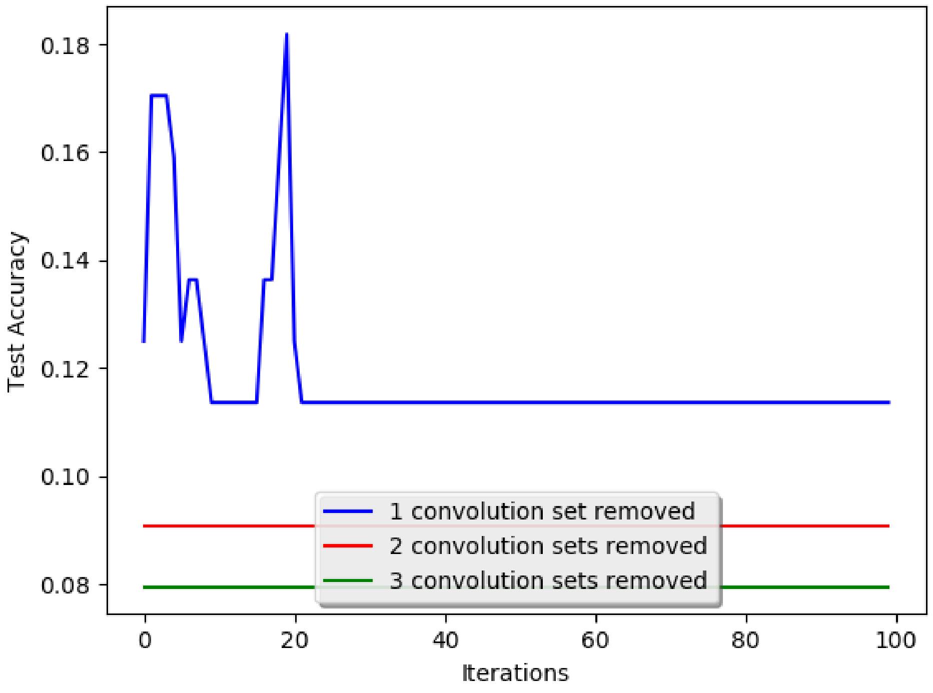


Figure 34: (Urban Tribes): Intermediate convolution Layers : Test Accuracy vs Iterations.

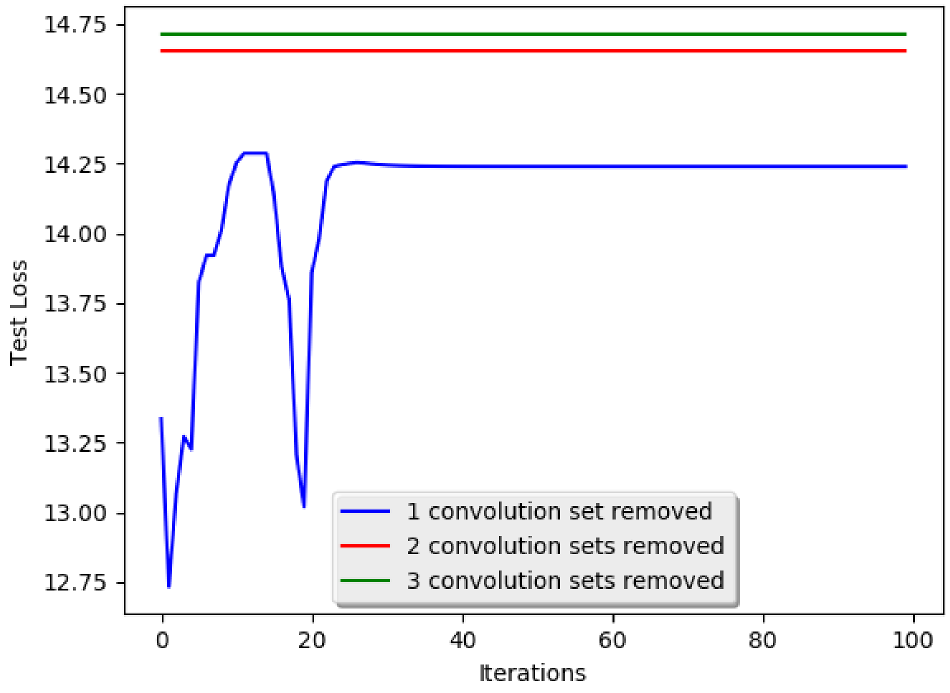


Figure 35: (Urban Tribes): Intermediate convolution Layers : Test Loss vs Iterations.

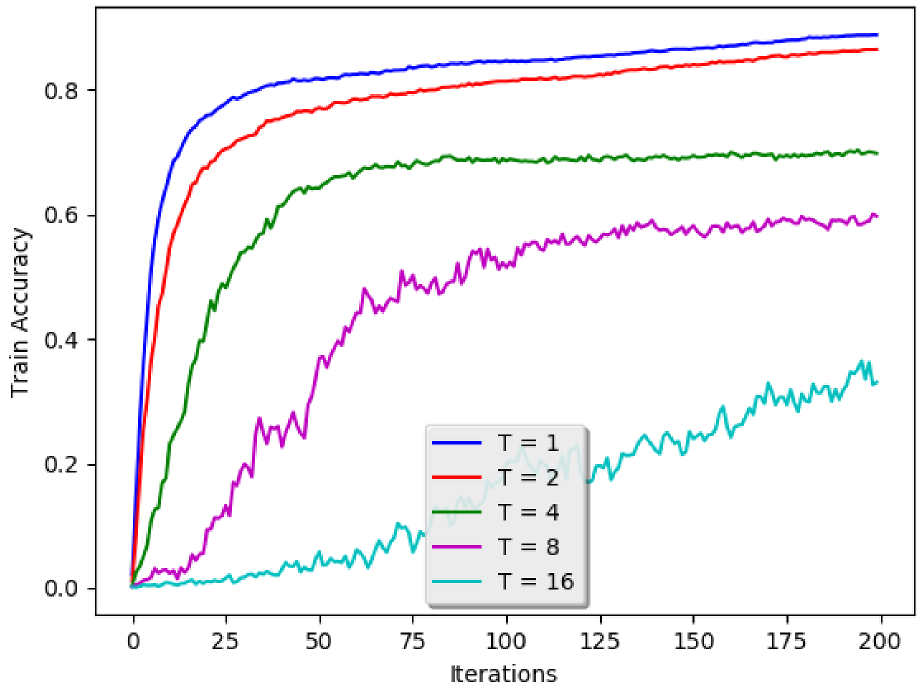


Figure 36: Train (Caltech256): Accuracy plots when trained on 2 images per class with different values of temperature T

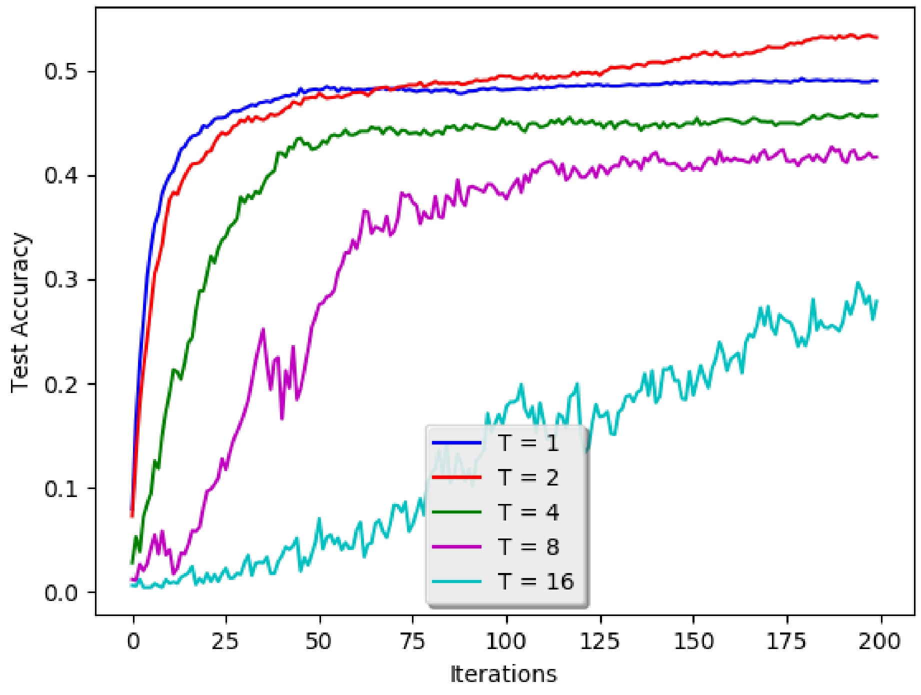


Figure 37: Test (Caltech256): Accuracy plots when trained on 2 images per class with different values of temperature T

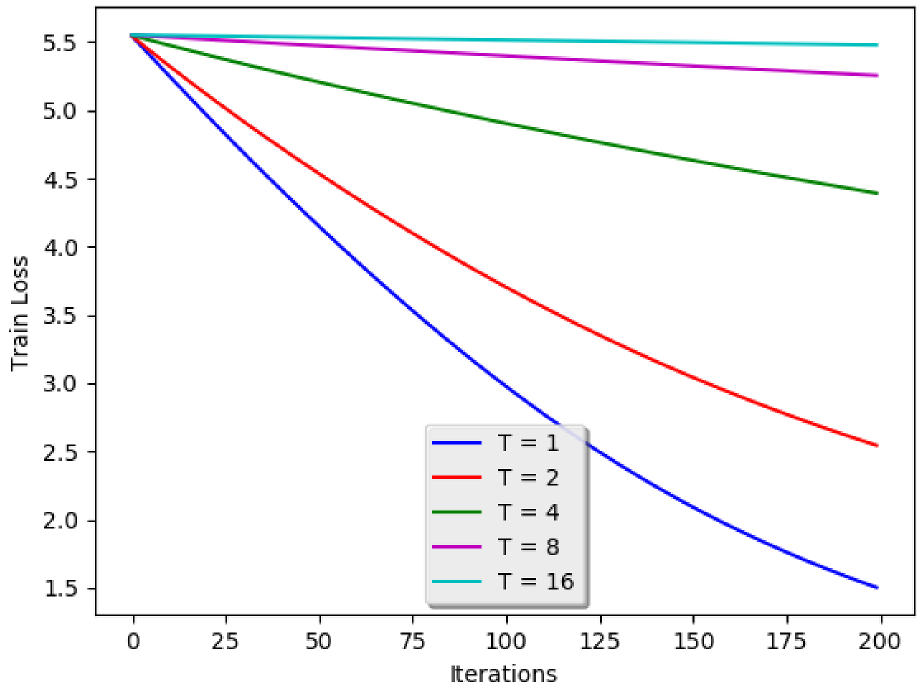


Figure 38: Train (Caltech256): Accuracy plots when trained on 2 images per class with different values of temperature T

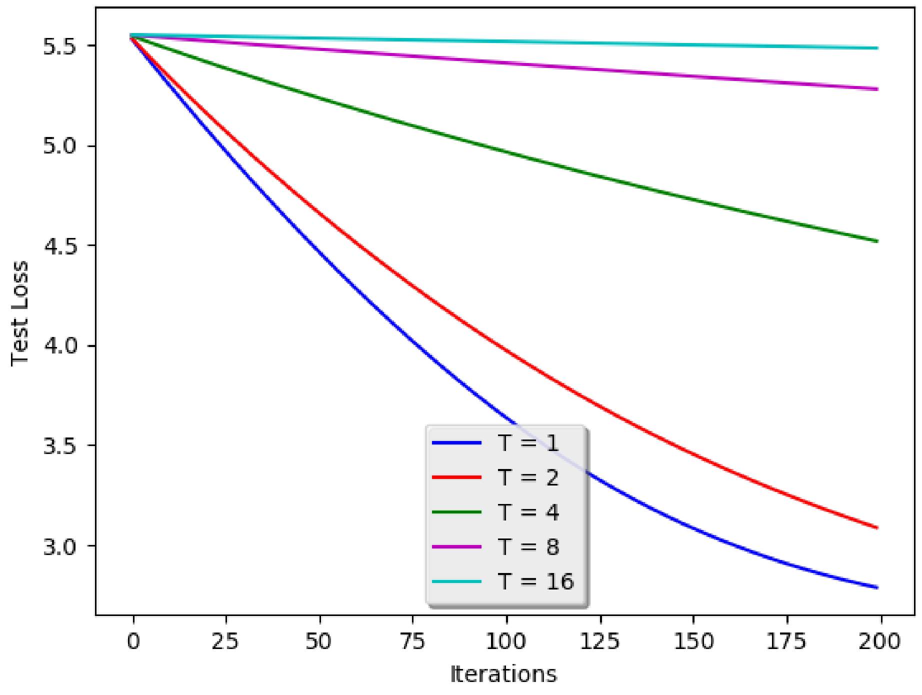


Figure 39: Test (Caltech256): Accuracy plots when trained on 2 images per class with different values of temperature T