An Exploration of Multi-layer Fully-connected Neural Network

Tian Li
CSE Department
University of California, San Diego
La Jolla, CA 92093
til080@eng.ucsd.edu

Zhipeng Yan
CSE Department
University of California, San Diego
La Jolla, CA 92093
zhy136@eng.ucsd.edu

Abstract

In this assignment, we apply variants of transfer learning for the task of image Classification, and analyze the performance of the outcoming models. By creating and tuning our own softmax layer, we firstly applied the VGG16 model to classify the Caltech 256 and Urban Tribes dataset. And then based on the idea of using similarity between datasets, we applied temperature-based Softmax regression to classify Caltech 256 again.

1 Introduction

Transfer Learning aims at solving new tasks using knowledge gained from solving related tasks. Due to the huge number of parameters, deep architectures of convolutional neural networks (ConvNet) cannot be trained on datasets of incomparable size. Thus it is common in practice to use existing models which are trained on very large scale datasets as feature extractors or initial layers of a ConvNet that can be fine-tuned to learn the underlying model of the dataset from new task.

VGG 16 is a very deep architecture of ConvNet. It consists of 16 trainable layers, including 13 convolutional layers and 3 fully connected ones. It achieved 92.6% accuracy rates on ImageNet dataset, and can also generalize well to other works such as the object detection of VOC-2007 dataset.

This report contains 5 sections. In Section 2 we will show the results and inferences of the performance of the VGG 16 model on CalTech 256 dataset. Section 3 is about the Urban Tribes dataset. And in section 4, we will directly take advantage of the original VGG 16 model, by introducing the temperature and appending a softmax layer on top. Finally, section 5 is the summary of what we have learned in this work, and the contribution of each team member.

2 Caltech256 Classification

In this section, we take advantage of pre-trained VGG 16 ConvNet model, which is trained on ImageNet, and transfer it to the classification of CalTech 256 dataset. To better understand and analyze the performance of the transferred model, we conduct 2 experiments. For the first one, we replace the last softmax layer of VGG 16 model by another softmax layer which predicts the label of CalTech 256 dataset, and record the results with respect to the number of input samples for each class. Thus we can get the relation between performance and the number of samples. And for the other one, the input constantly contains 16 samples for each class, and we put the softmax layer after different parts of the VGG model. Consequently, we can compare the representative capability

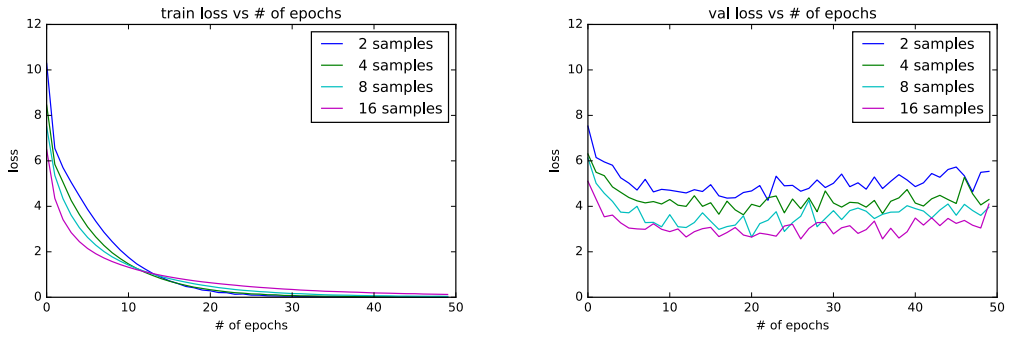
of different layers of a ConvNet.

To accelerate the training process, we used GPU instead of CPU to do the computation works.

2.1 Performance vs. Training Samples

Firstly, it is clearly shown by figure 1(a) and 1(b) that the network converges very fast, and works well on training set. On the other hand, it shows relatively poor performance on the validation set, which indicates the lack of the ability of generalization.

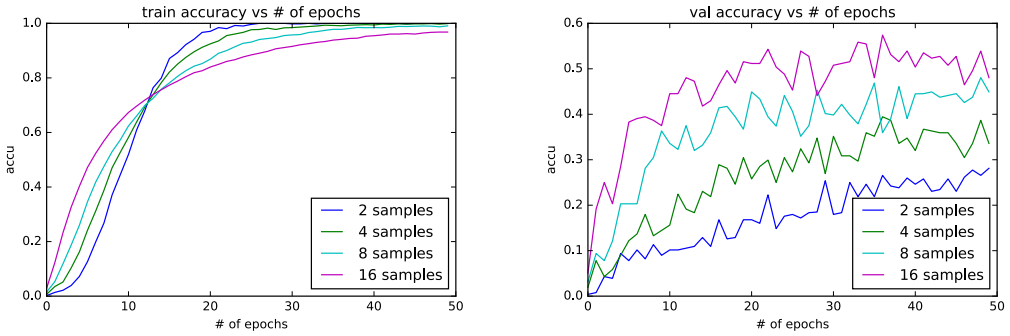
But we should treat this problem in a different perspective. Firstly, the model can achieve more than 95% accuracy on the training set after learning, which shows that the output of the last 'frozen' layer can almost be seen as linearly separable representations of the input samples for classification work, despite the fact that all these representations are learned from another dataset.



(a) (b)

Figure 1: Loss of (a). training set (b). validation set throughout the training

In addition, the number of trainable parameters of our model is $4096 \times 256 + 256 = 104,8832$. And for the simplest case, 512 samples (2 for each class) are used to train the network, indicating that there are at most $512 \times 4096 = 209,7152$ d.o.f in training data, which is only twice as much as the number of parameters, since the dimension of the output of the last 'frozen' layer is 4096. However, the model can finally get almost 30% accuracy on validation set, even if all the samples in it are new to the model. As a comparison, a purely random choice has the accuracy rate $1/256 \approx 0.004$. As a result, we concludes that our model shows the ability of generalization, thanks to the pre-trained representation of raw input.



(a) (b)

Figure 2: Accuracy of (a). training set (b). validation set throughout the training

With more samples used to train the model, as shown in figure 3, the training accuracy decreases, and the validation accuracy increases. The training accuracy drops simply because there are more

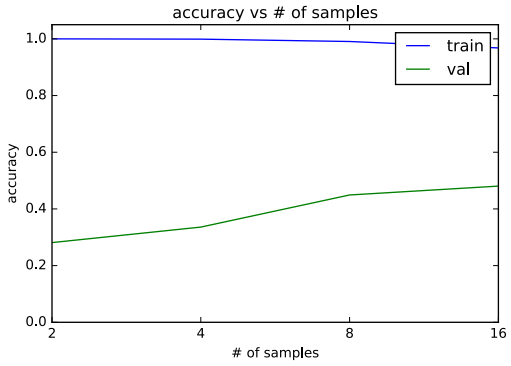
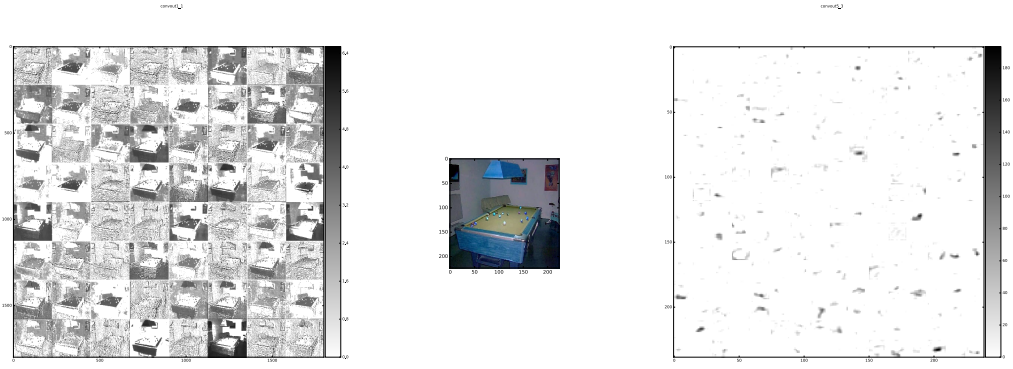


Figure 3: Accuracy vs. number of samples

linearly inseparable outliers with doubled or even more samples. And the rising validation accuracy, on the other hand, reveals that our model is able to work with the new dataset, as long as it 'meets' sufficient amount of samples.

Figure 4(a) shows the activation of the first convolutional layer of input image figure 4(b). And all of them can be seen as some manually extracted features from the input. And things are different in figure 4(c), which is the activation of the last convolutional layer given the same input. Here, each subfigure corresponds to a certain pattern of the input image, but not recognizable anymore. Considering the performance of the model, we should regard them as some high level features.



(a) Convolutional filter 1 (b) Original input (c) Convolutional filter 5

Figure 4: Activation of convolutional layer

2.2 Feature Extraction

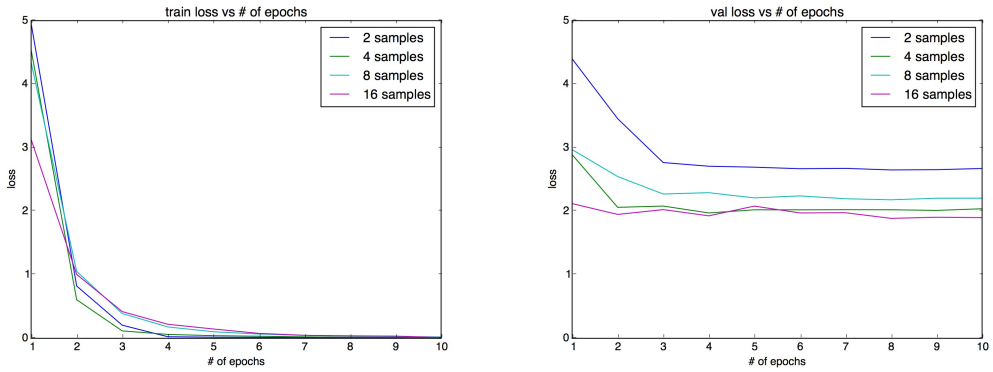
To show how different level of features affects the performance, we choose the output of 13^{th} and 10^{th} convolutional layer as the input feature of the final softmax classifier. Figure 6 and 7 show the corresponding curves. Apparently, features extracted by deeper layer can bring higher accuracy and smaller loss, generalize better to the transfer work, and thus are more meaningful representations of the original input. And it also reveals the great power of deep ConvNet.

3 Urban Tribe Classification

There are 11 classes in this dataset. We trained the model with 2,4,8,16 samples per category, using $batch\_size = 22$ and $epoch = 10$.

3.1 Inference

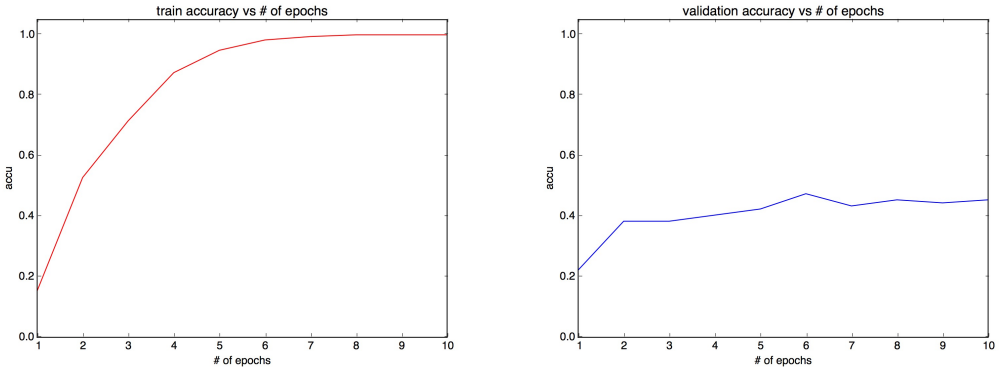
(a). Plot train and test loss vs. iterations



(a) (b)

Figure 5: Loss of (a). training set (b). validation set throughout the training

(b). Plot train and test accuracy vs. iterations
Here, both plots are generated by using 16 samples per category (c). Plot classification accuracy



(a) (b)

Figure 6: Accuracy of (a). training set (b). validation set throughout the training

vs. number of samples used per class

(d). Discuss the plots and report your inference from them
The baseline of error should be $log(11) \approx 2.397895$, obtained by completely random model. The training error and validation error decrease as the number of samples per category increases. The minimum training error is 0.0127 and the minimum validation error is 1.8999. Training errors decreases rapidly in each case and the validation error is much larger than training error, thus overfitting may exist.
The table shows the minimum error in each case.

No. of Samples	2	4	8	16
Training	0.0068	0.0115	0.0216	0.0127
Validation	2.6732	2.0397	2.2062	1.8999

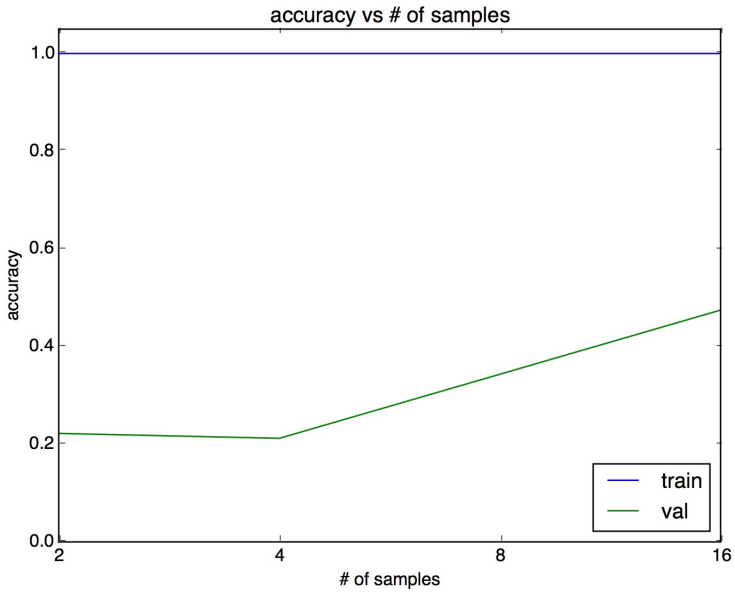
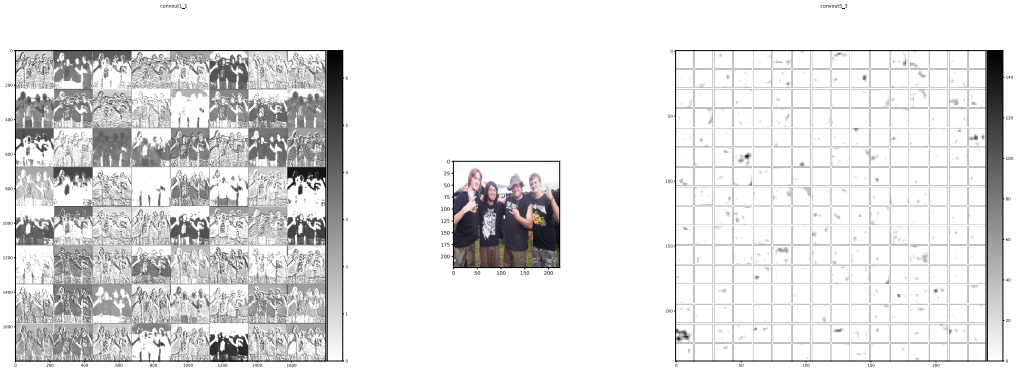


Figure 7: accuracy vs. number of samples

For the accuracy plots, we can see that as train accuracy increases rapidly during training, validation accuracy does not improve much after 2 iterations, which may also be caused by overfitting. Validation accuracy increases as number of samples increases, thus to achieve higher accuracy, we should more samples for training.

(e).Visualize filters from layers first and last convolutional layers of the trained model.



(a) Convolutional filter 1 (b) Original input (c) Convolutional filter 5

Figure 8: Activation of convolutional layer