CS 280 HW3 Mini Places Challenge

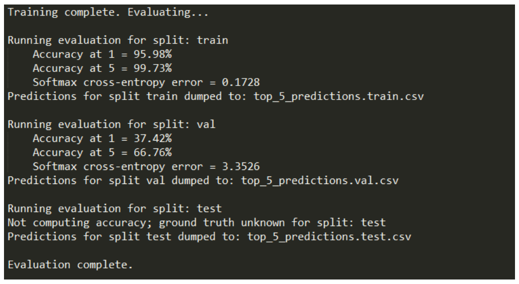
BetaGo: Jiaying Shi (24978491), Mo Zhou (21242515)

March 16, 2016

Best Model:

Our best model is a 17 layer convolutional neural network modified from VGG-16 network blablabla

The screenshot of the training log is shown as below:



```
Training complete. Evaluating...

Running evaluation for split: train
    Accuracy at 1 = 95.98%
    Accuracy at 5 = 99.73%
    Softmax cross-entropy error = 0.1728
Predictions for split train dumped to: top_5_predictions.train.csv

Running evaluation for split: val
    Accuracy at 1 = 37.42%
    Accuracy at 5 = 66.76%
    Softmax cross-entropy error = 3.3526
Predictions for split val dumped to: top_5_predictions.val.csv

Running evaluation for split: test
Not computing accuracy; ground truth unknown for split: test
Predictions for split test dumped to: top_5_predictions.test.csv

Evaluation complete.
```

From the above results we can see that in this case, the validation accuracy at 1 is about 37.42 while the accuracy at 5 is 66.76%. The training error is very small. That indicate in the best model we get, there are over fitting problems. We then tried to modified the network to reduce parameters but the accuracy on validation data set was not as good as this one. The Kaggle test score of this model is 0.37.

To analyze the results, we plotted the confusion matrices of of the validation data with the prediction with highest probability and the top 5 predictions. For the top 5 labels, if the top 5 predictions contains the true label, we regard it as an accurate prediction. Otherwise, we compare the true label with the prediction with highest probability.

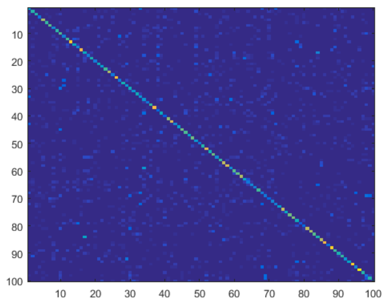


Figure 1: Confusion matrix of prediction with highest prediction

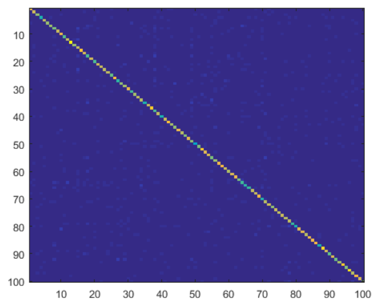


Figure 2: Confusion Matrix of top 5 predictions

From the above figures we can see that the top 5 predictions has much higher probability of covering the true label. For the first figure, the value of off-diagonal dots on row i column j represents the number images of class i are classified as class j. We take some of the wrong prediction and try to figure out what categories are easily misclassified. Taking category 29 labeled as caynon as an example, 13 images in the validation set is classified as "badlands". If we take a look at the two categories in the training data set we can see that the two categories are pretty similar. The following figures shows some images randomly chosen from the two categories in the training data set.

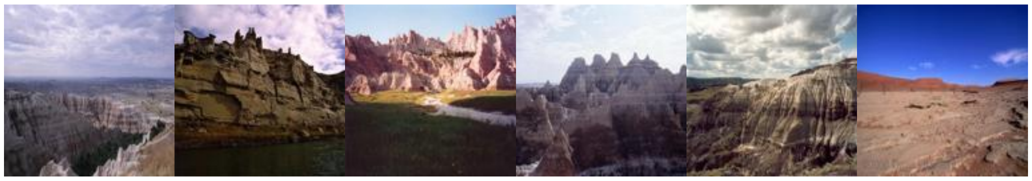


Figure 3: Examples of category "badlands"

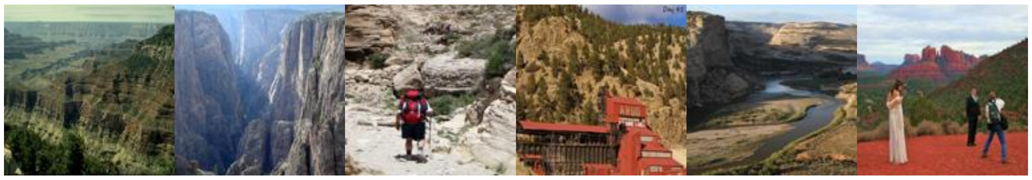


Figure 4: Examples of category "canyon"

Even people may misclassify these two classes since they share similar "features". Different people may have different definitions for the two categories. And labels of the training data set are made up by people, we can totally rule out this effect. The misclassfication of the two categories actually reflects that the model can cluster the two categories together and it can actually identify that these two categories share common features.

Analysis:

The baseline train accuracy is obtained by running the original code, where we get:
train: Accuracy at 1 = 44.25%; Accuracy at 5 = 74.27%; Softmax cross-entropy error = 2.0991
validate: Accuracy at 1 = 34.25%; Accuracy at 5 = 64.27%; Softmax cross-entropy error = 2.6122

1 Layer Effect

1.1 Remove a convolution layer

Notice that the training accuracy is much better than the validation accuracy, we suspect that the model is overfitting. Thus we first tried to delete convolution layer 4 from the AlexNet. The resulting net has 4 convolution layers, all with ReLu and 3 with pooling, followed by 3 FC layers. This net, however, generates accuracy slightly worse than the baseline:
train: Accuracy at 1 = 41.63%; Accuracy at 5 = 72.08%; Softmax cross-entropy error = 2.2216
validate: Accuracy at 1 = 33.89%; Accuracy at 5 = 63.53%; Softmax cross-entropy error = 2.6298

1.2 Add a convolution layer

Observing that removing layers may not increase accuracy, we tried to add layers. We then add a convolution layer with ReLu and another pooling layer after the second pooling layer. The performance is however disappointing:
train: Accuracy at 1 = 37.25%; Accuracy at 5 = 67.90%; Softmax cross-entropy error = 2.3961
validate: Accuracy at 1 = 30.11%; Accuracy at 5 = 59.08%; Softmax cross-entropy error = 2.8144

1.3 Add a FC layer

We also tried adding a FC layer to the original network. It is added after FC7 and it also contains 1024 nodes. This net, however, generates accuracy slightly worse than the baseline:
train: Accuracy at 1 = 43.29%; Accuracy at 5 = 73.17%; Softmax cross-entropy error = 2.1340
validate: Accuracy at 1 = 33.29%; Accuracy at 5 = 63.14%; Softmax cross-entropy error = 2.6635
Notice that the training accuracy of this net is similar to the original net, however the validation error is greater. It might be due to overfitting of the added FC layer. Hence 3 FC layers seem to be appropriate for our dataset.

2 Parameter Effect

Notice that parameters can effect the outcomes greatly, we also experimented with different model parameters for better performance. By changing the crop size from 96 to 114, halfing the stepsize and setting the momentum from 0.9 to 0.8, our net generates accuracy slightly worse than the baseline:
train: Accuracy at 1 = 40.00%; Accuracy at 5 = 70.75%; Softmax cross-entropy error = 2.2839
validate: Accuracy at 1 = 31.42%; Accuracy at 5 = 61.60%; Softmax cross-entropy error = 2.7254

3 Fine-tuning

One effective approach to improve model accuracy is using fine-tuning because it has been demonstrated that the middle layers of a CNN usually holds general characteristics. Therefore, we used the resulting weights of a AlexNet trained on the ImageNet dataset as the intial weights of our model. The ImageNet model is trained on a outside dataset. Note that the FC layers in the two models have different number of nodes, so we discard the weights for the FC layers. As expected, Fine-tuning converges much faster than starting from scratch. We are able to achieve 40% training accuracy at 1 with only thousands of iterations. After 50000 iterations, we obtain:
train: Accuracy at 1 = 69.13%; Accuracy at 5 = 91.62%; Softmax cross-entropy error = 1.0797
validate: Accuracy at 1 = 44.34%; Accuracy at 5 = 73.68%; Softmax cross-entropy error = 2.2397
We observe that the validation error is much smaller than the other methods we tried, comfirming the effectiveness of fine-tuning. However, fine-tuning ues data ouside of the given train data set. We are not allowed to submit the test predictions from fine-tuning so we are not able to compare its performace on the test set ot the performance of other methods.

4 Deeper Nets

4.1 Variations of VGG

Observing that more convolution layers produces better result, we tried to use a deeper net to train our model. Three variations of the 16-layer VGG model is used: one orignal, one with 4 convolution layers less and one with 6

more convolution layers. We reduce the batch size and increase the step size to save memory. The performance of the original VGG has best performance after 15000 iterations and is shown below:
train: Accuracy at 1 = 95.98%; Accuracy at 5 = 99.73%; Softmax cross-entropy error = 0.1728
validate: Accuracy at 1 = 37.42%; Accuracy at 5 = 66.76%; Softmax cross-entropy error = 3.3526
We observe significant overfitting of the model. However, the validation error is still slightly better than the results from AlexNet.

4.2 Variations of GoogLeNet

We also implemented the GoogLeNet on our miniplaces dataset. GoogLeNet has 22 layers with more convolution layers than VGG. It requires more memory and time to run. To save on memory, we reduce the batch size to 64 and increase the iteration size to 100. It turns out the training error jumps greatly between consecutive iterations even after 30000 iterations. We therefore reduce the iteration size in the hope of a more robust model. The resulting model with 30000 iterations has the following performance:
train: Accuracy at 1 = 21.56% Accuracy at 5 = 51.17% Softmax cross-entropy error = 3.0915 Predictions for split train dumped to: top_5_predictions.train.csv
validate: Accuracy at 1 = 20.24% Accuracy at 5 = 49.99% Softmax cross-entropy error = 3.1386 Predictions for split val dumped to: top_5_predictions.val.csv
Since the loss improves pretty slow for a big model like this and we do not have good enough resource and time to train the model with more iterations, though we suspect that this model will perform better but the accuracy at 30000 iteration is the worst one.

5 Visualizations

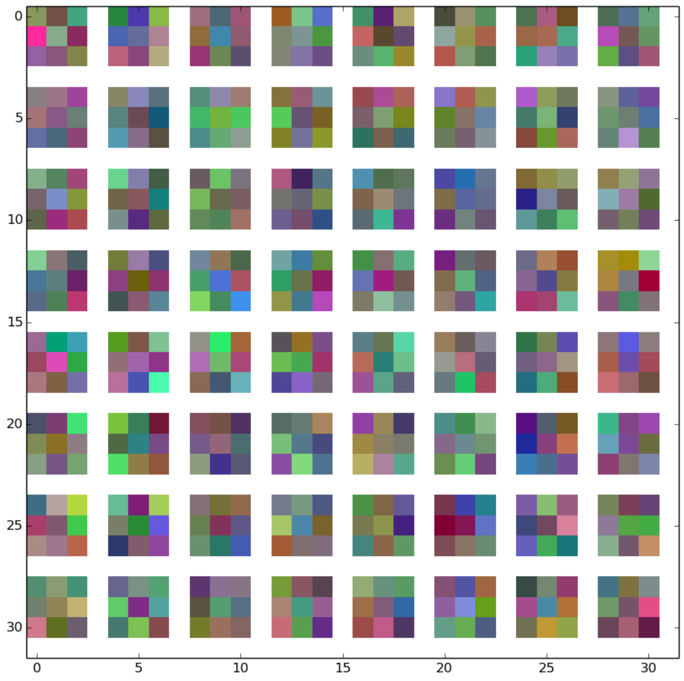


Figure 5: Filters in conv1_1

Figure 5 shows the first layer of the filters, conv1_1 in our best model. The filters are 3×3. They present very general features that are Gabor-like in some sense. We also observe that each subimage has a somewhat general shape and color which represent the building blocks of the images.