Eric Page Project0

Experiment 1:

Learning Rate: 1e-4
Optimizer: Adam
Loss Function: Cross-Entropy
Epochs: 20
Batch Size: 1024
Models Tested: 3
All Training Data Used

I designed my model to follow a similar pattern as other more advanced networks, several layers of convolutional layers followed by a couple fully connected layers and finally a softmax layer. I intentionally left out convolutional stride and pooling because I felt the images were too small to reduce the amount of information. Using the cross entropy loss function was an obvious choice since this is a classification problem with no overlap. 20 epochs was chosen because most of the models tended to converge around that time. The optimizer was chosen because of its strengths over SGD and the learning rate was experimentally determined.

The only difference between the largest and second largest models tested was the larger model had an additional convolutional layer. The difference between the two larger models was small in terms of parameters, but clearly impacted the training time as seen in illustration 5 and a significant accuracy increase most visible in illustration 4. The smallest network had half the conv. filters as the next largest with the same number of layers explaining it's relatively poor performance.

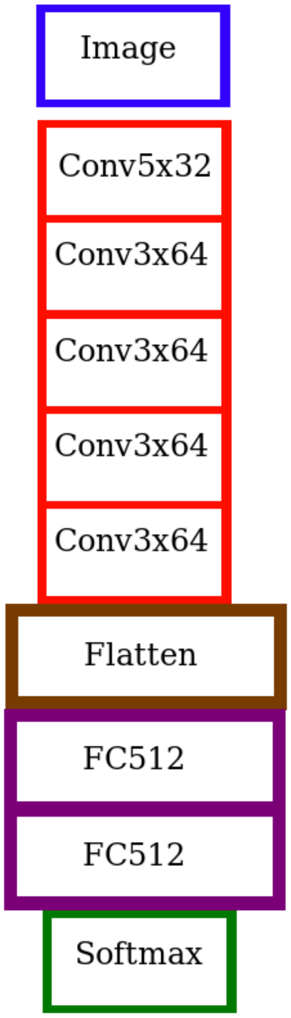

Illustration 1: Largest Architecture

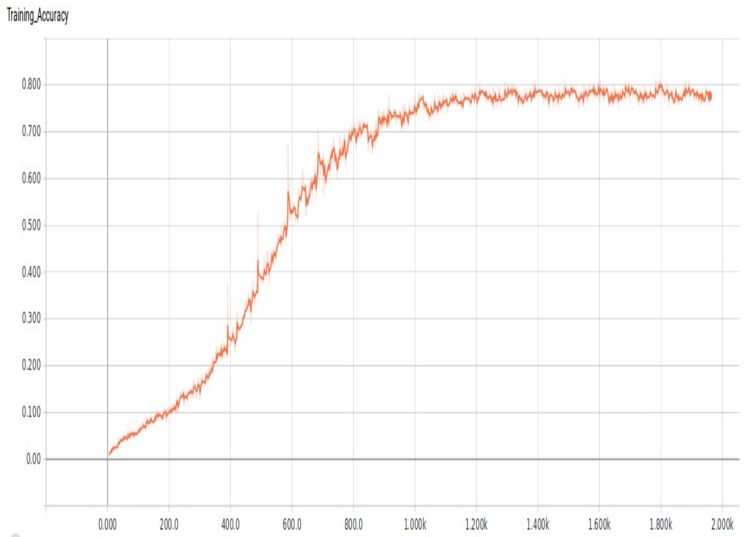

Illustration 3: Largest Architecture Training Accuracy vs Iterations

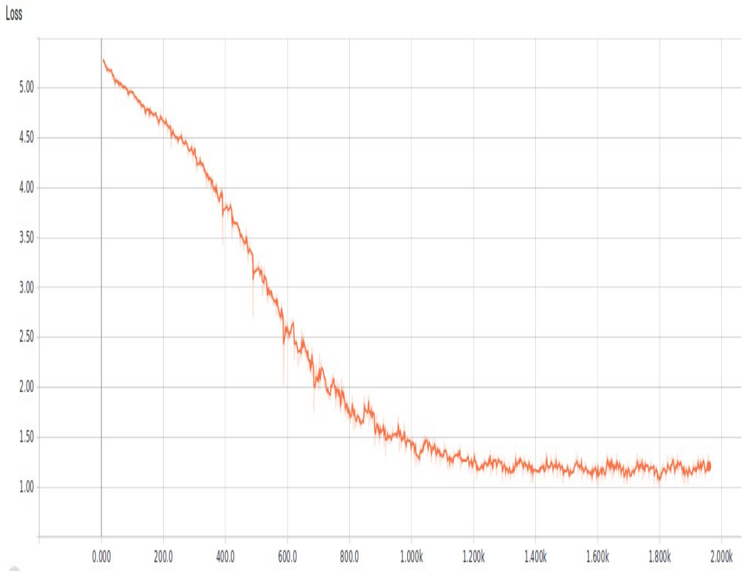

Illustration 2: Largest Architecture Loss vs Iterations

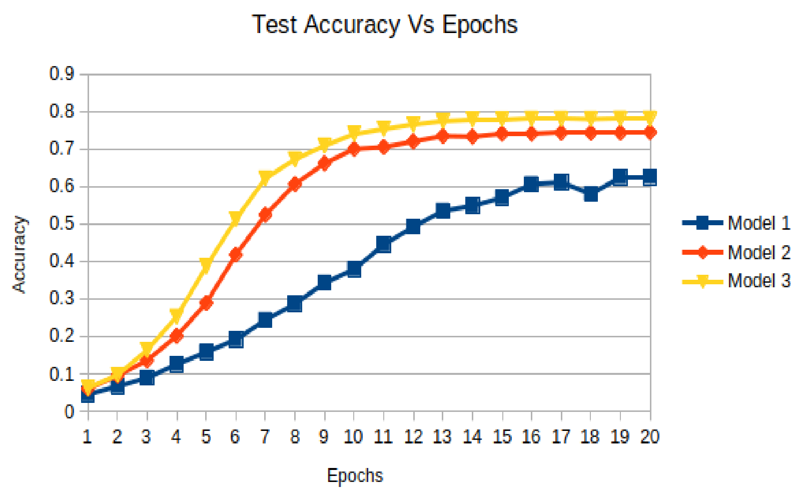

Illustration 4

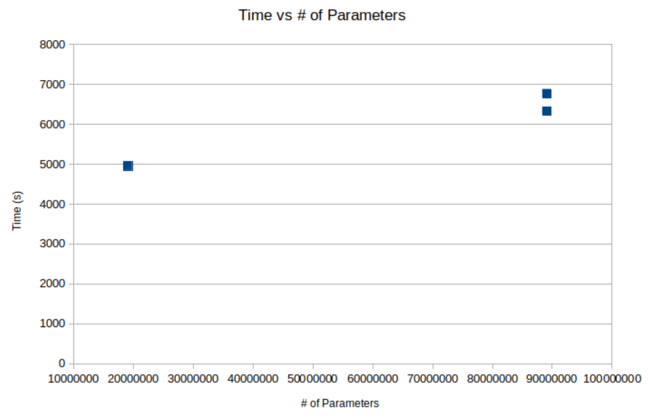

Illustration 5

Experiment 2:

Learning Rate: 1e-4
Optimizer: Adam
Loss Function: Cross-Entropy
Epochs: 20

Batch Size: 1024
Amount of Training Data Used: 30%, 70%, 100%

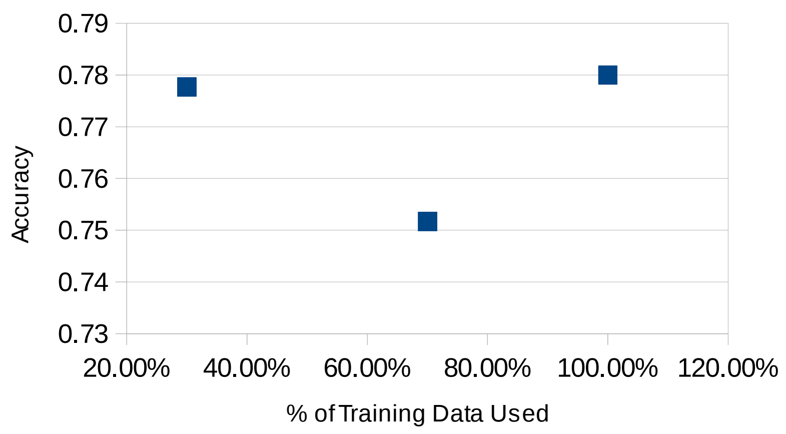

Illustration 6: Test Accuracy vs Amount of Training Data

Surprisingly, the amount of data used didn’t have a strong correlation with the test accuracy. This may be because my model doesn’t use any normalization methods such as L2 loss or dropout. Predictably, training time decreased with the use of less data.

Experiment 3:

Learning Rate: 1e-6
Optimizer: Adam
Loss Function: Cross-Entropy
Epochs: 20
Batch Size: 1024
Model: AlexNet

I chose AlexNet because it has been around a while and more pretrained models are available for it. To tune it, I kept all of the convolutional layers locked as well as the first fully connected one. As one might expect, tuning a pre-trained network starts off at a relatively high accuracy from the start. I did expect it to get a higher accuracy but I used a fairly low learning rate and in hindsight, retraining both fully connected layers would have probably increased performance.

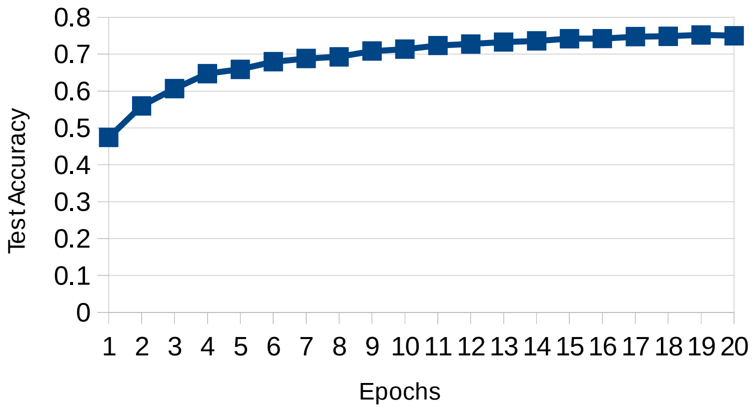

Illustration 7: Tuning Test Accuracy vs Epochs

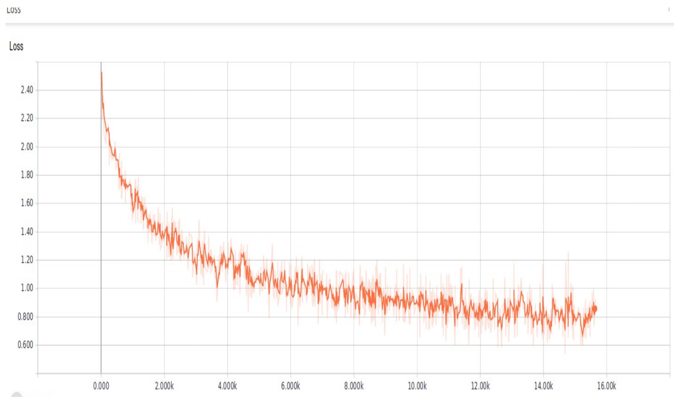

Illustration 8: Tuning Loss vs Iterations

References:

Functions to import tiny imagenet
https://github.com/seshuad/IMagenet/blob/master/TinyImagenet.ipynb

For pretrained alexnet and model
http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/

Object Orienteted Tensorflow
https://danijar.com/structuring-your-tensorflow-models/