MP2 - CS291K (SHARATH RAO)(9972332)

Training Neural Network

I am using Keras library on top of theano library in python(see Appendix). I chose the initial parameters based on my observations from MP2. I set the learning rate to 0.01, reg = 0.0, batch size =32, momentum =0.95, loss type=Categorical Cross Entropy, Iterations, Epoch = 49000, 10 epochs, Max Pool=2 layers, Pool of 2x2 filter, Feature Maps= 32,32,64,64, Filter Size and Stride=3x3 and stride of 1 which gave me a validation accuracy of 34.10% with a loss of 2.2024.

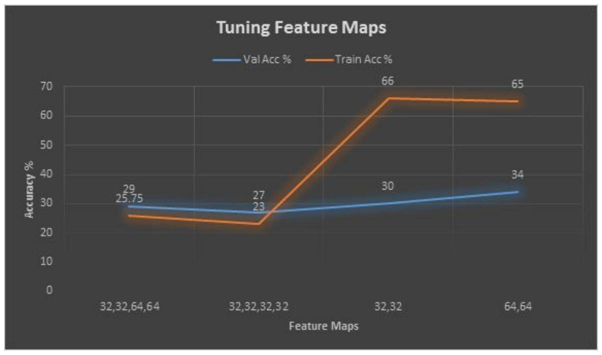
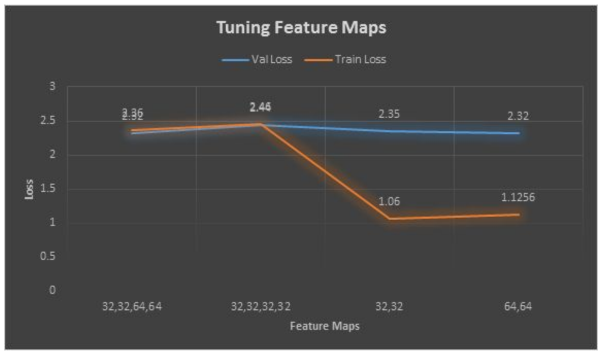
Initial Architecture

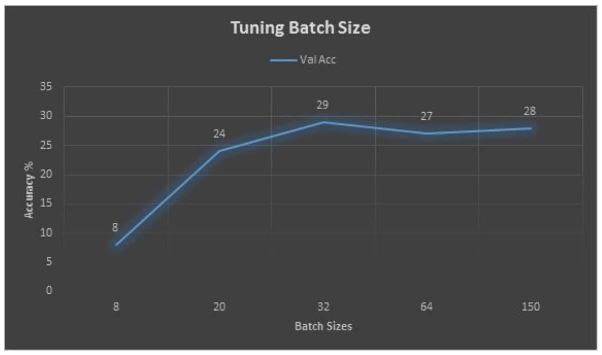
Convolution of 32 feature maps with filter of size 3x3, Relu activation Convolution of 32 feature maps with filter of size 3x3, Relu activation Max Pooling with pool_size 2x2 Dropout with p =0.25 Convolution of 64 feature maps with filter of size 3x3, Relu activation Convolution of 64 feature maps with filter of size 3x3, Relu activation Max Pooling with pool_size 2x2 Dropout with p =0.25 Flatten Dense with output size 512, Activation Relu Dropout with p=0.5 Softmax on 20 classes

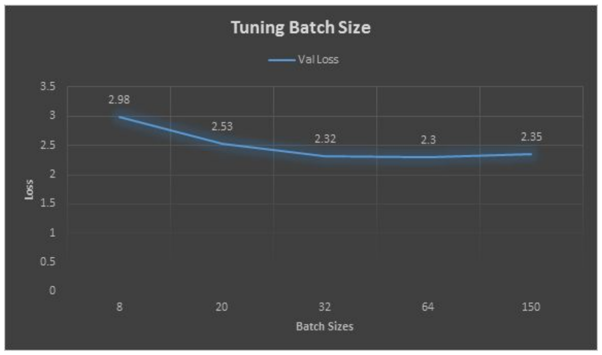
Model 2 (20 epochs)

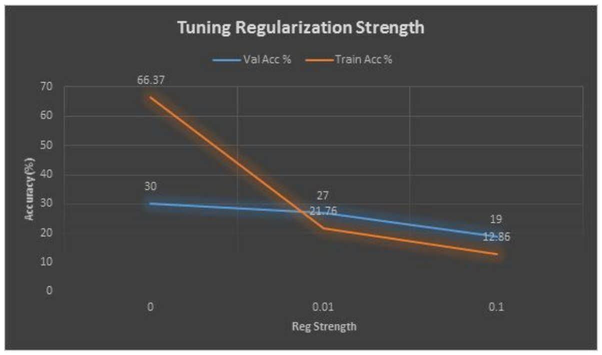
Training Acc	32.86%	Val Acc	39.3%	Test Acc	31.5%
Training Loss	2.18	Val Loss	1.99	Test Loss	2.19

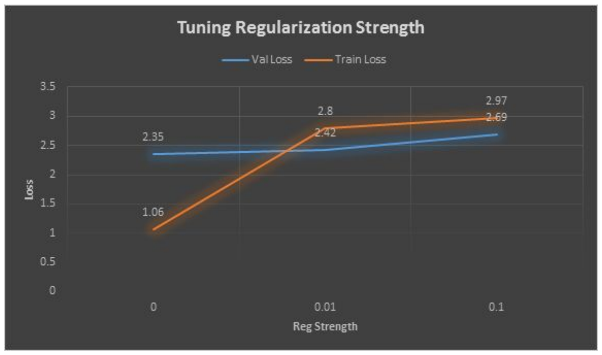
Hyperparameters Optimization

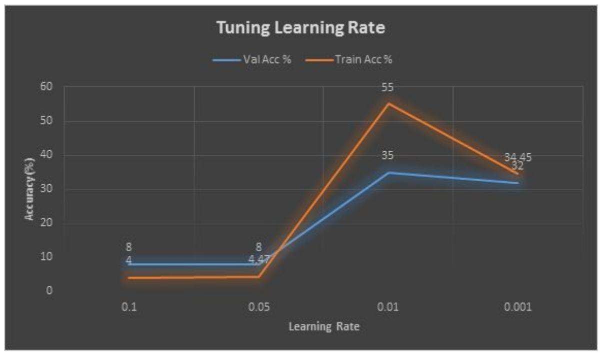
	

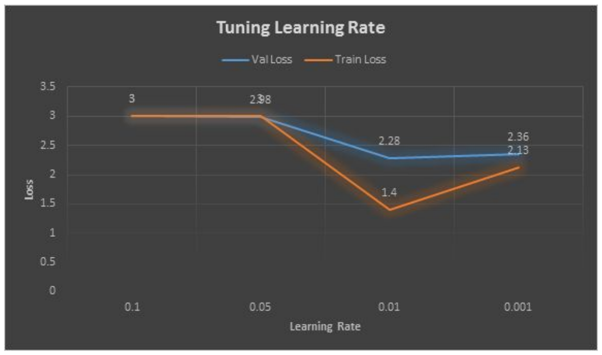












Row 1: Changing Feature Maps. Having more feature map layers increases the time and iterations needed to converge and therefore we see the lower accuracy for the first two. Overall, more feature maps(64) is better.
Row 2: Changing the Batch Size. From validation data, 32 seems to be a good batch size.
Row 3: Tuning regularization strength:Adding L2 regularization reduces overfitting (see how training acc falls as we go right). But, it also slows down the speed at which we get good accuracy.
Row 4: Tuning the Learning Rate: 0.01 is the best learning rate.

Tuning the number of Epochs

Epochs	1	2	3	4	5	6	7	8
Training Accuracy	0.1	0.18	0.23	0.29	0.33	0.39	0.47	0.52
Training Loss	2.88	2.64	2.5	2.32	2.16	1.98	1.7	1.49
Val Accuracy	0.12	0.22	0.23	0.27	0.27	0.28	0.28	0.30
Val Loss	2.83	2.51	2.44	2.40	2.47	2.42	2.43	2.35

Clearly, more the epochs better the validation accuracy. At this point I have not seen any observation that proves otherwise.

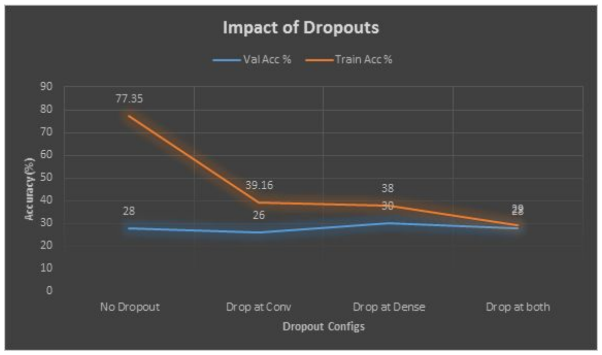
Extra Credits

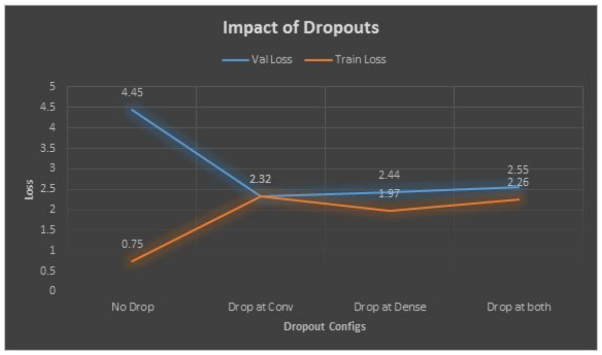
Impact of Max Pooling

	Without Max Pooling	With Max Pooling
Training Accuracy	75.96	55%
Training Loss	0.79	1.4
Best Val Accuracy	27%	35%
Val Loss	2.93	2.28

Max Pooling reduced overfitting tremendously as we can see it here. Therefore, I would use Max Pooling in the final model.

Use dropout(5 points)

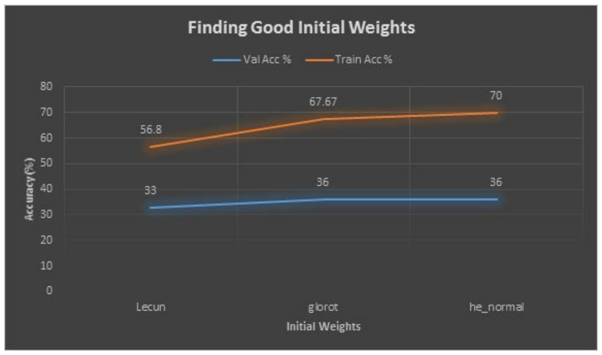


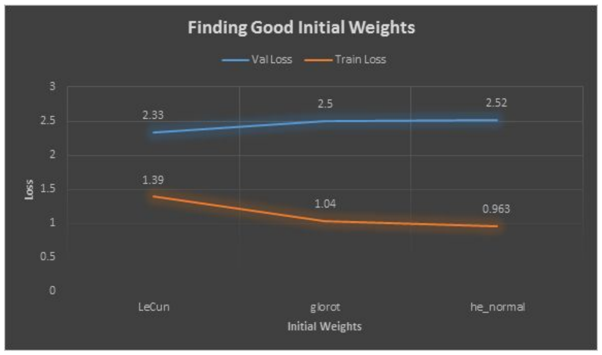


Dropouts have been added in 1)Nowhere 2) Convolutional Layer (p=0.25) 3) Dense Layers(p=0.5) 4) Both the layers. Without Dropout there is huge overfitting (See left graph first point from left) .From the graph it shows that training accuracy and validation accuracy are almost same when we use dropouts at convolutional layer and dense layers both.

Try other initialization method(5 points)

The following three types of initialization were done.
lecun_uniform: Uniform initialization scaled by the square root of the number of inputs based on LeCun 98.
glorot_normal: Gaussian initialization scaled by fan_in + fan_out based on Glorot 2010
he_normal: Gaussian initialization scaled by fan_in based on He et al., 2014

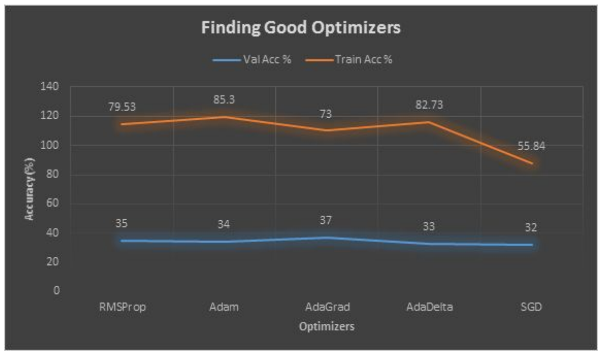


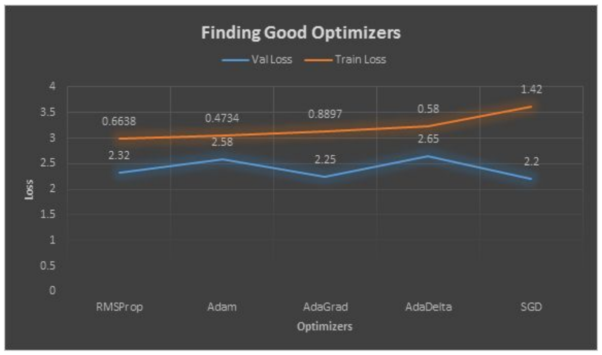


The graphs show that he_normal is the best initial weight for given dataset.

Try different optimization method(5 points)

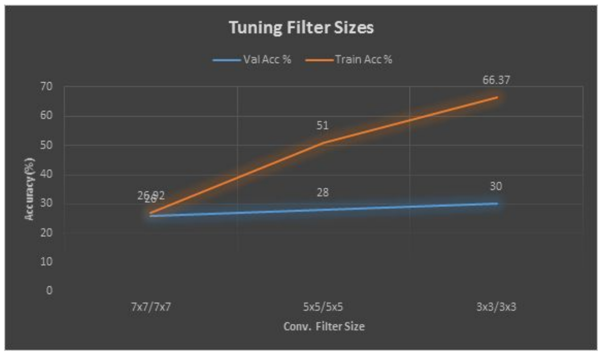
RMSProp:An optimizer that utilizes the magnitude of recent gradients to normalize the them.
AdaGrad: The learning rate is adapted component-wise, and is given by the square root of sum of squares of the historical, component-wise gradient.
Adadelta: Matthew Ziller based paper for adaptive learning rate.
SGD: Stochastic Gradient Descent
Adam: It is based on adaptive estimates of lower-order moments.

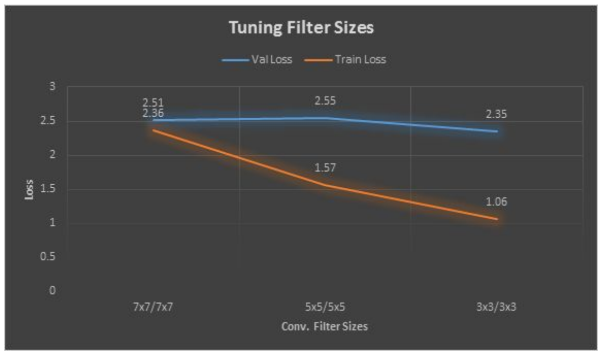




Choice = AdaGrad

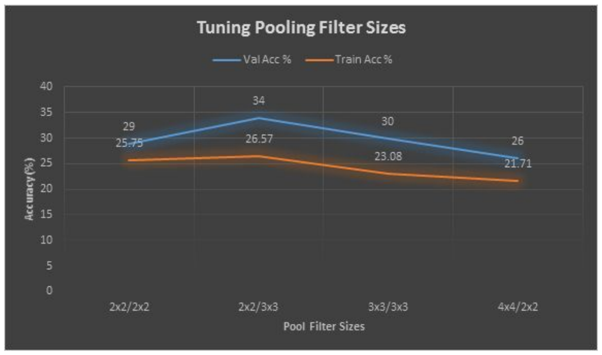
Try different filter sizes of the convolution layer(5 points)

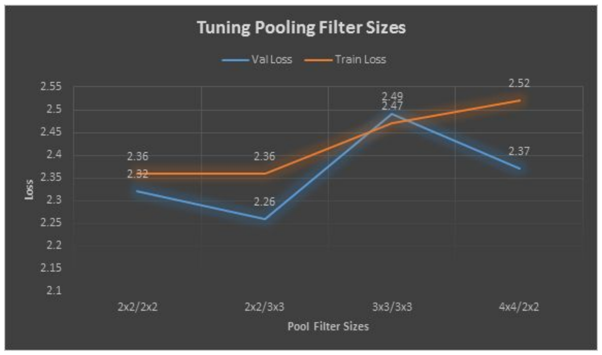




Increasing filter sizes seem to take away useful information. Or maybe the convergence is slower. Either way, for quicker results, I would go with 3x3 filters at both the layers.

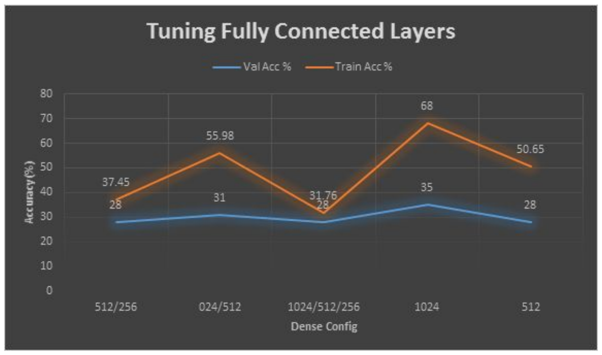
Try different filter sizes of the pooling layer (5 points)

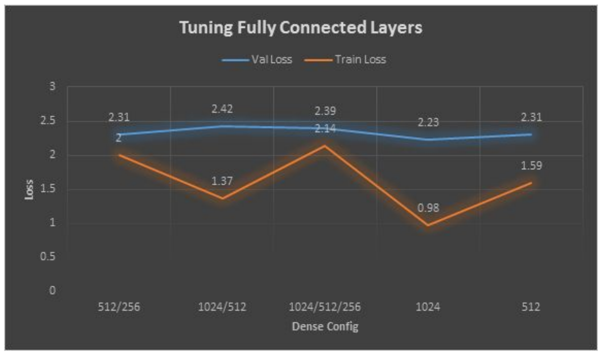




Best validation accuracy is when I use 2x2 Max Pooling after first set of feature maps and 3x3 after second.

Try different number of hidden neurons for the fully connected layer(5 points)





From the graph we can see the dense layer plays an important role wrt training accuracy. Increasing the output size of the hidden layers (512 to 1024) improves training accuracy alone. Meanwhile a three hidden layers of output size 1024,512 and 256 reduces overfitting. The best validation accuracy is when we use a single hidden layer of output size 1024.

Final Best Model (Epoch =4)

Although, the hyper params suggest using 64 feature maps, it took a lot of time to train. In the interest of time, I chose the almost equivalent result in the hyper params - 32 feature maps. This reduced the training time and the result are as follows.

Training Acc	34.56%	Val Acc	36%	Test Acc	36.8%
Training Loss	4.78	Val Loss	2.08	Test Loss	2.1

Final Architecture

Convolution of 32 feature maps with filter of size 3x3, Relu activation
Convolution of 32 feature maps with filter of size 3x3, Relu activation
Max Pooling with pool_size 2x2
Dropout with p =0.25
Flatten
Dense with output size 1024, Activation Relu
Dropout with p=0.5
Softmax on 20 classes
SGD Regressor

Challenges

- Setting up the environment in windows
- Decreasing time taken to train the network
- Adding more feature maps, increased training time by a huge margin.

Possible Improvements

1. Try more deeper architectures
2. Adding dimension reduction techniques

Using Keras and Theano

```
sudo apt-get numpy
sudo apt-get scipy
sudo apt-get install libblas-dev liblapack-dev
git clone https://github.com/Theano/Theano.git
cd theano/
python setup.py develop
git clone https://github.com/fchollet/keras.git
cd keras/
python setup.py develop
```

Let me know if you have any trouble with installation (sharathrao13@gmail.com)