CS 480/680 Machine Learning

Assignment 4 Solutions

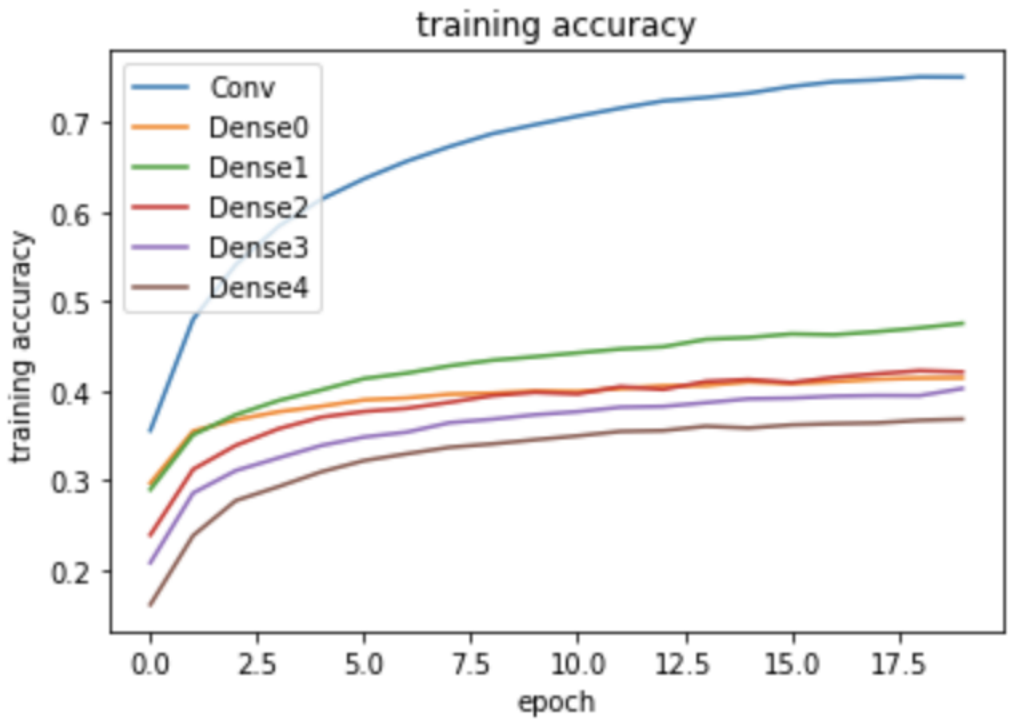
August 6, 2019

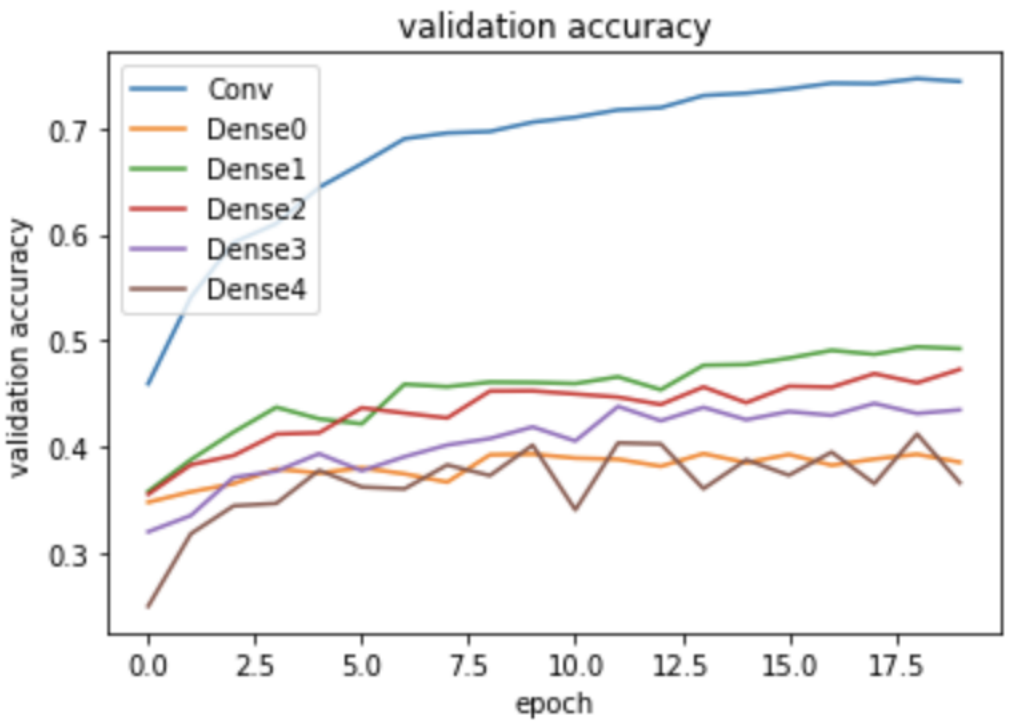
Question 1

1a

The convolutional network obtained the best results, followed by the dense networks with 1, 2, 3, 4 and 0 hidden layers. The convolutional network is much better because it is sparse with shared weights that induce equivariance. This simplifies the optimization, while requiring less data to generalize well. As we increase the number of dense hidden layers, the optimization is more difficult and unfortunately, the optimizer (RMSprop) fails to find the best weights. In theory, with at least 2 hidden layers, we could set the weights of all hidden layers except for the first one to compute the identity function. This should give us results as good as for one hidden layer, but the optimizer fails to find this solution.

NB: Overfitting is not a problem since similar results are observed for training accuracy and testing accuracy. In theory, gradients should not vanish since we are using ReLU and the networks are not very deep, but gradient vanishing might still be a problem.

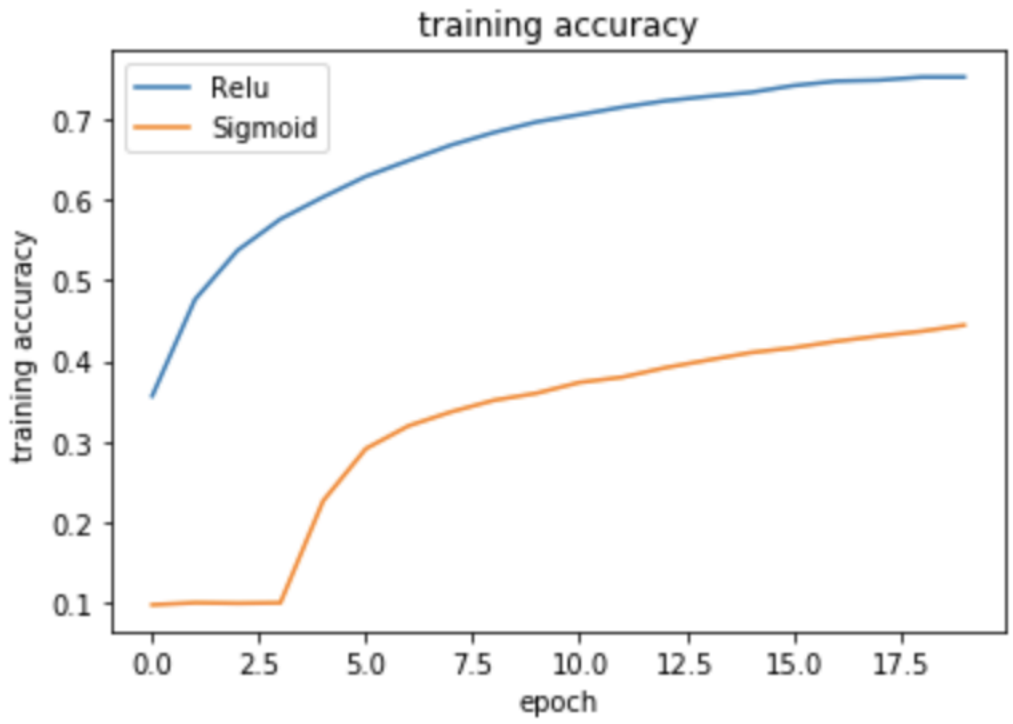

training accuracy
Conv
Dense0
Dense1
Dense2
Dense3
Dense4
training accuracy
0.2
0.3
0.4
0.5
0.6
0.7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch

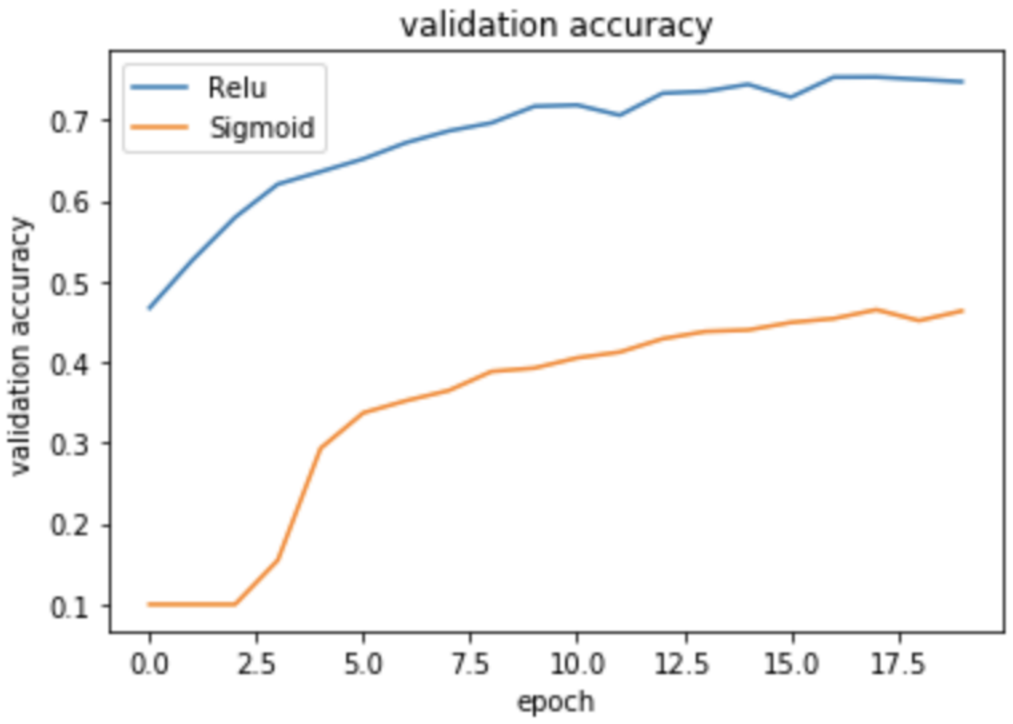

validation accuracy
Conv
Dense0
Dense1
Dense2
Dense3
Dense4
validation accuracy
0.3
0.4
0.5
0.6
0.7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch

1b

ReLU gives much better results than sigmoid units. Sigmoid units are more subject to the gradient vanishing problem. The lack of improvement in the first few epochs might be due to the fact that the starting point is in one of the tails of the sigmoid with near 0 gradient.

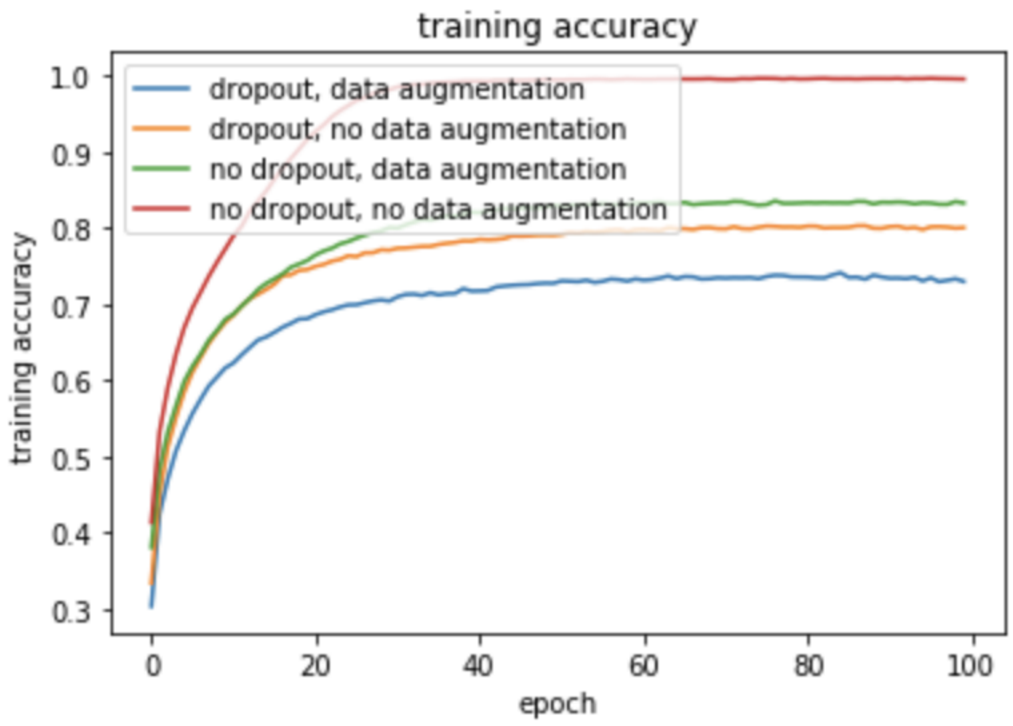
NB: Overfitting is not a problem.

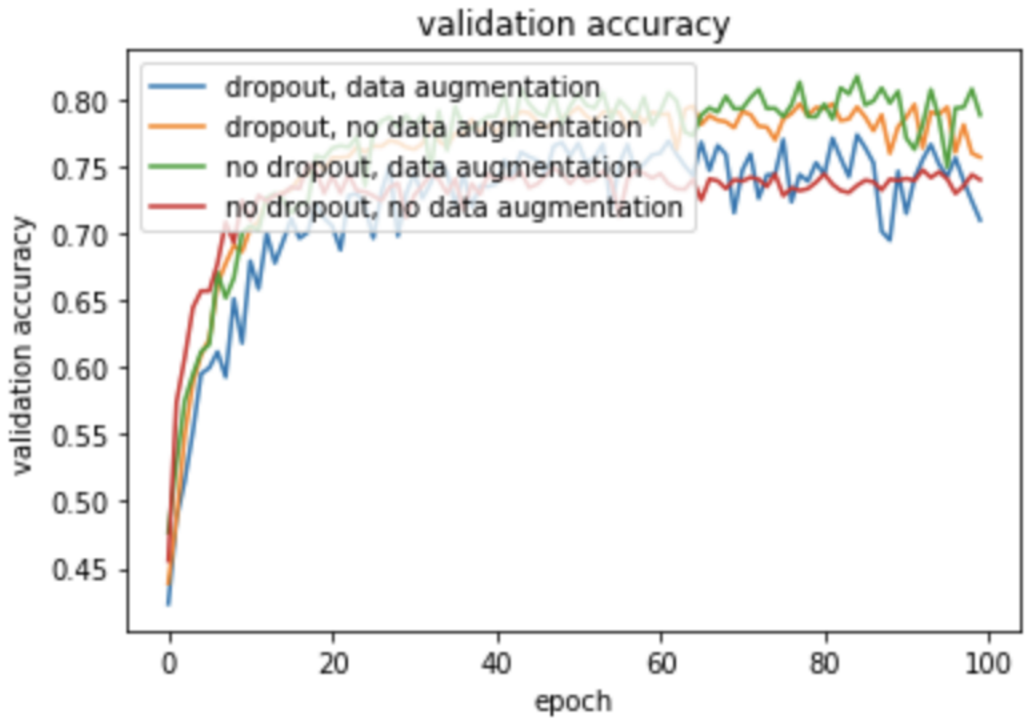

training accuracy
Relu
Sigmoid
training accuracy
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch


validation accuracy
Relu
Sigmoid
validation accuracy
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
epoch

1c

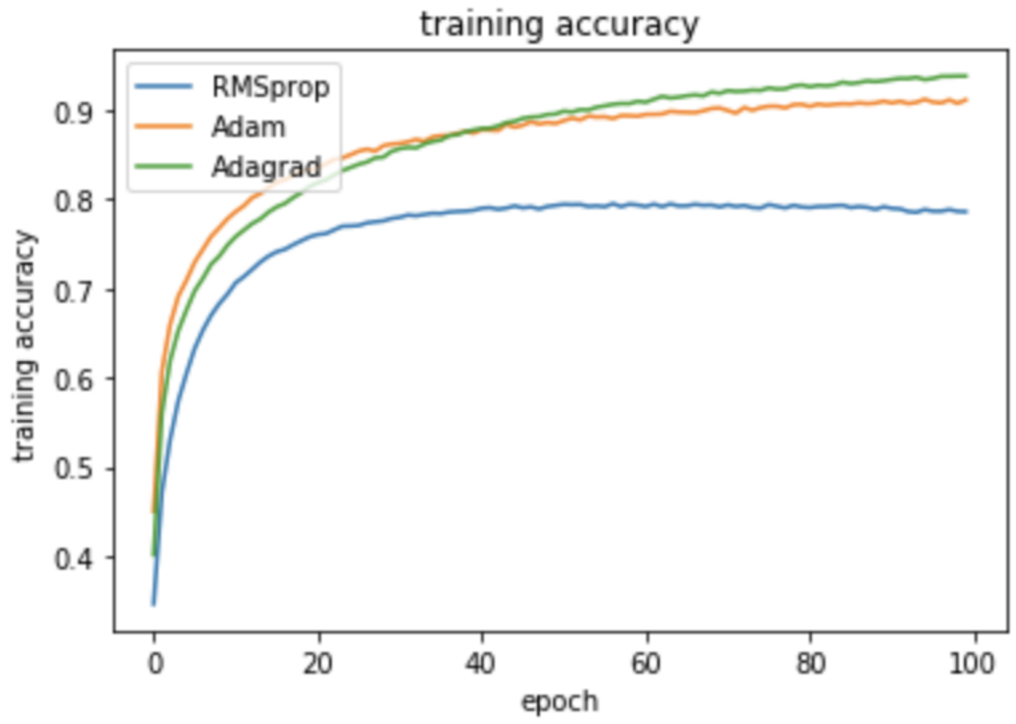
The regime without dropout and without data augmentation learns the fastest. However, it also overfits the most as we increase the number of iterations. Data augmentation helps tremendously to improve generalization. Dropout helps to improve generalization when there is no data augmentation, but it worsens the results when there is data augmentation.

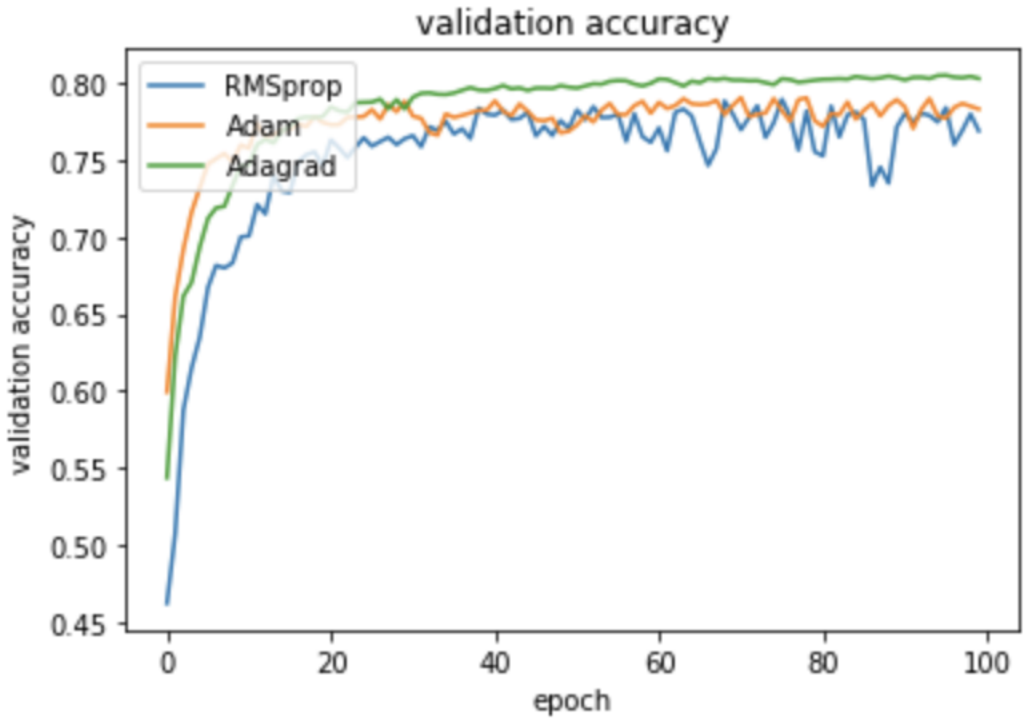

training accuracy
dropout, data augmentation
dropout, no data augmentation
no dropout, data augmentation
no dropout, no data augmentation
training accuracy
epoch


validation accuracy
dropout, data augmentation
dropout, no data augmentation
no dropout, data augmentation
no dropout, no data augmentation
validation accuracy
epoch

1d

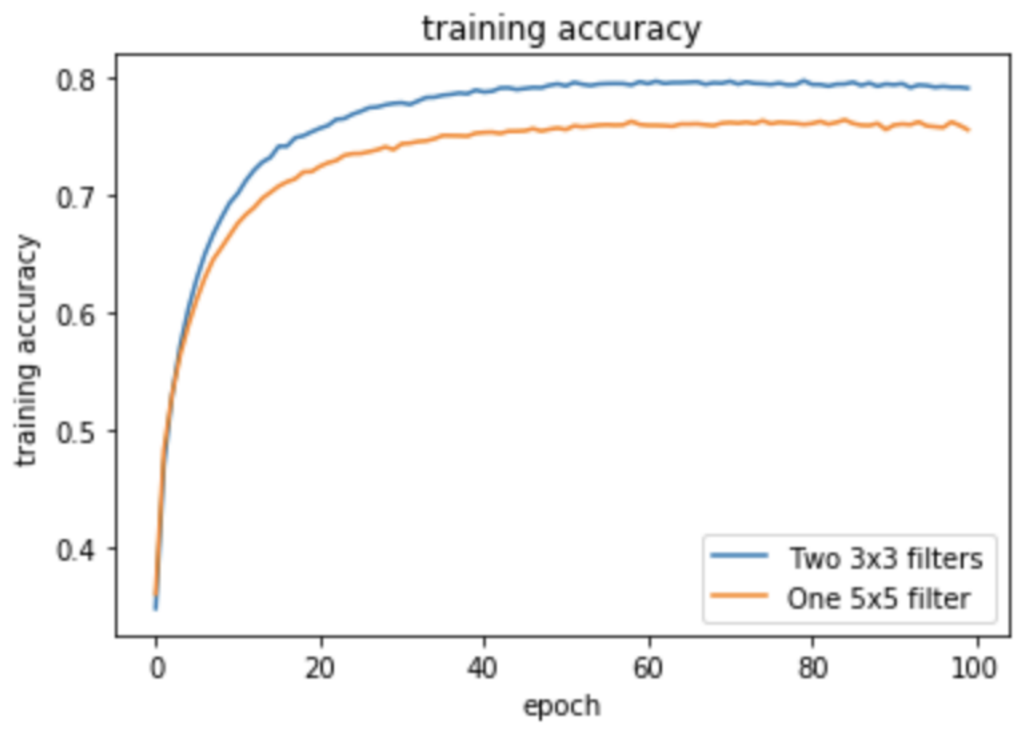
The Adagrad optimizer performed best followed by Adam and RMSprop. Adagrad quickly decays the learning rate by dividing by the square root of the sum of squares of partial derivatives. This was beneficial in this problem and as a result of quickly decaying learning rates, the accuracy curve was fairly smooth. In contrast, RMSprop decayed the learning rate more slowly, which worsened the results and produced a non-smooth learning curve. Adam was the approach that yielded the best validation accuracy in the initial epochs, but it was eventually overtaken by Adagrad. It is not clear why Adam performed worse than Adagrad in the long run.

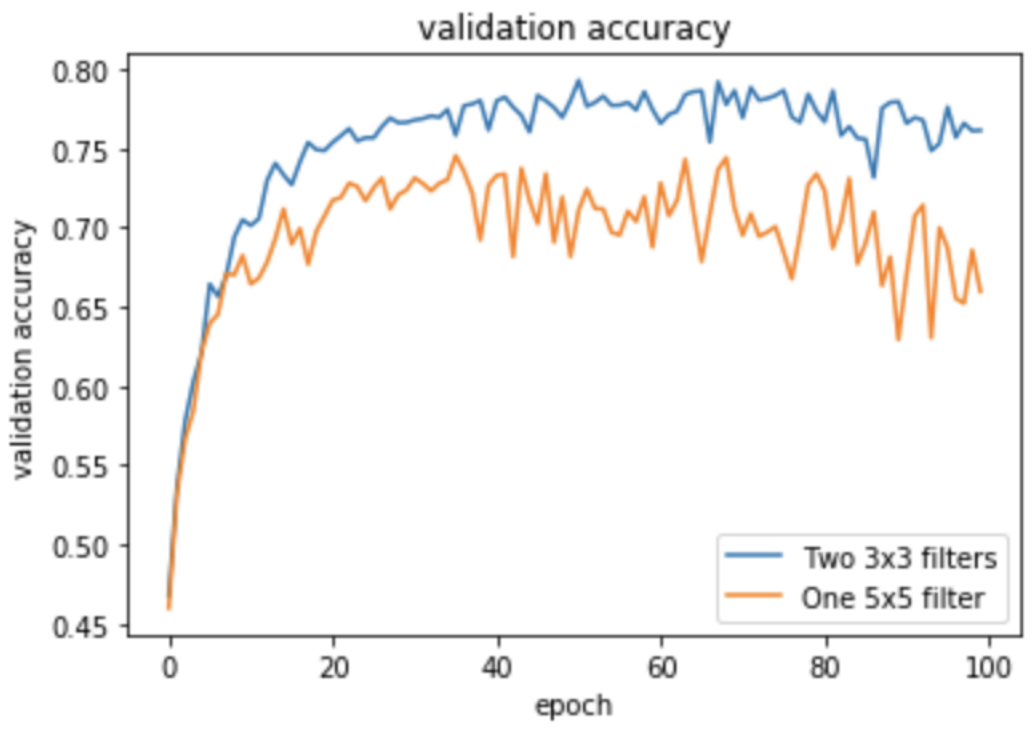

training accuracy
RMSprop
Adam
Adagrad
training accuracy
epoch


validation accuracy
RMSprop
Adam
Adagrad
validation accuracy
epoch

1e

The network with two stacked 3x3 filters for each convolutional layer performed better than a single 5x5 filter. This is consistent with the theory. By stacking pairs of 3x3 filters, we obtain deeper networks with fewer parameters while ensuring that a 5x5 receptive field is still used in the computation of each feature map.


training accuracy
training accuracy
epoch
Two 3x3 filters
One 5x5 filter
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100


validation accuracy
validation accuracy
epoch
Two 3x3 filters
One 5x5 filter
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
20
40
60
80
100

Question 2

2a

No, it is not translation invariant. Consider an image I and its translation I'. The hidden units activated by the image I are different from those activated by its translation I'. Since the weights in the softmax layer can be arbitrary, the output of the network depends on which hidden units are activated and therefore I and I' generally yield different outputs.

2b

No, max pooling only leads to local invariance and not global invariance. Consider an image I and its translation I'. Max pooling over patches of 4x4 only ensures that local translations within each patch do not affect the output. Since the image is shifted globally, then different hidden units will be activated by I and I' in both the convolution and max pooling layers. Since the weights in the softmax layer can be arbitrary, the output of the network depends on which hidden units are activated and therefore I and I' generally yield different outputs.

The above solutions are just examples. Other solutions with legitimate explanations were also accepted.