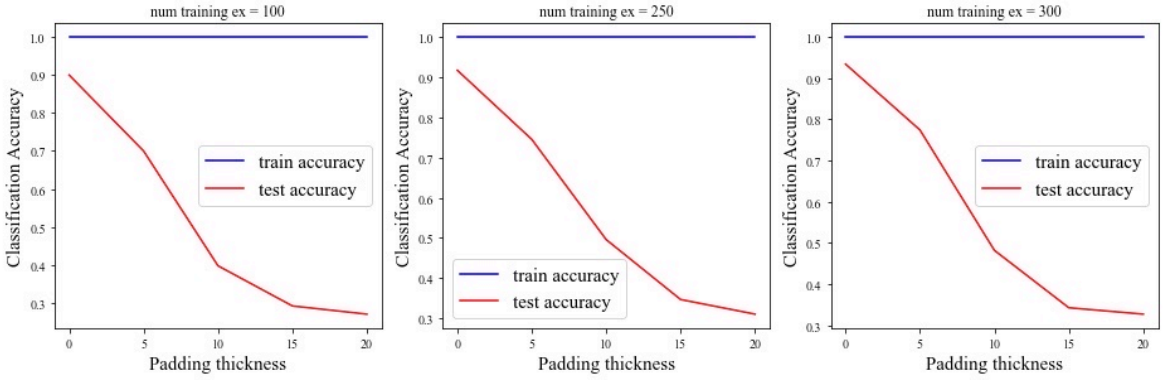
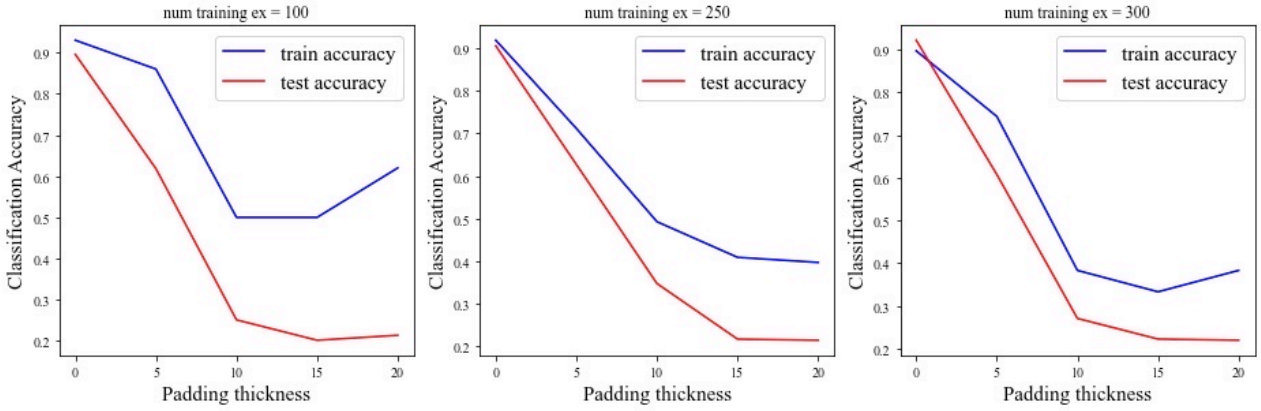
Exercise 1

1) Figure 38 using multiclass logistic regression:



2) Figure 38 using naive Bayes Classifier:



Clearly we can see that neither of these algorithms are able to learn shift-invariant features. For the multi-class logistic regression we see that our training accuracy is high but our test accuracy decreases rapidly as we increase padding. For the Bayes classifier, our accuracy for both the train and test data decreases rapidly as we increase the padding thickness. We also see overfitting as we add more training examples.

Exercise 2

Input $1 \times a \times b$

$\rightarrow$ conv 1 ($1 \times r \times r$ filters $\times n_1$, stride $= s_f$)

$$n_1 \times \left(\frac{(a-r)}{s_f}+1\right) \times \left(\frac{(b-r)}{s_f}+1\right)$$

$\rightarrow$ ReLU + maxpool ($d \times d$, stride $= s_p$)

$$n_1 \times \left(\frac{\left(\frac{(a-r)}{s_f}+1\right)-d}{s_p}+1\right) \times \left(\frac{\left(\frac{(b-r)}{s_f}+1\right)-d}{s_p}+1\right)$$

$\rightarrow$ conv 2 ($n_1 \times r \times r$ filters $\times n_2$, stride $= s_f$)

$$n_2 \times \left(\frac{\left(\frac{\left(\frac{(a-r)}{s_f}+1\right)-d}{s_p}+1\right)-r}{s_f}+1\right) \times \left(\frac{\left(\frac{\left(\frac{(b-r)}{s_f}+1\right)-d}{s_p}+1\right)-r}{s_f}+1\right)$$

$\rightarrow$ ReLU + maxpool ($d \times d$, stride $= s_p$)

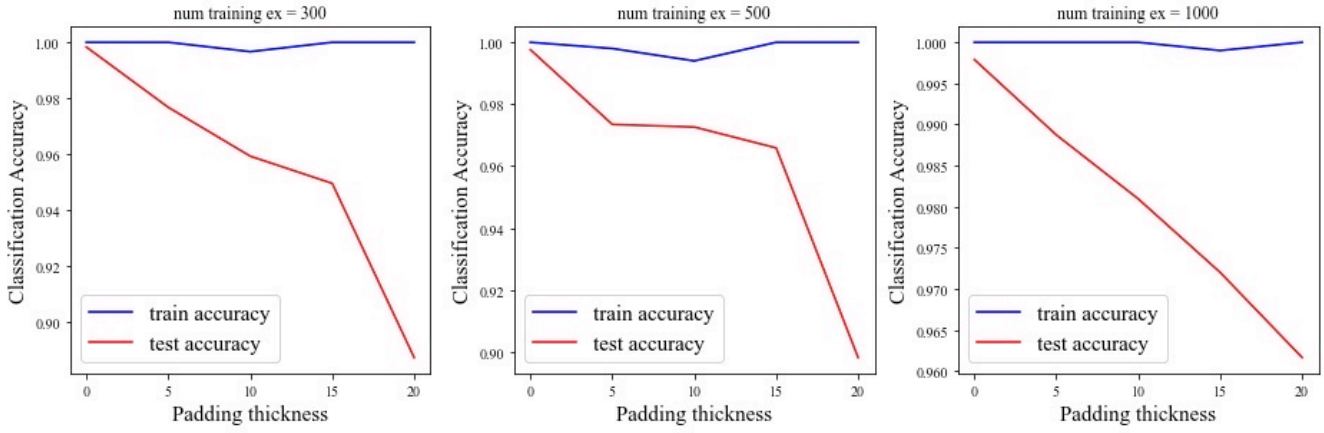
$$n_2 \times \left(\frac{\left(\frac{\left(\frac{\left(\frac{(a-r)}{s_f}+1\right)-d}{s_p}+1\right)-r}{s_f}+1\right)-d}{s_p}+1\right) \times \left(\frac{\left(\frac{\left(\frac{\left(\frac{(b-r)}{s_f}+1\right)-d}{s_p}+1\right)-r}{s_f}+1\right)-d}{s_p}+1\right)$$

$$n_2 \times \left(\frac{a-r+s_f-ds_f+s_fs_p-rs_fs_p+s_f^2s_p-ds_f^2s_p}{s_f^2s_p^2}\right) \times \left(\frac{b-r+s_f-ds_f+s_fs_p-rs_fs_p+s_f^2s_p-ds_f^2s_p}{s_f^2s_p^2}\right)$$

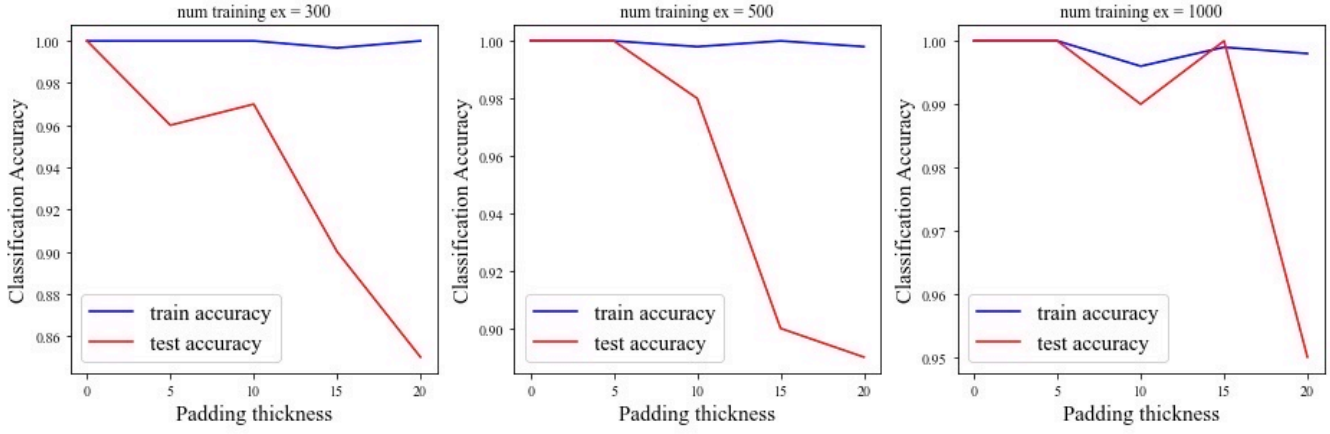
$\rightarrow$ dimension of feature vector that goes into first dense layer

Exercise 3

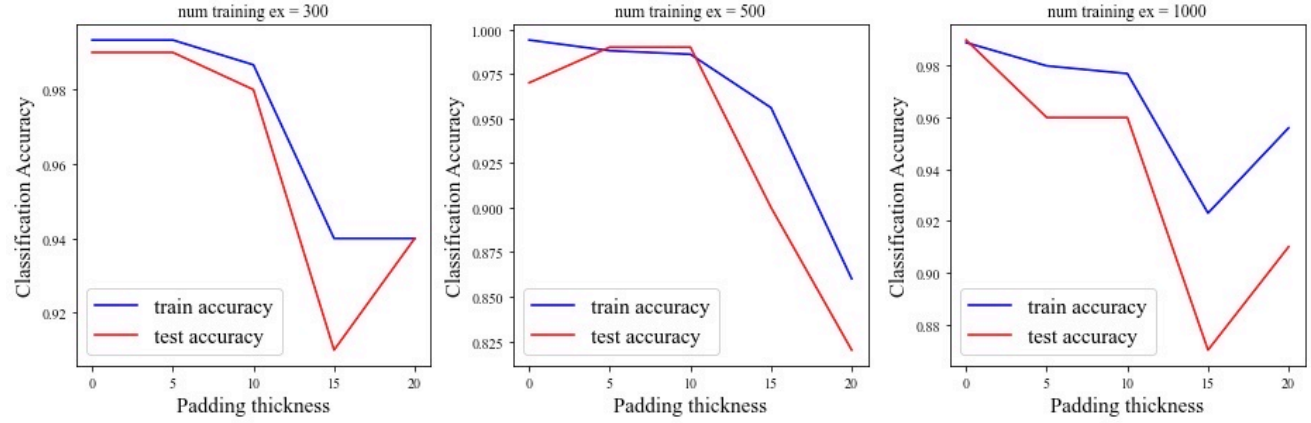
3-layer FFNN digits [0,1]



Same as above but w/ only 100 test images



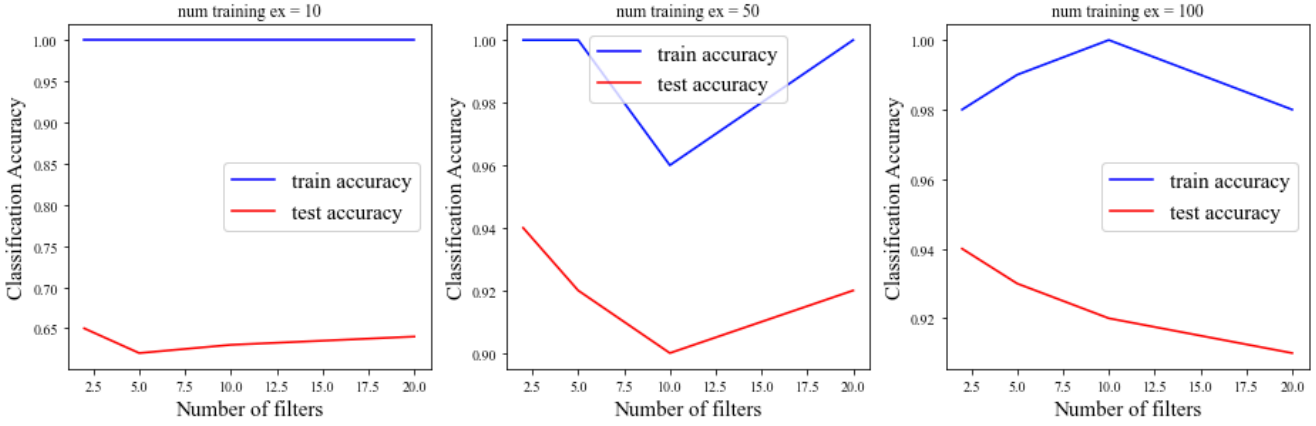
CNN digits [0,1] w/ 100 test images



For padding thickness it seems that we have great accuracy w/ CNN when padding is less than 10. When padding thickness is >10, FNN seems to have a higher accuracy. This may be due to overfitting or maybe due to too many filters. CNN though took much longer though.

Exercise 4

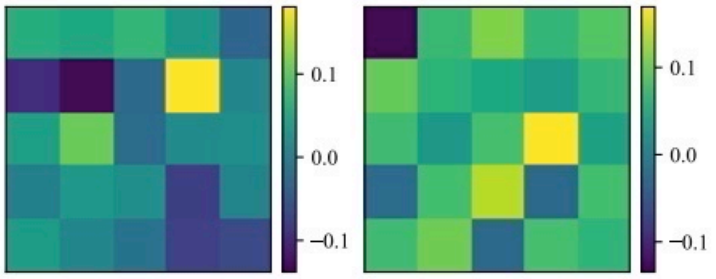
CNN for digits [0,1] w/ 100 random test images



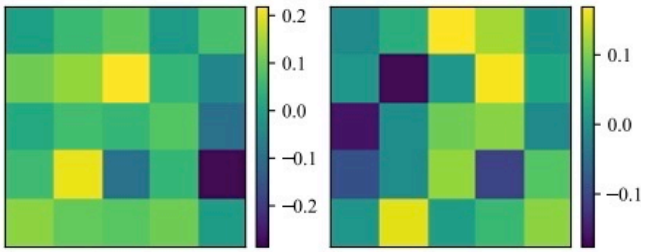
Looking at this graph, it seems to be that the optimal number of filters is two. We can see that in each of the graphs above, our test accuracy goes down after any additional filters after two. This probably means that we have a great deal of overfitting after more than two filters. It is also important to note that we have 94% accuracy with 2 filters in both graphs with 50 and 100 training examples. In this example it seems that any more than 50 training examples doesn't make a difference, but this may be due to the fact that we limited our test images to 100.

Exercise 4

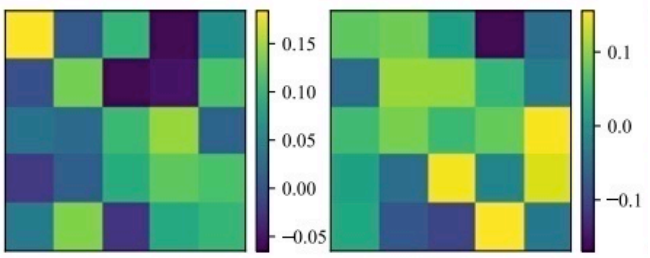
Digit 0 convolutional filter 1



Digit 0 convolutional filter 2



Digit 1 convolutional filter 1



Digit 1 convolutional filter 2