COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

5241 STATISTICAL MACHINE LEARNING Milestone project

Rong Xu
UNI: rx2180

Professor Xiaofei Shi
May 09, 2022

Introduction

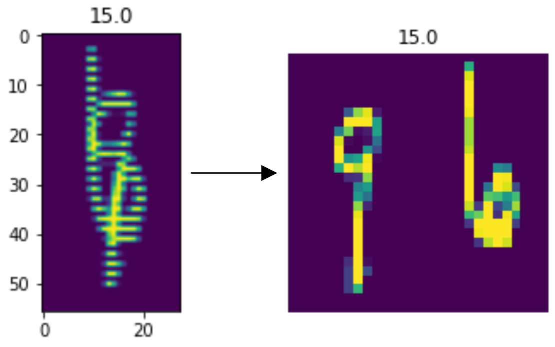
In this project, I'm using the famous MNIST data set to observe the performance of different Neural Networks (NN) and the influence of parameters on the networks.

The MNIST database of handwritten digits is one of the most commonly used datasets for training various image processing systems and machine learning algorithms. MNIST has a training set of 60,000 examples and a test set of 10,000 examples. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal effort on preprocessing and formatting.

To train the model and observe the performance of models, there are three subsets of the MNIST are considered:

1. Training set: Obtained by training data loaded from the original data set and using the random split to keep 5/6 of the original data set.
2. Validation set: Obtained by training data loaded from the original data set and using the random split to keep 1/6 of the original data set.
3. Testing set: Obtained by testing data loaded from the original data set.

The data set used in the last question is the costumed data set which is a little bit different. Besides read in all the data, there's another step of splitting numbers from the original data.



Since questions 3,4,5,7 are in the same structure, the overall procedure is introduced first as follows:

Step(a). The training data set is used to train the model while the validation set is used to observe the different performance of loss and accuracy on the training set and validation set during training using different seeds. ([3,42,123,456,789])
Step(b). During the training procedure in step a, the accuracy metric is also calculated on training and validation sets. Then, the accuracy of the training and validation set, and the loss on the validation set are plotted.
Step(c). The weights of all filters in each layer are plotted.
Step(d). Retrain the model while using different parameters. (Learning rate in (0.01, 0.2, 0.5) and momentum in (0.0, 0.5, 0.9))

In the project, I built up five different net structures to answer different questions.

```
Net1(
(fc1): Linear(in_features=784, out_features=100, bias=True)
(output): Linear(in_features=100, out_features=10, bias=True)
)

Net2(
(conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
(out): Linear(in_features=1176, out_features=10, bias=True)
)

Net3(
(conv1): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1))
(conv2): Conv2d(16, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
(batch1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(conv4): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(batch2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(batch3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(fc1): Linear(in_features=2304, out_features=512, bias=True)
(fc2): Linear(in_features=512, out_features=128, bias=True)
(fc3): Linear(in_features=128, out_features=10, bias=True)
)

Net4(
(conv1): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1))
(conv2): Conv2d(16, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
(batch1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(conv4): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(batch2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(batch3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(fc1): Linear(in_features=2304, out_features=512, bias=True)
(fc2): Linear(in_features=512, out_features=128, bias=True)
(fc3): Linear(in_features=128, out_features=19, bias=True)
)

Net5(
(conv1): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1))
(conv2): Conv2d(16, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
(batch1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(conv4): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(batch2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(batch3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
(fc1): Linear(in_features=2304, out_features=512, bias=True)
(fc2): Linear(in_features=512, out_features=128, bias=True)
(fc3): Linear(in_features=128, out_features=19, bias=True)
(dropout): Dropout(p=0.25, inplace=False)
)
```

Findings other than required questions

The seed will influence the behavior of models.

Under the same structure and parameters, the different seeds might lead to the different performances of the models concluding from the outcome of step (a) and step (b). Although using the seed I chose, all the training procedures converge and the differences are not obvious, there can be some extreme conditions leading to different convergence statuses.

Convolutional layers do improve the behaviors of neural networks.

Although the performances of SVM and ANN are already pretty well on the MNIST data set, adding convolutional layers can further enhance the accuracy of the accuracy. Besides, deeper networks are more likely to have better performance on prediction, however, there's no guarantee. It's highly dependent on the structure of the networks.

Momentum and learning rate is crucial to the model performance.

During step(d) we can observe dramatic differences in outcome under different combinations of learning rates and Momentum. There's no ground truth selection of combination of those two parameters that can outperform for all data sets but for the MNIST data set specifically, among all the combinations considered, learning rate = 0.01 and momentum = 0.9 seems to be the best one.

Data set quality is important to the model performance.

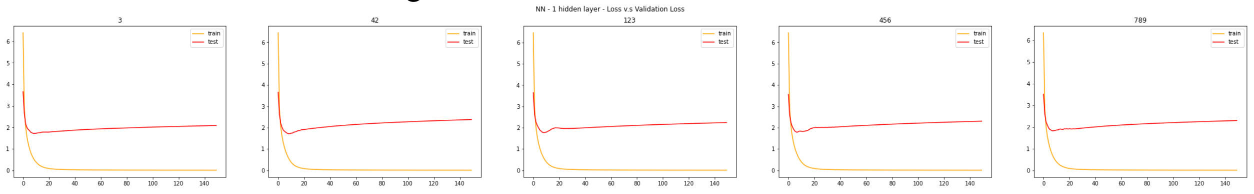
In the last question, a customed data set is used to train the deep model that I've used before. However, the performance is poorer. It's because there is more than one projection onto a target now. This makes it harder for the model to learn the patterns.

Answers to required questions

3.(a) How does the network's performance differ on the training set versus the validation set during learning? Use the plot of training and testing error curves to support your argument.

A:

1. The training error is lower than the validation error mostly.
2. The elbow of the curve shows around the same stage.
3. After the elbow, the loss of validation set goes up while the loss of training set stays low. This can be an indicator of overfitting.



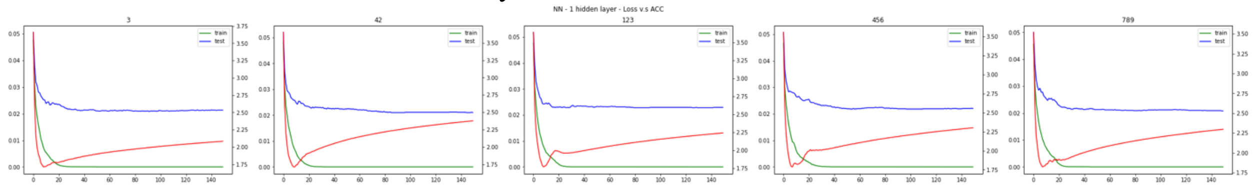
3.(b) We could implement an alternative performance measure to the cross entropy, the mean
miss-classification error.
Plot the classification error (in percentage) vs. number of epochs, for
both training and testing. Do you observe a different behavior compared to the behavior of the cross-entropy error function?

A:

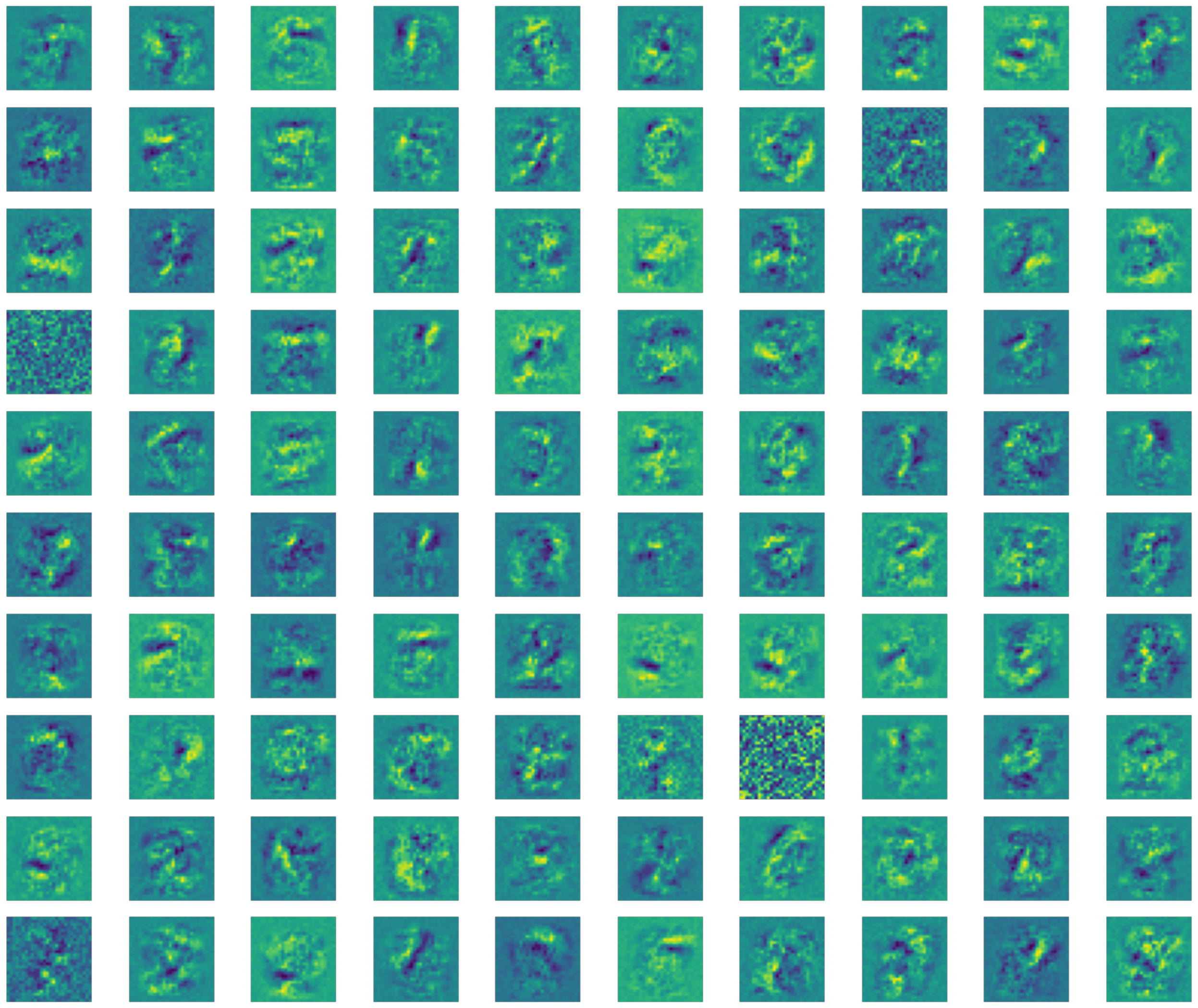
The red line here is the loss of the validation set. The green and blue lines are the accuracies of the train and test set separately.

Although the loss indicates there is the risk of overfitting, the accuracy stays steady after the turning point for both training and test set. This can be the advantage of using the cross-entropy criterion.

In my opinion, since the classification accuracy is a discretized result while the predictions are numerical, when overfitting starts, the probability the model gives out is being closer to the middle but not crossing the line. This will not influence the classification result. Thus, the loss will deteriorate worse than the accuracy.

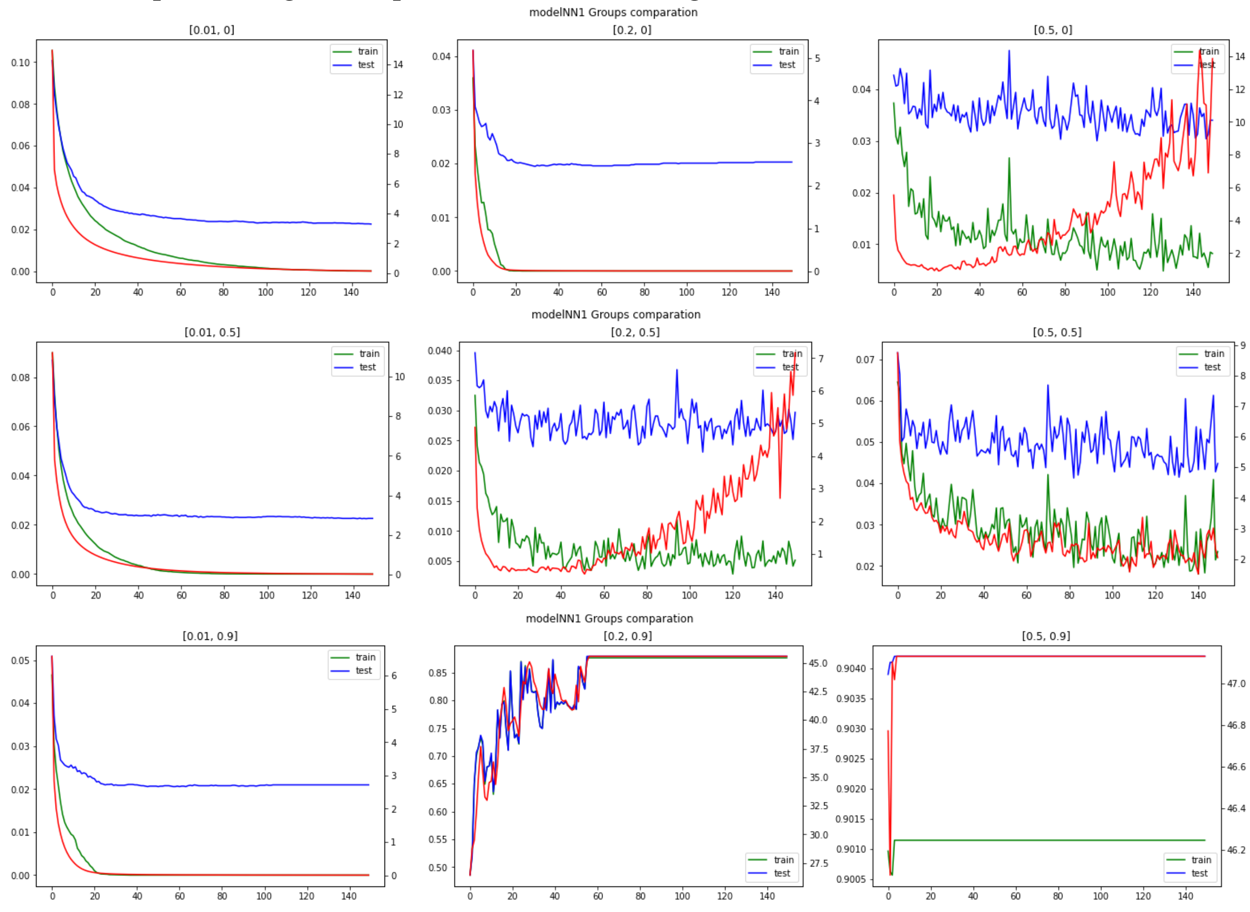


3.(c) Visualize your best results of the learned W as one hundred 28×28 images (plot all filters
as one image, as we have seen in class). Do the learned features exhibit any structure?
A:
The model does learn some features. The centers of the figures show some blur patterns. However, there is an exception. But in NN, it will not influence the overall performance usually since there are so many neurons and layers.

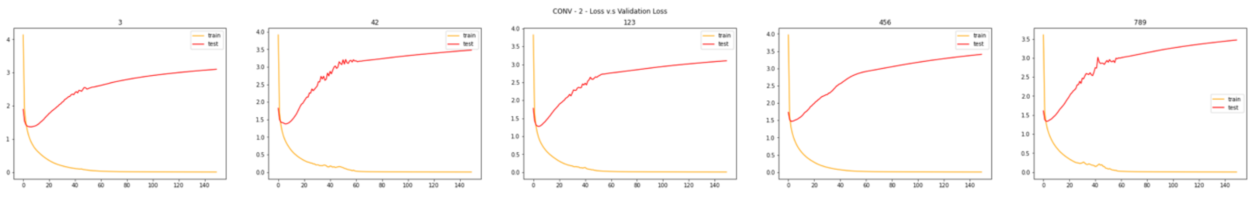


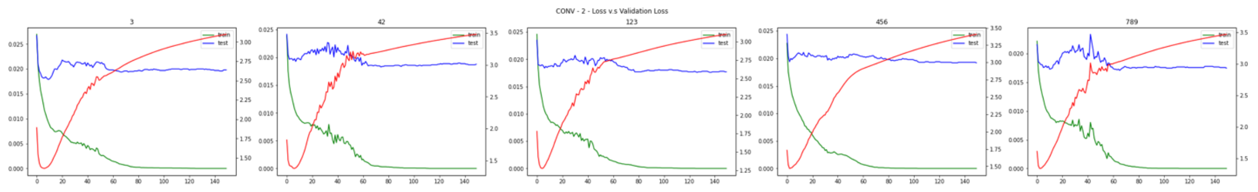
3.(d) Try different values of the learning rate. You should start with a learning rate of 0.1. You should then reduce it to .01, and increase it to 0.2 and 0.5. What happens to the convergence properties of the algorithm (looking at both average cross entropy and % incorrect)? Try momentum of 0.0, 0.5, 0.9. How does momentum affect convergence rate? How would you choose the best value of these parameters?
A:
The learning rate and momentum are both influencial to the model performance.

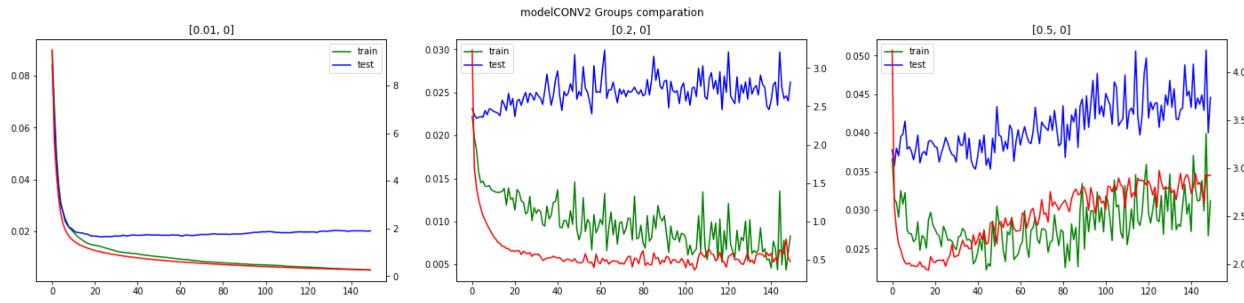
When momentum = 0, the convergence properties will not be influenced by the learning rate we are trying. The model coverage no matter what while the minimums they settle in and the epochs they use to find the best model are a little bit different.
However, when momentum ≠ 0, the convergence properties will be affected by the momentum and the learning rate. The higher the momentum, the more likely that the model will not converge. Also, the higher momentum will make the curve more fluctuant.
The best pair among all the parameters we are using is lr = 0.01 and momen = 0.9

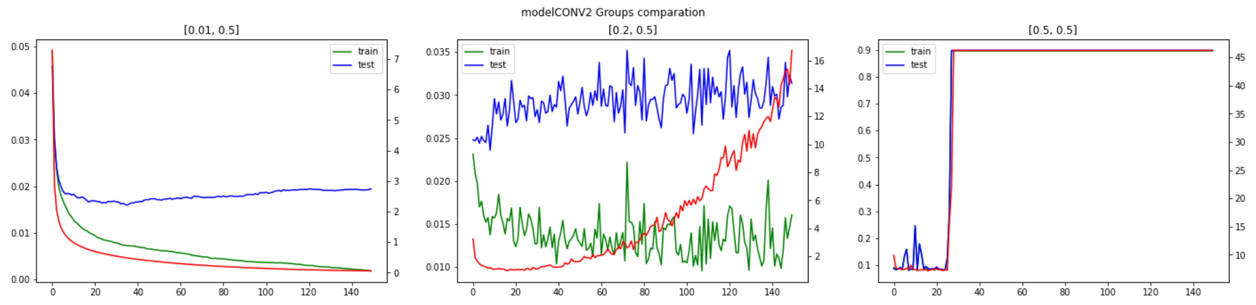


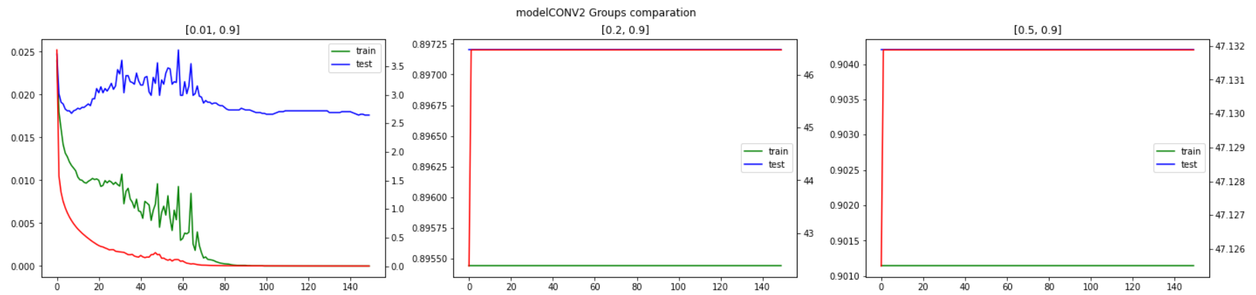
4. Redo part 3(a) - 3(d) with a CNN i.e. with one 2-D convolutional layers → Relu activation → Maxpooling with appropriate hyperparameters. Compare the best result from the single layer neural network and the CNN, what could you conclude?
A:
A convolutional layer set will increase the model performance but it will take a longer time to train. It is also more likely to overfit. This is the "trade-off". We need to evaluate the resource we have and the desired accuracy we can achieve. As seen from the plots, both the accuracy and the loss if lower than the results given by the simple one-layer linear network under similar parameters conditions.





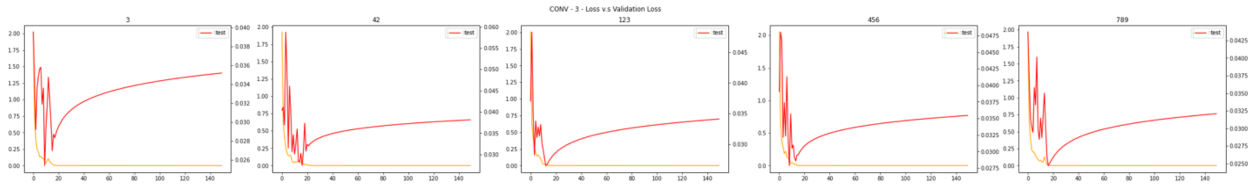


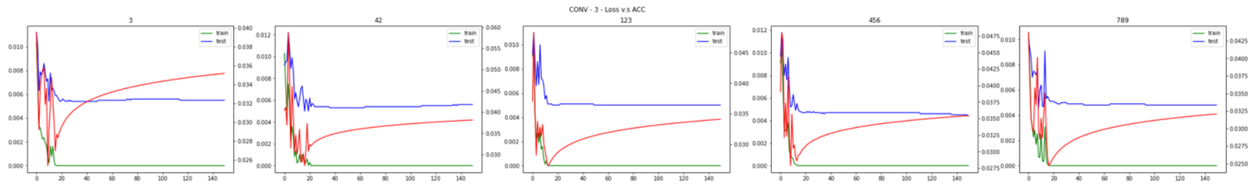




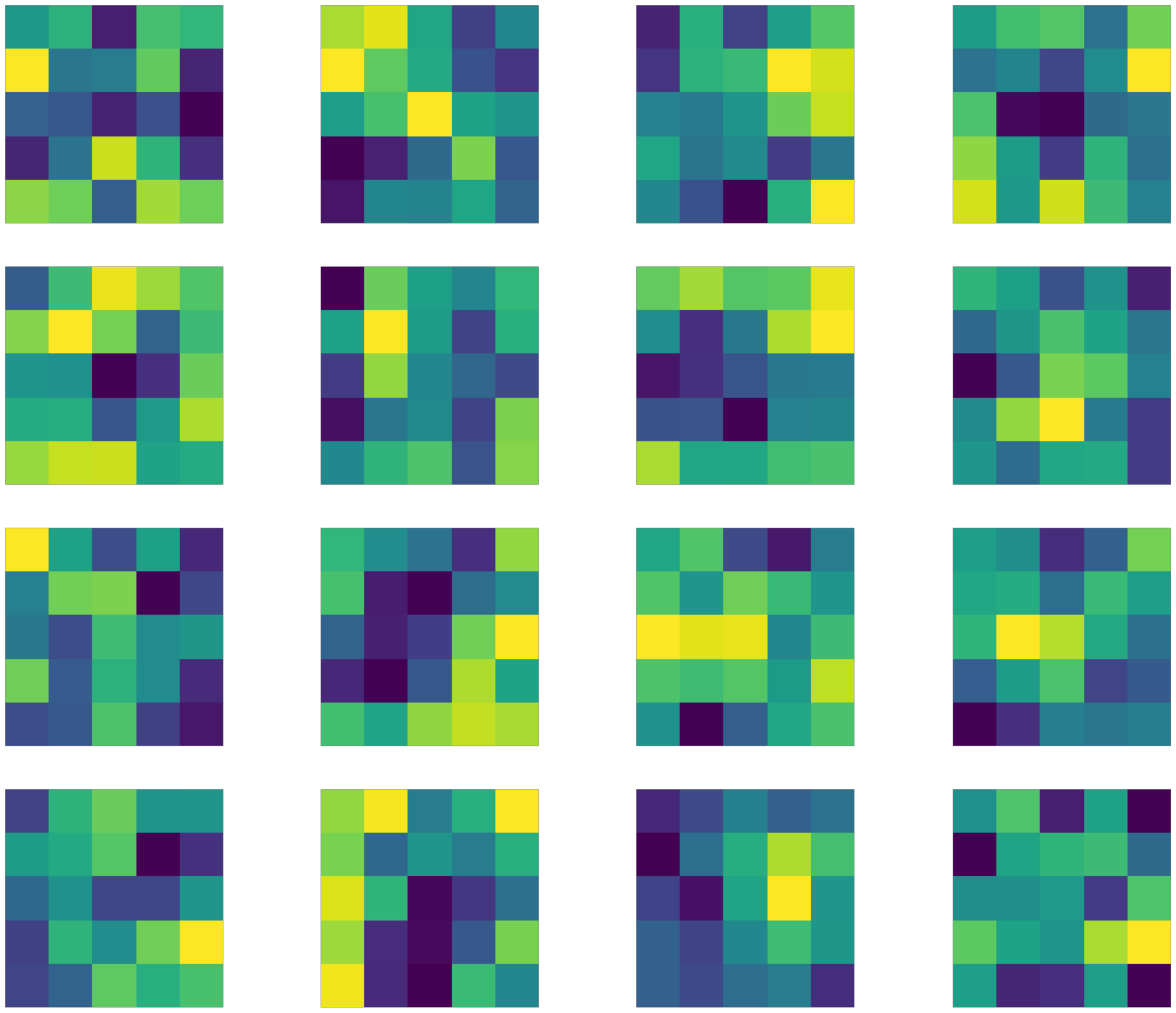
5. Redo part 3(a) - 3(d) with your favorite deep learning architecture (e.g., introducing batch normalization, introducing dropout in training) to beat the performance of SVM with Gaussian Kernel, i.e., to have a test error rate lower than 1.4%.

A: As shown in the plots below, the accuracy is way lower than 1.4% at the end of training around 0.6%.

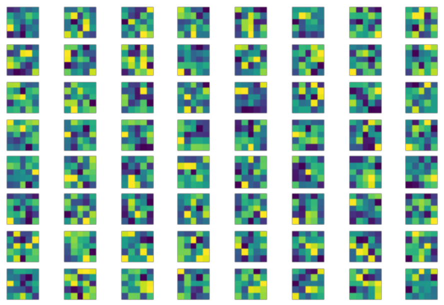




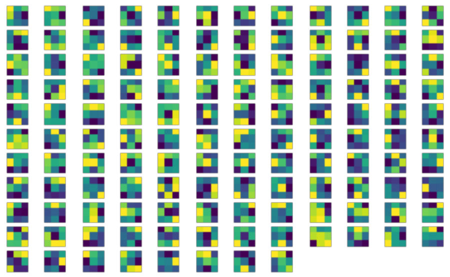
```
Total convolutional layers: 5
```



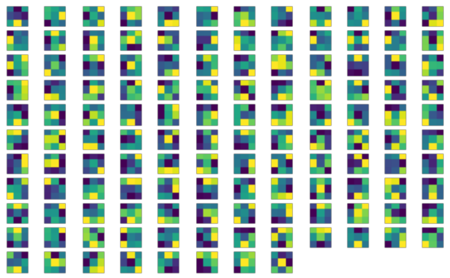
```
======= Conv filter 0 =======
```



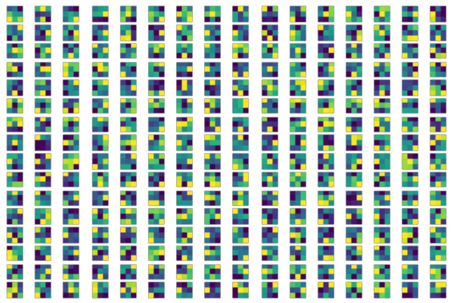
```
======= Conv filter 1 =======
```



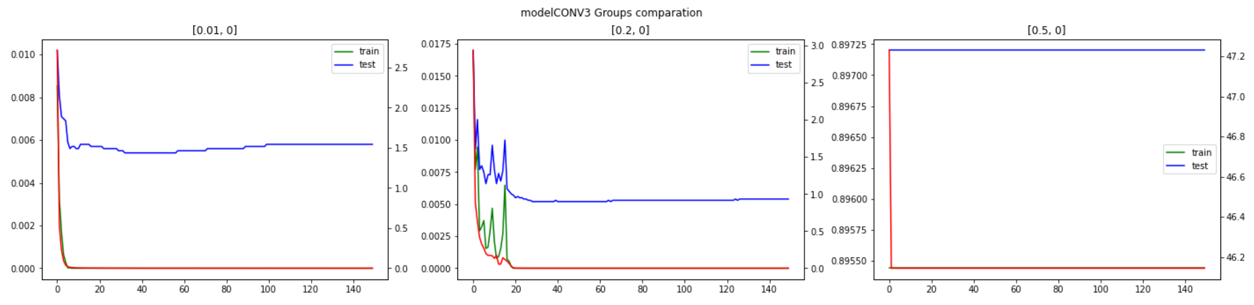
```
======= Conv filter 2 =======
```

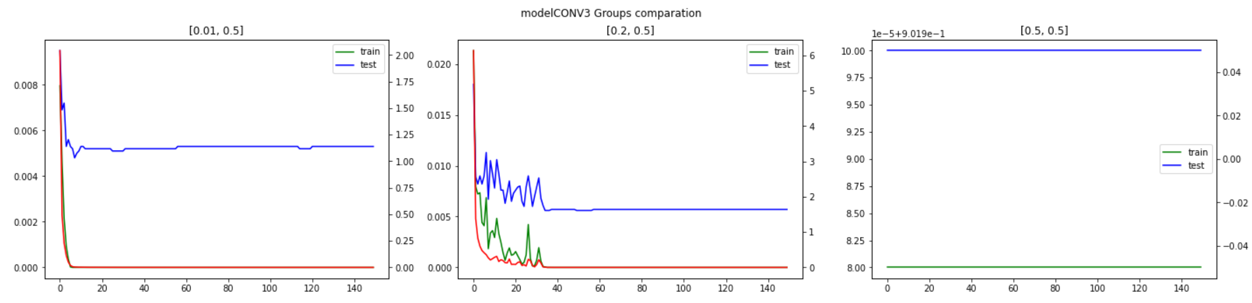


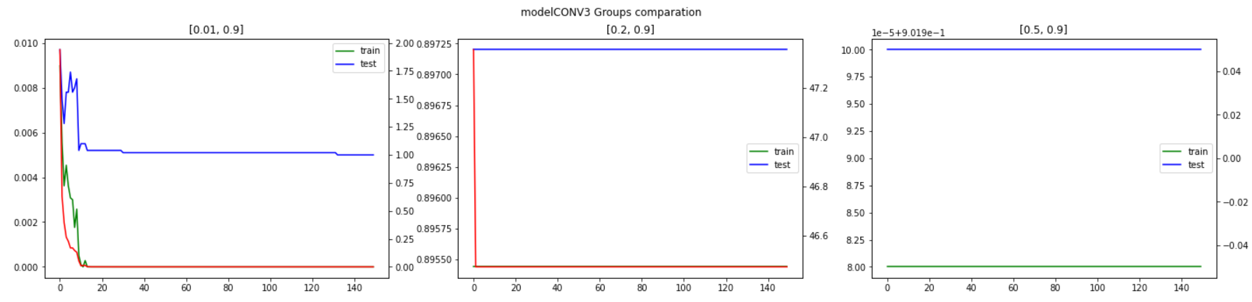
```
======= Conv filter 3 =======
```



```
======= Conv filter 4 =======
```



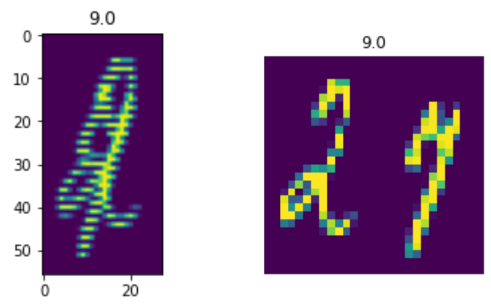




6. As a warm up question, load the data and plot a few examples. Decide if the pixels were scanned out in row-major or column-major order. What is the relationship between the 2 digits and the last coordinate of each line?

A:

The sum of two digits equal to the last number in each line. They are scanned out in row-major.



In the following questions, all images are preprocessed to convert from the left side condition into the right side condition.

7. Repeat part 3(a) - 3(d) with at least two of your favorite deep learning architecture (e.g., introducing batch normalization, introducing dropout in training) with respect to with train.txt, val.txt and test.txt. In particular,

(a) Using train.txt to train your models.

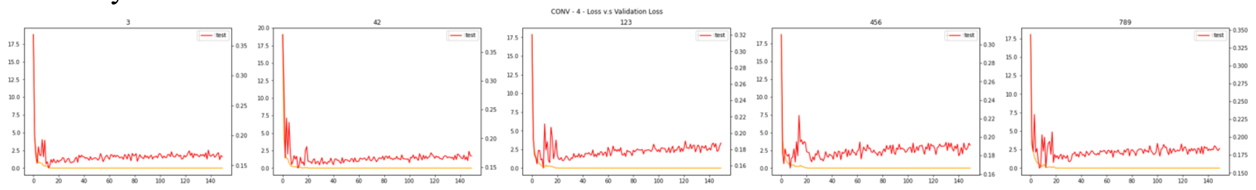
(b) Using the validation error (i.e., the performance on val.txt) to select the best model.

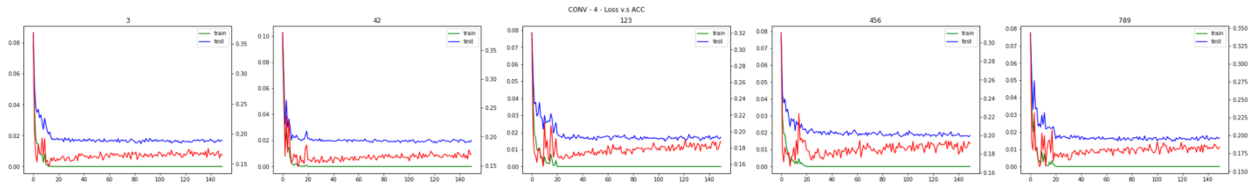
(c) Report the generalization error (i.e., the performance on test.txt) for the model you picked. How would you compare the test errors you obtained with respect to the original MNIST data? Explain why you cannot obtain a test error lower than 1%.

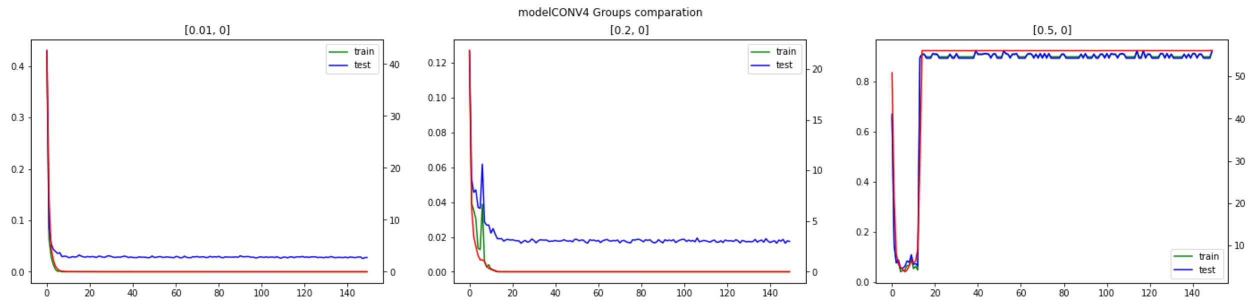
A:

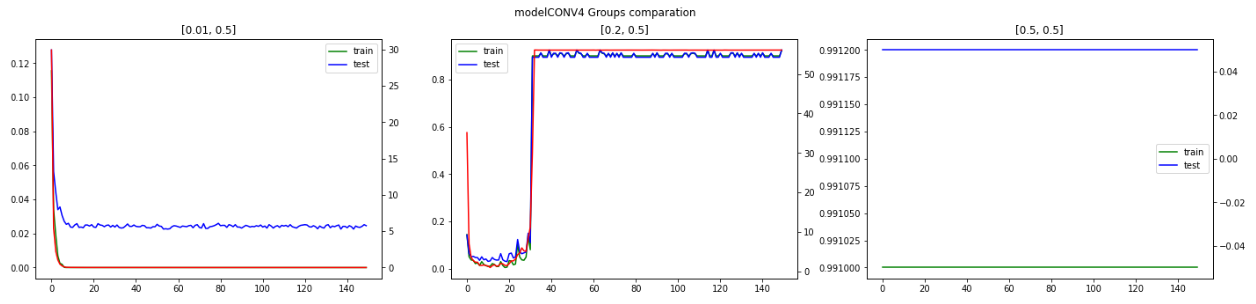
There are more figures matching the same number. For example, we have three pictures representing each number. Since we are using the sum of numbers of two pictures to obtain the target, each target now is related to more than nine pics since there can be so many combinations to obtain the target value. This will make it much harder for computers to learn the pattern. To them, the features become more irregular.

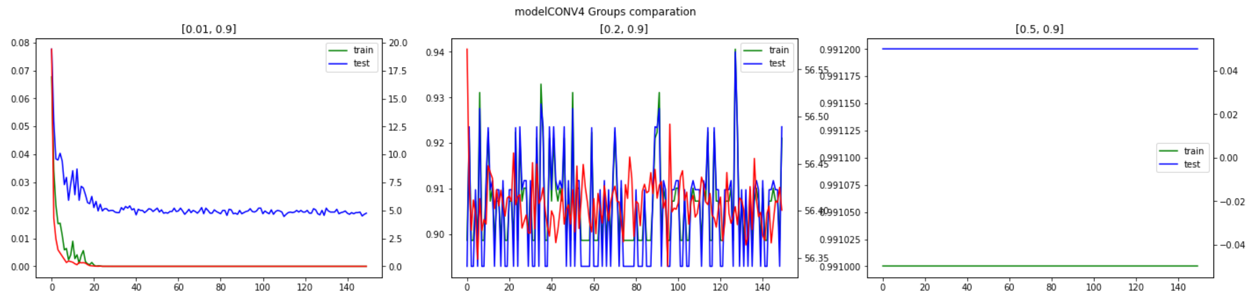
Model I for Q7
Accuracy on test set: around 0.035%



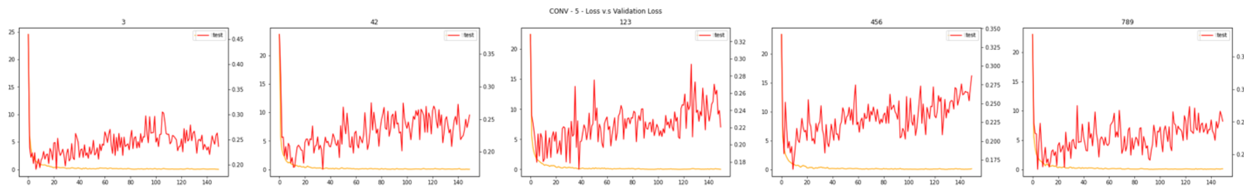


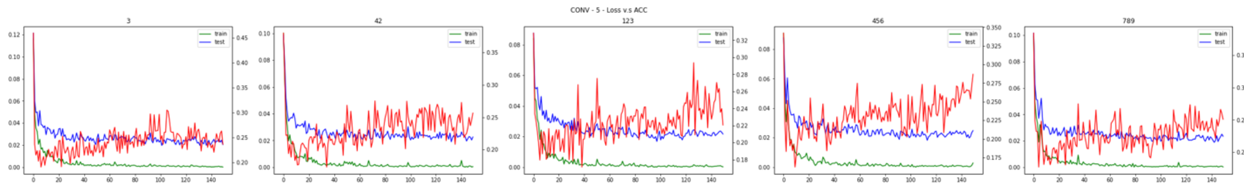


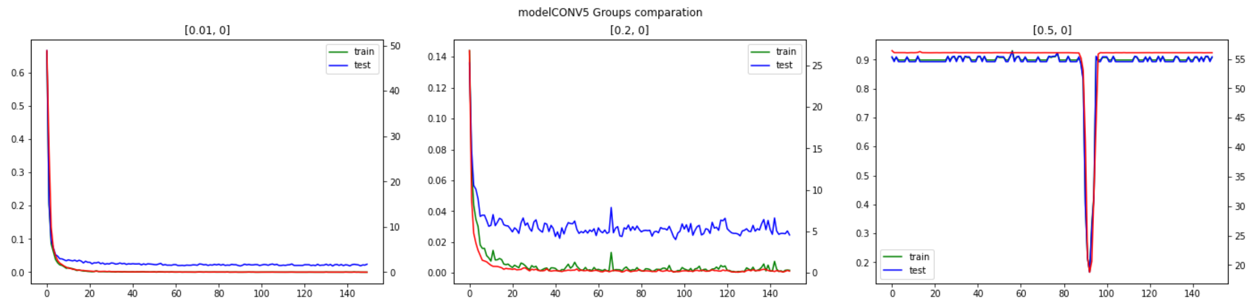


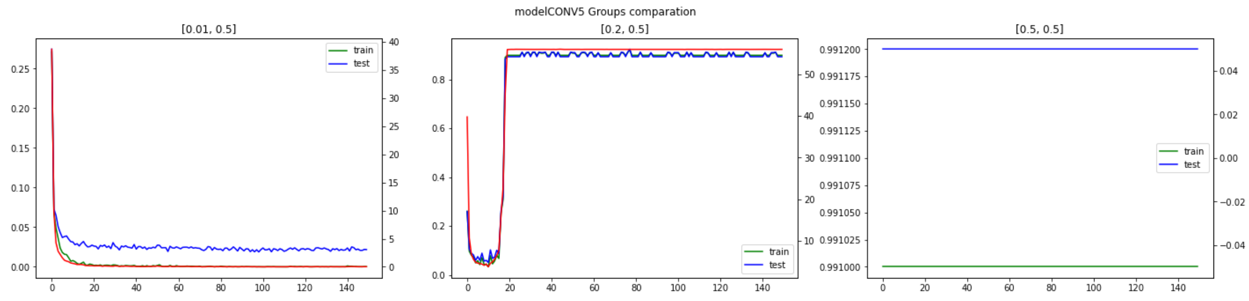


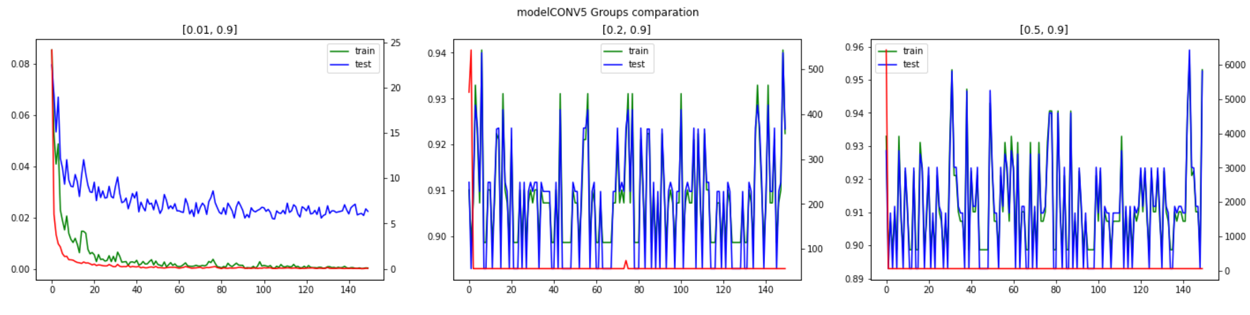
Model II for Q7
Accuracy on test set: around 0.038%


CONV - 5 - Loss v.s Validation Loss
3
42
123
456
789
test


CONV - 5 - Loss v.s ACC
3
42
123
456
789
train
test


modelCONV5 Groups comparation
[0.01, 0]
[0.2, 0]
[0.5, 0]
train
test


modelCONV5 Groups comparation
[0.01, 0.5]
[0.2, 0.5]
[0.5, 0.5]
train
test


modelCONV5 Groups comparation
[0.01, 0.9]
[0.2, 0.9]
[0.5, 0.9]
train
test