Problem Set 2

CMSC 828L

Eleftheria Briakou

October 8, 2018

1 Image classification approach

For both MNIST and Fashion MNIST datasets, I chose to experiment with simple cnn configurations, as shown in Figure 5. Following, I describe the basic structural units and hyperparameters of the system as well as the intuition behind their usage:

- **Feature extraction:** Convolutional layers produces output feature maps of the input image; the ReLu activation function is used at their outputs to introduce nonlinearities to the model. Given that the number of parameters that the model has to learn is large, I also used a pooling layer to downsample the dimensionality of the feature maps produced by the convolutional layer using the maximum operator, and finally I experimented with the dropout regularization to avoid overfitting.
- **Classification:** The second part of the network (fully connected with one hidden layer) uses the aforementioned features to classify the input image into one of the 10 classes (the softmax activation functions converts the outputs into probabilities).
- **Learning:** As our problem is a multi-class classification task I chose to minimize the categorical cross entropy, using the Adadelta()[1] optimizer from keras.
- **Hyperparameters:**
  - $(i \times i)$: the convolution filter
  - n: the number of filters
  - $k \in [0, 1)$: the dropout parameter
  - $(j \times j)$: the pooling filter
  - M: number of convolutional layers
  - H: number of nodes of fully connected layer[2]

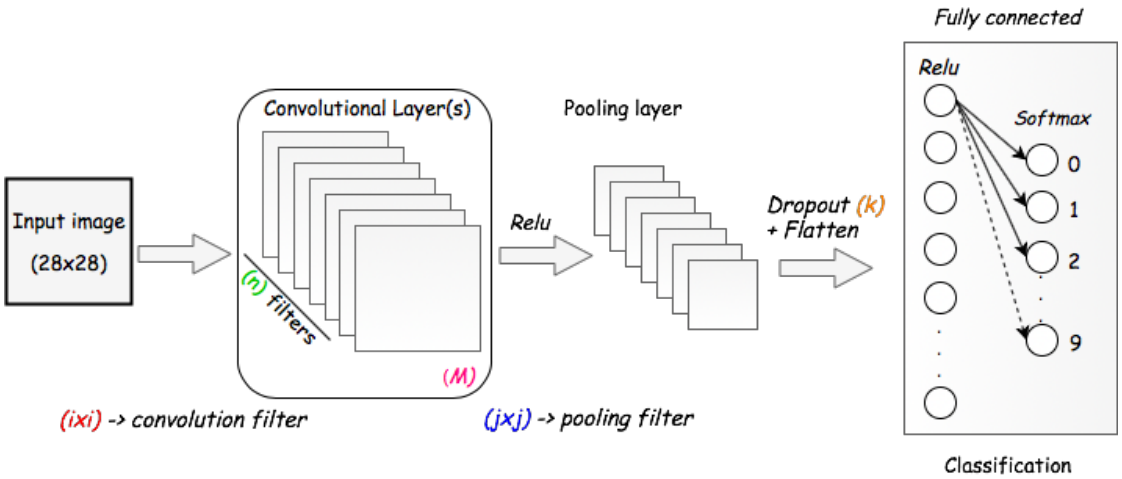


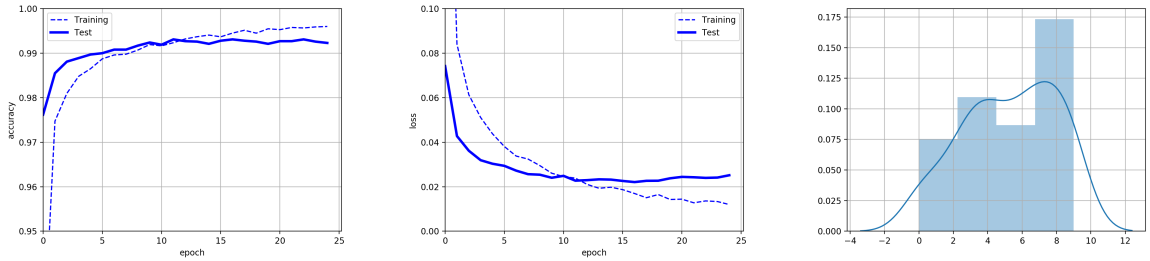
Figure 1: Abstract depiction of the basic structural elements and tunable hyperparameters of the neural network configuration used for image classification.

All in all, I tuned the hyperparameters of the above configuration using a grid search method for the two image datasets. In the following sections I present the results that I got for the best configurations. Plots of the thorough search could be found in the Appendices sections.

[1] Experimentation with other optimizers showed the adadelta performs better.

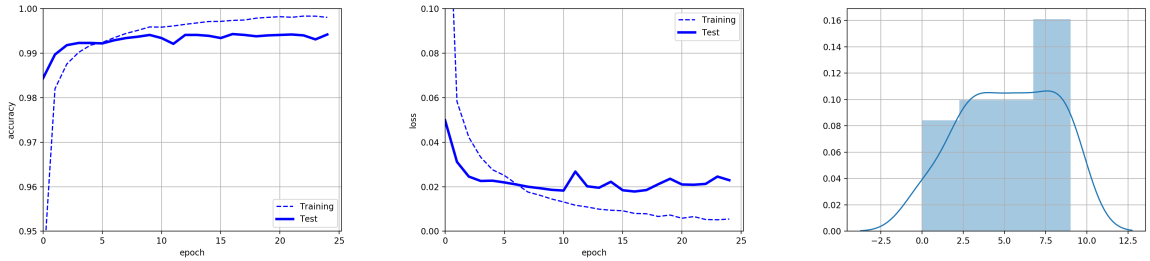
[2] The submitted code does not support a grid search for this parameter; for the following experiments its value is $H = 128$.

2 MNIST dataset



(a) Accuracy (b) Loss function (c) Distribution of misclassified

Figure 2: Results of best configuration, for $M = 1$ ($i = 7$, $n = 32$, $j = 2$, $k = 0.3$). Test accuracy: **99.23**%



(a) Accuracy (b) Loss function (c) Distribution of misclassified

Figure 3: Results of best configuration, for $M = 2$ ($i = 7$, $n = 32$, $j = 2$, $k = 0.3$). Test accuracy: **99.42**%

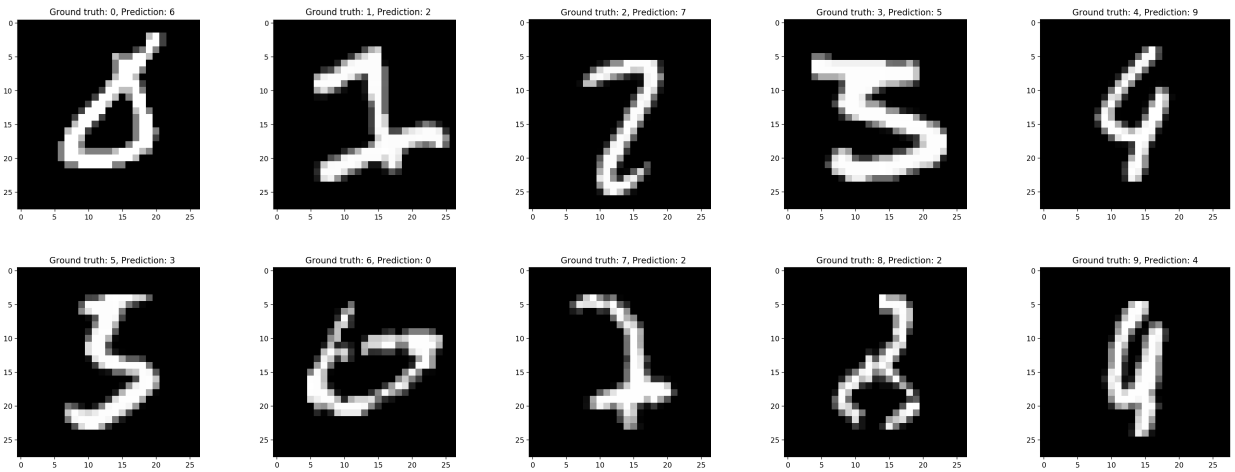


Figure 4: Misclassified examples from each cluster. Generally I noticed that most of the examples my neural network failed to correctly classify were difficult even for a human to label correctly. So, this failure is attributed to the intrinsic difficulty of the hand written recognition task, as there is a variety of different writing styles that could 'confuse' both the automatic task and humans.

Generally, the neural network configuration that I chose seems to work sufficiently well even when a single convolutional layer is used. The testing accuracy and loss do not deviate a lot from the training results, which indicates that with the right choice of hyperparameters I avoid overfitting. Moreover — except for the above analysis— I printed out the weights and biases of the final layer after training is completed to ensure that my neural network learns reasonable parameters. This information is presented in Figure 5.[3]

[3]This is also produced by the code; I will only include this for this dataset in my report.

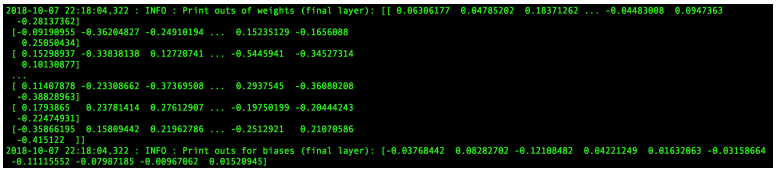
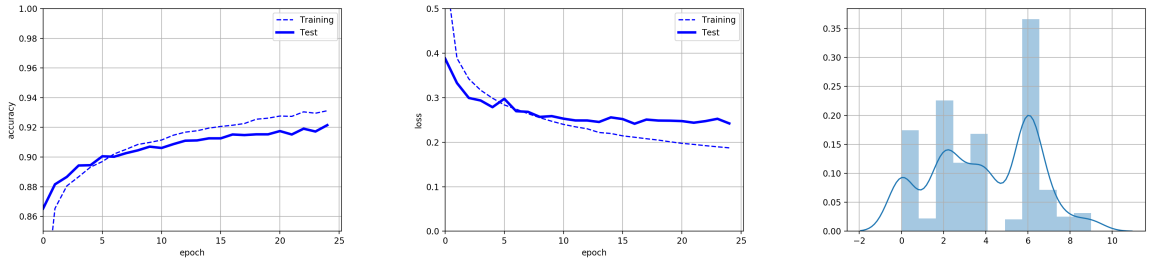


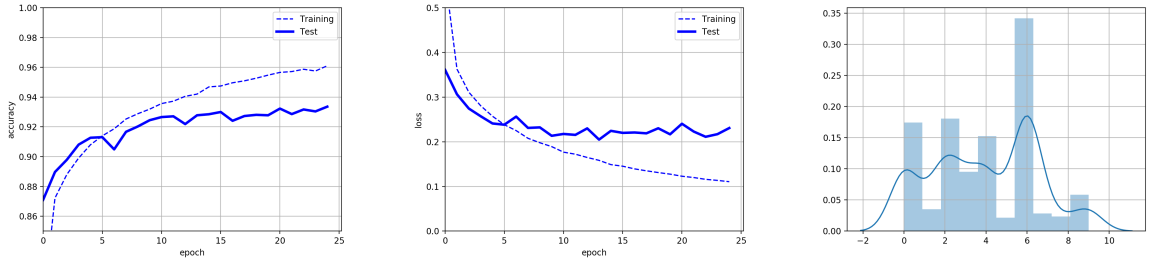
Figure 5: Representative print outs of weights and biases.

3 Fashion MNIST



(a) Accuracy (b) Loss function (c) Distriuubtion of misclassified

Figure 6: Results of best configuration, for $M = 1$ ($i = 3$, $n = 64$, $j = 2$, $k = 0.3$). Test accuracy: **92.15**%



(a) Accuracy (b) Loss function (c) Distriuubtion of misclassified

Figure 7: Results of best configuration for $M = 2$ ($i = 3$, $n = 64$, $j = 2$, $k = 0.3$). Test accuracy: **93.36**%

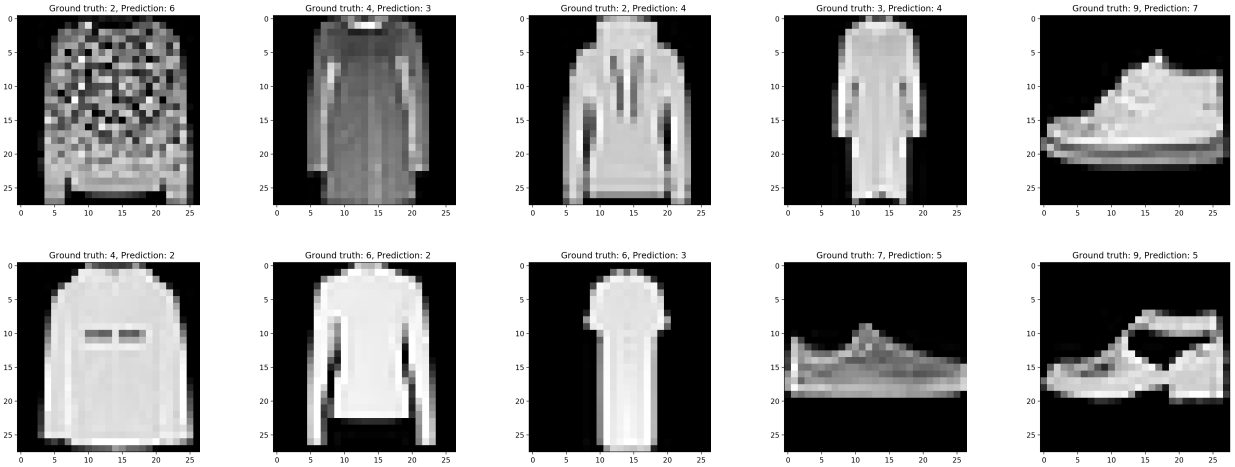


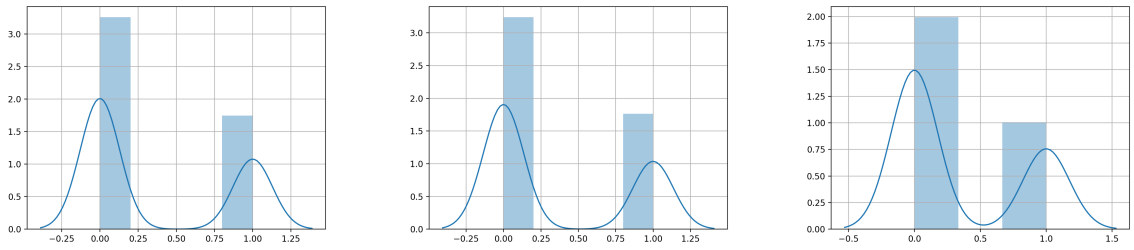
Figure 8: Misclassified examples from each cluster. Again, I notice that most of the misclassified examples were also hard to classify for humans. Encouragingly the wrong predictions are correlated with the ground truth values (e.g., instead of predicting that an image is a sandal it predicts a sneaker, which are both shoes).

Generally, the Fashion MNIST dataset is more challenging than MNIST. However, the same architecture produces satisfactory results for both datasets which implies that there is a correlation between them. It also worth mentioning though that different hyperparameters lead to the best results for each of them.

4 Breast Cancer

For this task I had to pre-process the data (normalize them) and split them into training and test subsets. To gain an inside into the data, I also plotted the distributions of the subsets over the two candidate classes, as shown in Figure 9.

4.1 Data split



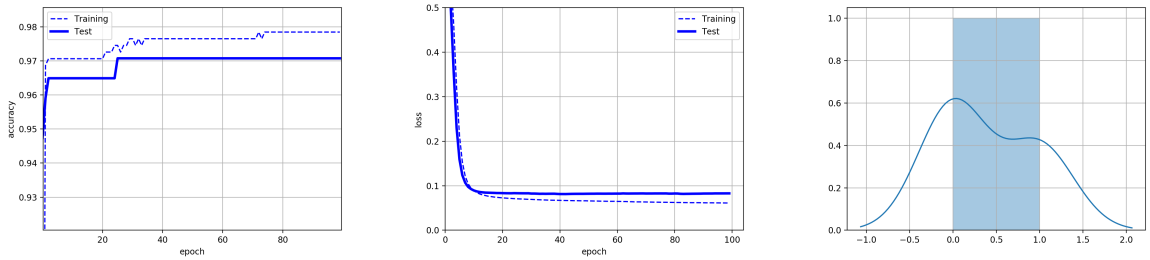
(a) Dataset (b) Training (80%) (c) Test (20%)

Figure 9: Distribution of breast cancer dataset and training and test subsets over the two candidate classes. Generally, I notice that the dataset is unbalanced among the two classes; this imbalance should also be reflected to both the training and test sets.

4.2 Neural network

For this task a chose to experiment with a fully connected neural network with ReLu activation functions to introduce nonlinearities and a final softmax output function which produces classes probabilities. The loss function which was chosen is the binary cross entropy, with Adadelta() optimizer. Given that the number of features (9) and the complexity of the problem (binary) is small, a simple hidden layer was found to be enough to produce satisfactory results in terms of accuracy.[4] The hyperparameters that I tuned for this experiments are:

- N: number of nodes of hidden layer
- b: batch size
- l: number of hidden layers



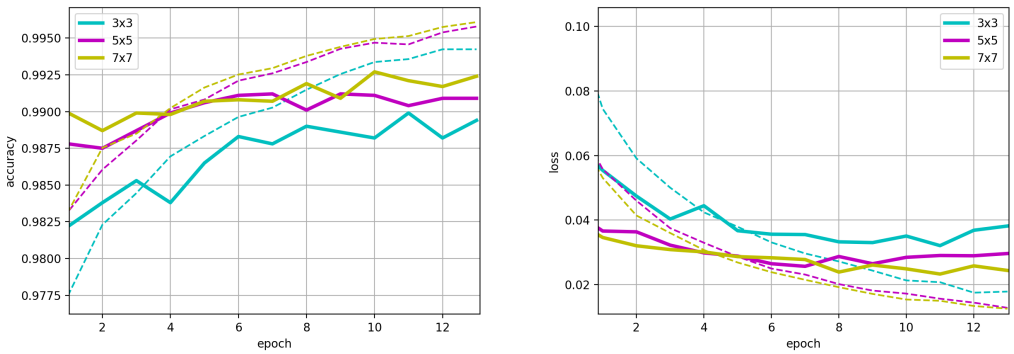
(a) Accuracy (b) Loss function (c) Distriubution of misclassified

Figure 10: Results of simple cnn configuration ($N = 30$, $b = 100$, $l = 1$). Test accuracy: **97.08**%

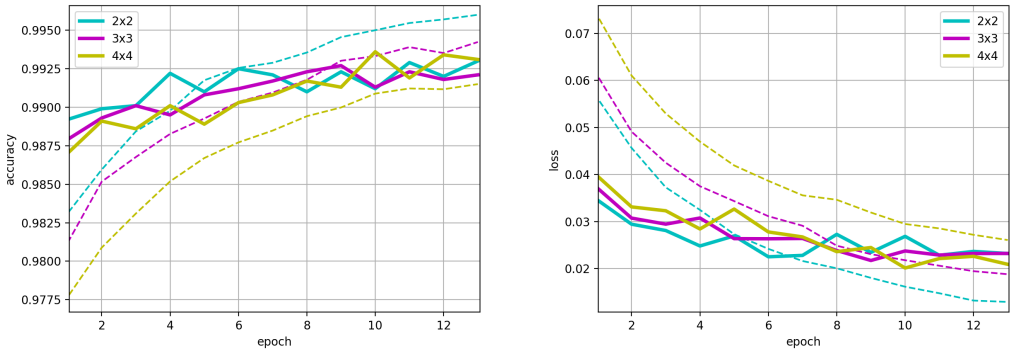
Finally, as dipicted in Figure 10 (c) the imbalance of the dataset is also depicted in the distribution over the misclassified classes, as we notice that there are more examples incorrectly classified as benign (0) than malignant (1).

[4]Ideally precision-recall are better metrics for this task, since there is also a class imbalance, but these metrics are currently removed from keras.

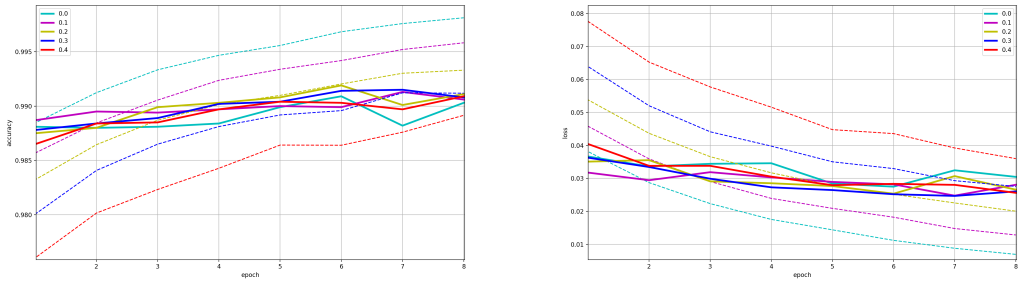
4.3 Appendix A: Hyperparameter tuning (on MNIST)



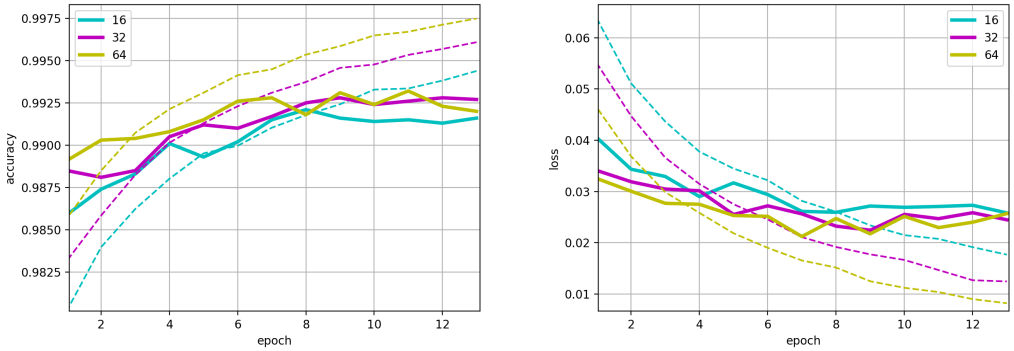
(a) Different filter sizes.



(b) Different pooling sizes.



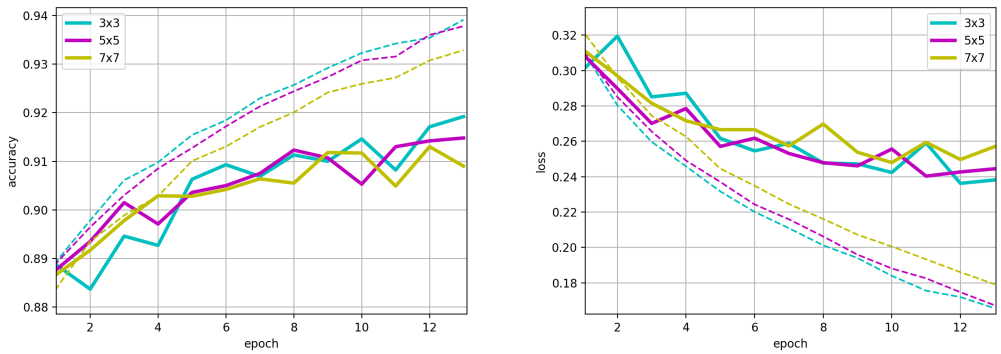
(c) Different dropout configurations.



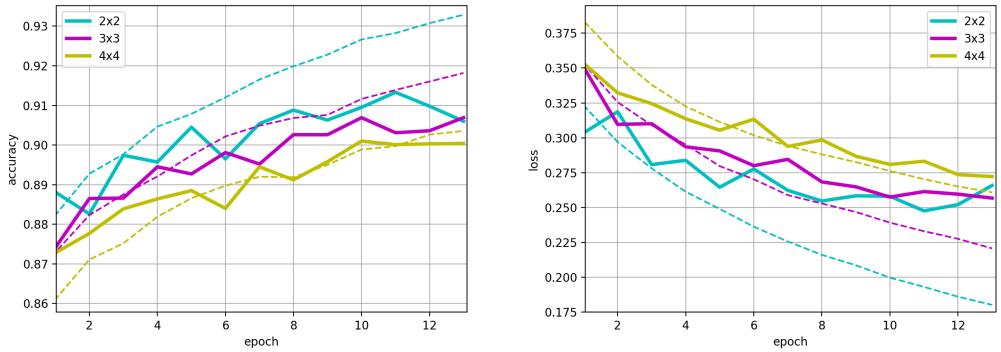
(d) Different dropout configurations.

Figure 11: Tuning hyperparameters of cnn layer for MNIST classification. Dashed lines correspond to accurancies and losses of training sets; continuous lines correspond to test sets.

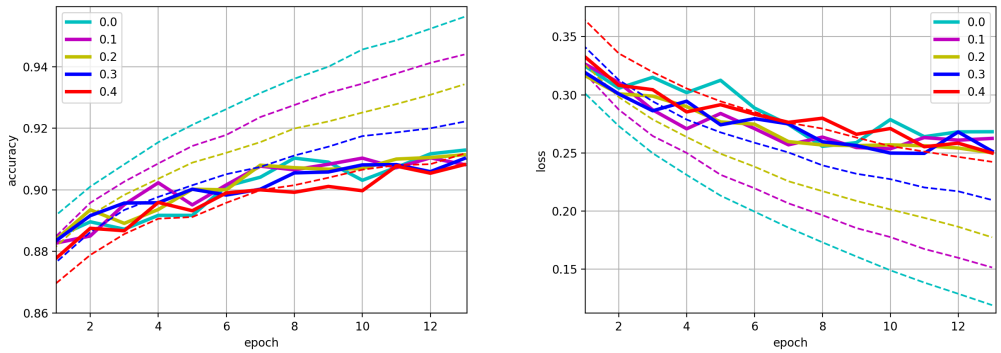
4.4 Appendix B: Hyperparameter tuning (on FMNIST)



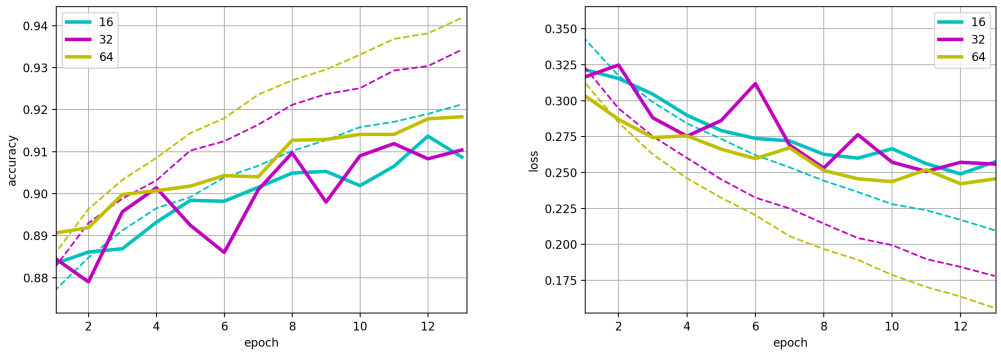
(a) Different filter sizes.



(b) Different pooling sizes.



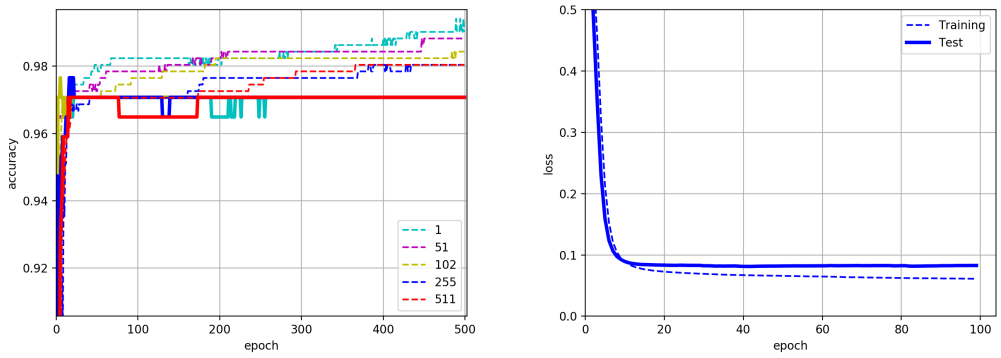
(c) Different dropout configurations.



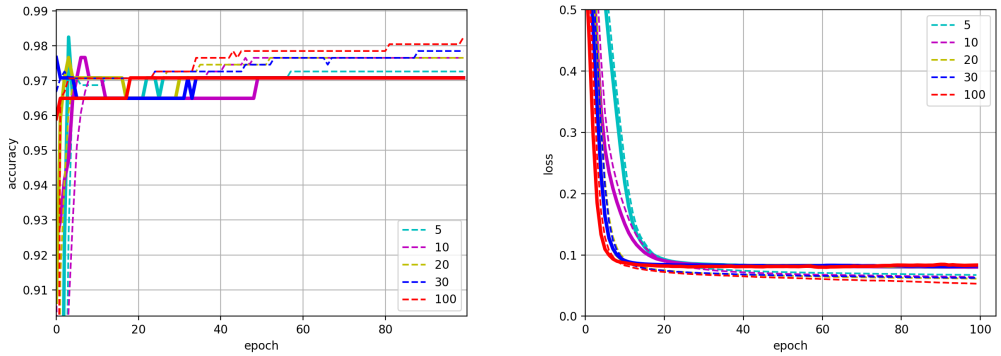
(d) Different dropout configurations.

Figure 12: Tuning hyperparameters of cnn layer for FMNIST classification. Dashed lines correspond to accurancies and losses of training sets; continuous lines correspond to test sets.

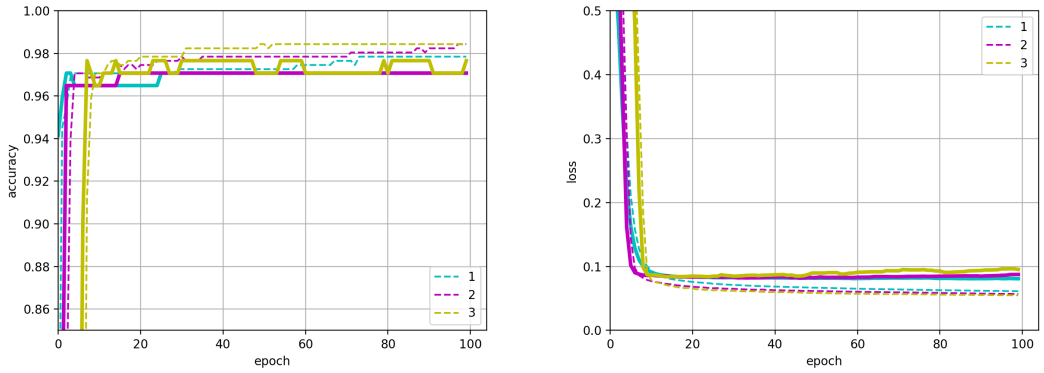
4.5 Appendix C: Hyperparameter tuning (on Breast Cancer)



(a) Different batch sizes (b).



(b) Different node sizes for hidden layer (N).



(c) Different number of hidden layers (l).

Figure 13: Tuning hyperparameters of cnn layer for breast cancer classification. Dashed lines correspond to accurancies and losses of training sets; continuous lines correspond to test sets.