First, we choose Convolution Neural Network (CNN) for this assignment as we are classifying MNIST-like dataset. We want to use CNN, not only it is widely acknowledged for high accuracy when classifying images, but also because we already had some experience with classifying images like this. So in short, CNN can help us to extract important features of the images to help increase accuracy while classifying.

Our model is inspired from the VGG architectures. In details, our model consists of:

3 x (2 x Convo2D + Max Pooling + Dropout) + Flatten + Dense + Dropout + Dense(output)

The first 3 blocks have two convolution layers following by a max pooling and a dropout. Details arguments for these three blocks:

1. The number of output filters in each blocks are 32, 64, and 128

2. All block use kernel size 3 x 3. We want to retrieve as much information as possible. Also, 5x5 gave worse result, about 0.3% from our best score.

3. Activation functions are all ReLu, a very common and effective in CNN

4. Kernel initializer: We use He uniform since this has been proved to work very effective with ReLu

5. Padding: same. We use this because the details at border of the images are sometimes important (important features not always in the center of the image). Also, we want to preserve the spatial size to nicely decrease the output size.

After 3 VGG-like block, I put one more dense layer of 128 nodes before the output layer. This layer uses the same activation function and kernel initializer. Lastly, the output layer consists of 10 nodes, using softmax as activation function. All dropout are 0.2, we tried some surrounding values like 0.3 or 0.4 but 0.2 appears to work well. The dropout here is for preventing overfiting.

We did not preprocessing the train data much. The only thing we did is normalize the data from grayscale to [0, 1].

Since this is the classifying problem, we choose the optimizer to be SGD with learning rate of 0.01 and momentum of 0.9. We tried to reduce learning rate to 0.001 and increased the number of epochs, but somehow that did not work better than 0.01. Loss is chosen to be “categorical_crossentropy” and metric being used is “accuracy.”

We trained the model with 150 epochs and 20% of the train data to be validation data. Batch size is 64. To compare the accuracies between trials, we made a script to compare the prediction from different model. We assume the total test data is 10000, so if the difference between our best

result and another output is greater than 10 or 20, we would say that other output will give worse score.