# Report

We are building a model using the Multi-layer Perceptron (MLP) Network with the following parameters:

**1] Input Size = 784**
- Since we have 784 features per data point

**2] Number of Neurons in the Hidden layer = 100**
- The network is able to achieve a good accuracy using 100 nodes in combination with the rest of configuration; and results in a good cumulative loss which leads to > 98% accuracy for the validation set.

**3] Number of Neurons in the Output layer = 4**
- Since the output labels are one-hot encoded with 4 values.

**4] Learning Rate = 0.2**
- After trying out different values, we found 0.2 to be most effective minimum learning rate.
- The network could steadily approach the minima, as opposed to higher values, which resulted in greater fluctuations in the cumulative error between different epochs.

**5] Activation Functions:**
- **For Hidden Layer: Sigmoid**
  Which was the ideal function to be used at the hidden layer
- **For Output Layer: SoftMax**
  The output is in one-hot encoded format, with only 1 of the 4 indices holding the value '1'. Hence it makes sense to use SoftMax here, which would normalize the output to a network of probability distribution.

**6] Error/Loss Function: Cross-entropy Loss Function**
- For one-hot encoding, cross-entropy implies that the output label should hold zeroes at all the indices, except one.
- This works best for our configuration, which uses SoftMax as the activation function at the output layer.

**7] Maximum Epochs allowed = 30**
- 30 epochs allowed the network to reach a good-enough cumulative loss value, by not completely exhausting the CPU resources with the given configuration.

**8] Training Size = 90%, Validation Size = 10%**
- We use 90% of the training data for training and 10% for validation. The held-out validation set is used to test the network on unseen data and helps us choose the optimal hyper-parameter setting for our network.

**9] Individual Acceptable Loss = 0.001**
- Used to calculate the acceptable Cumulative Loss, depending on the size of the input set.

The model can be configured differently by changing the parameters passed to constructor (of MLP class object) in '__init__.py' (Please refer this file to understand how the model is instantiated).

After training the model, it was able to achieve 98.22% accuracy on testing the validation set.