CS 574 - Computer Vision Using Machine Learning

PROJECT REPORT
ASSIGNMENT 1

140101008 - AMOGH SHANKAR GUPTA
140101023 - GAURAV MANCHANDA
140101039 - MAULIK PATNI
140101045 - NITISH GARG
140101063 - SAMBHAV KOTHARI

Problem Statement

Use the following machine learning algorithms to classify the given image sample from MNIST database into one of the 10 possible classes(0 to 9):

1. Softmax regression without convolutional layers
2. Softmax regression with convolutional layers

Further, Exercise on various parameters such as number of hidden layers, type of activation functions, number of convolution kernels, size of convolution kernels. Also exercise on over-fitting problem with possible solutions.

Language / libraries used

- Programming language - Python 3
- Libraries
    - We used **Keras** API for building all neural network models. Keras uses **Tensorflow** and **Theano** libraries in its backend.
    - **Matplotlib** was used for visualisation of results using graphs.

Studying the dataset

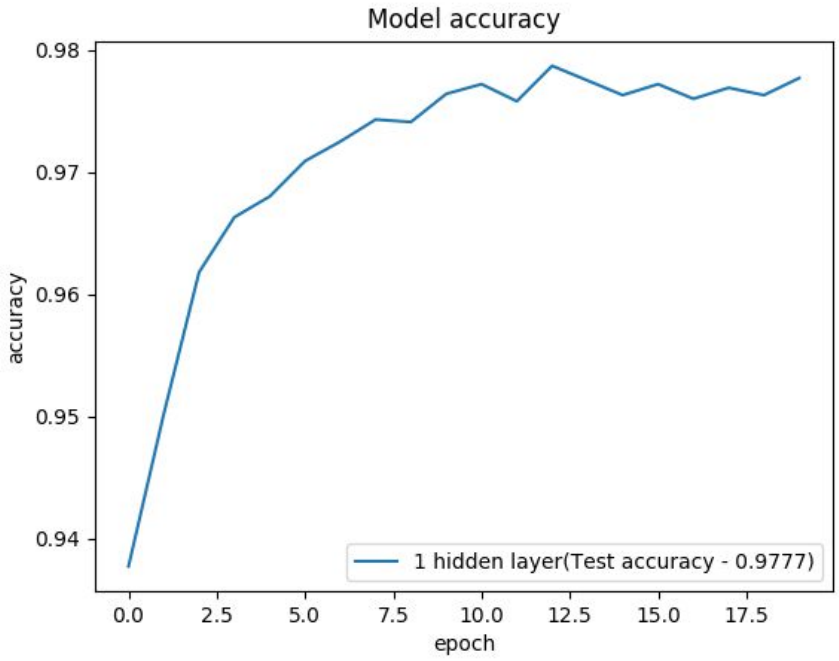
The MNIST data comes in two parts. The first part contains 60,000 images of numerical digits to be used as training data. The images are greyscale and 28 by 28 pixels in size. The second part of the MNIST data set is 10,000 images to be used as test data. Again, these are 28 by 28 greyscale images. We'll use the test data to evaluate how well our neural network has learned to recognize digits. We regard each training input as a 28×28=784-dimensional vector. Every image in the dataset is labelled with its correct classification - 10 classes (digits 0-9).

Since the assignment involved building 2 models, one without convolutional layers and one with convolutional layers, we study both the models one by one carrying out experiments to figure out appropriate hyper-parameters by plotting accuracies and observing how it varies with different parameters.

Softmax regression without convolutional layers

We first build a simple model with the following properties and then analyze its performance by varying different parameters.

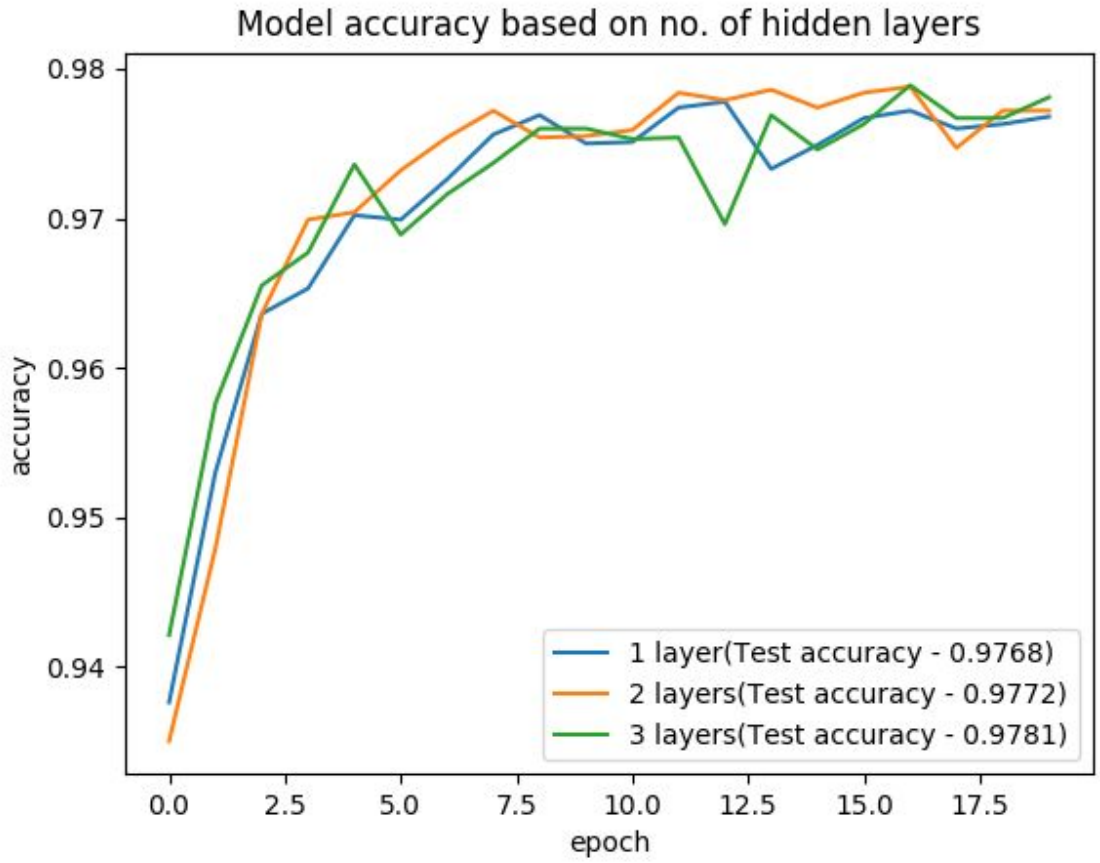
- Neural network with a single hidden layer consisting of 100 neurons and 784 neurons in the input layer and then finally 10 neurons in the output layer.
- We use softmax as the activation function. The fact that a softmax layer outputs a probability distribution is used here to get the probabilities of an image to be in one of the 10 categories.
- Batch size of 200 is used and model is run for 20 epochs.



We get a fairly good accuracy of 97.77% on the test data. Now, we test out different parameter types and values in a hope to get better results.

Number of hidden layers

We train and evaluate our model by using 1, 2 and 3 hidden layers and plot out the results. Number of neurons in the first, second and third hidden layers are 100, 50 and 30 respectively.



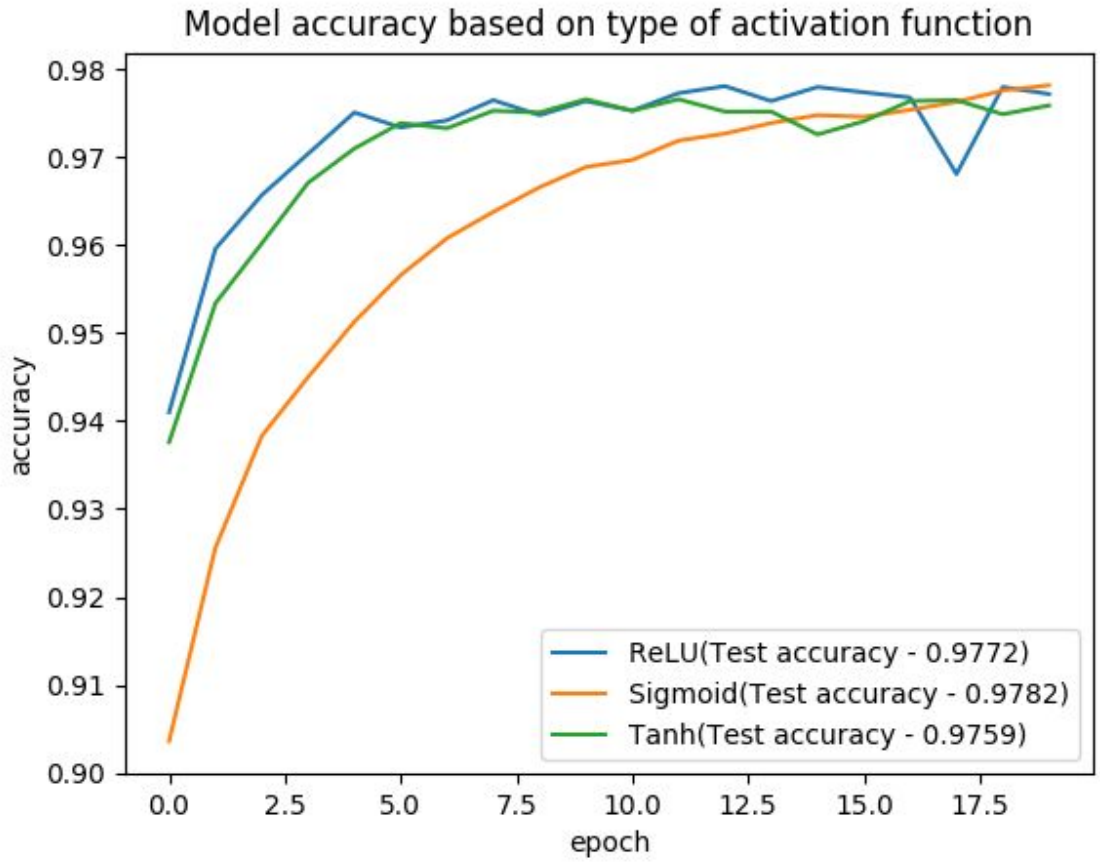
We see that on increasing the number of hidden layers, the accuracy of our model improves slightly. This is as expected because increasing the number of layers means our model can take more complex decisions. Every subsequent layer can take decisions by weighing up decisions given by its previous layer, resulting in more sophisticated decision-making process.

Number of epochs

An epoch basically consists of one forward pass and one backward pass of all the training examples. No. of epochs has a significant impact on the performance of our network. Usually, less number of epoch results in underfitting while large epochs result in overfitting of data. Hence, it is very important to figure out correct amount of epochs for a given model to have a good accuracy.

Type of activation function

Since we prefer the output to be probabilistically distributed in 10 classes, we let the last layer, i.e, the output layer to have softmax as its activation function. However, we study 3 different activation functions on our hidden layers (2 hidden layers are used in this example) - Sigmoid, Tanh and ReLU.

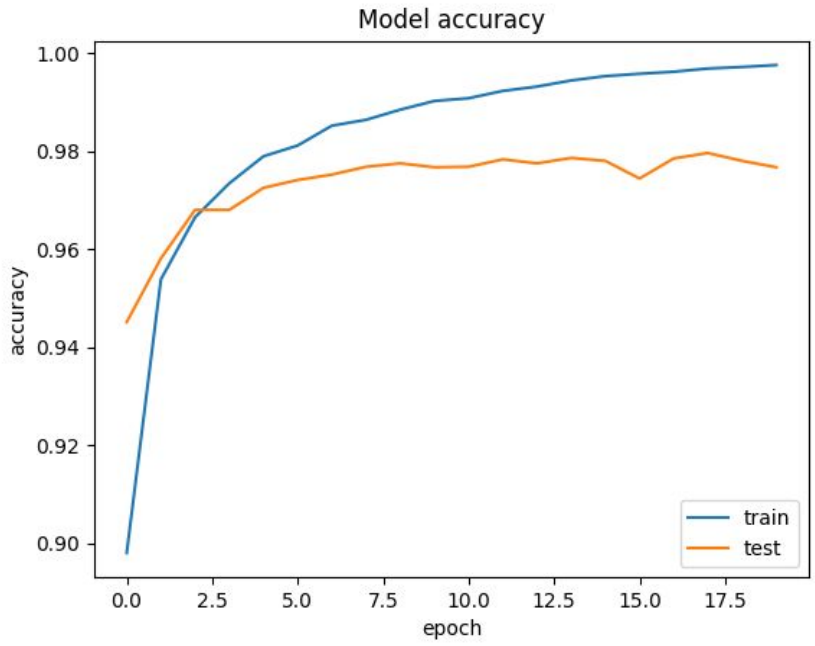


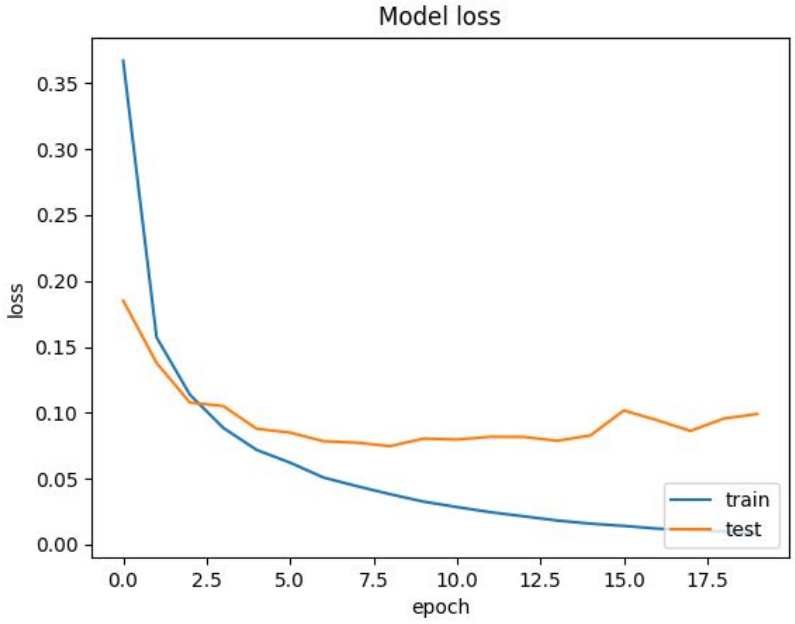
As expected, the model using sigmoid activation function in its hidden layers learns very slowly as compared to the other models. This is because the partial derivatives of the sigmoid function tends to be zero at extremes, resulting in learning slowdown.

Overfitting problem

Overfitting occurs when our model does not generalize well to real-world cases although it fits the training data well. It happens when the model's complexity is unnecessarily

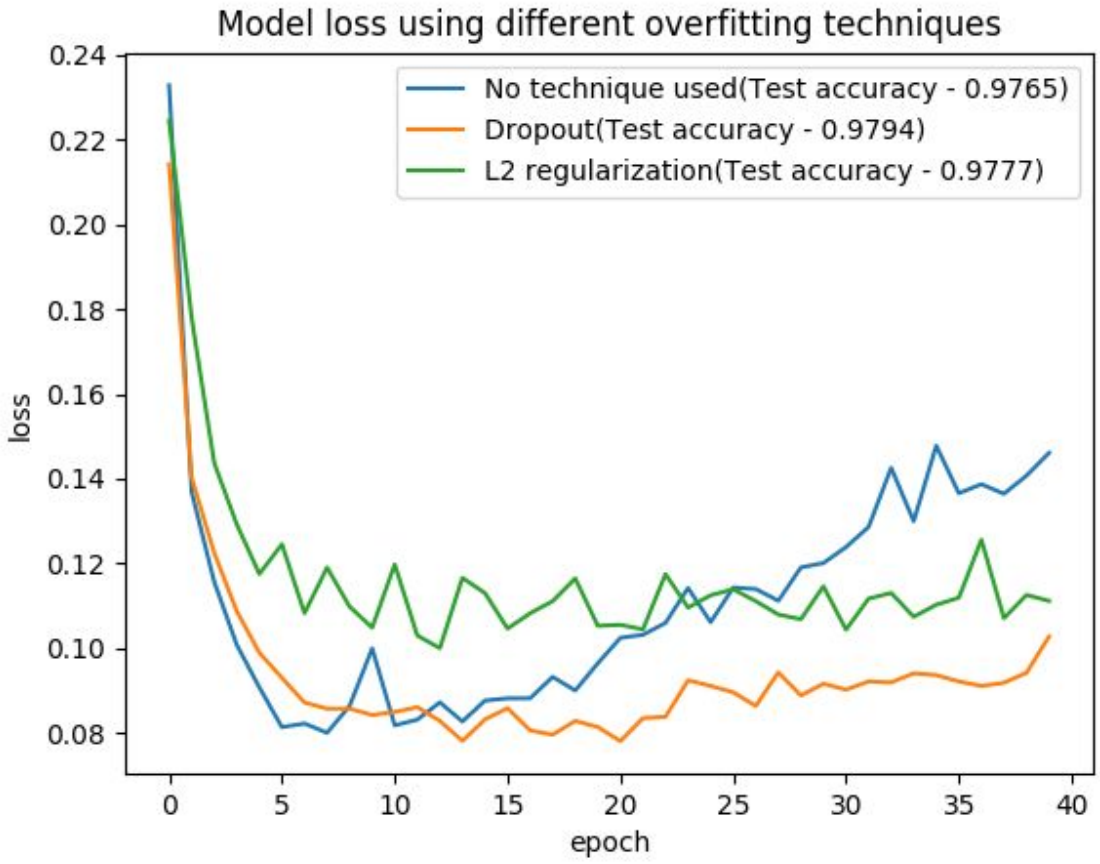
increased or it is trained for high epochs than required. After a point of time, test accuracy ceases to increase as shown in the below figure.





We study two methods to remove the problem of overfitting of data -

- **Dropout** - Dropout is a technique where we modify the network by temporarily removing few of the hidden nodes and carrying out feedforward and backpropagation on the modified network.
- **L2 regularization -** It is a regularization technique that works on sum of the squares of weights. Regularization can be viewed as a way of compromising between finding small weights and minimizing the original cost function.

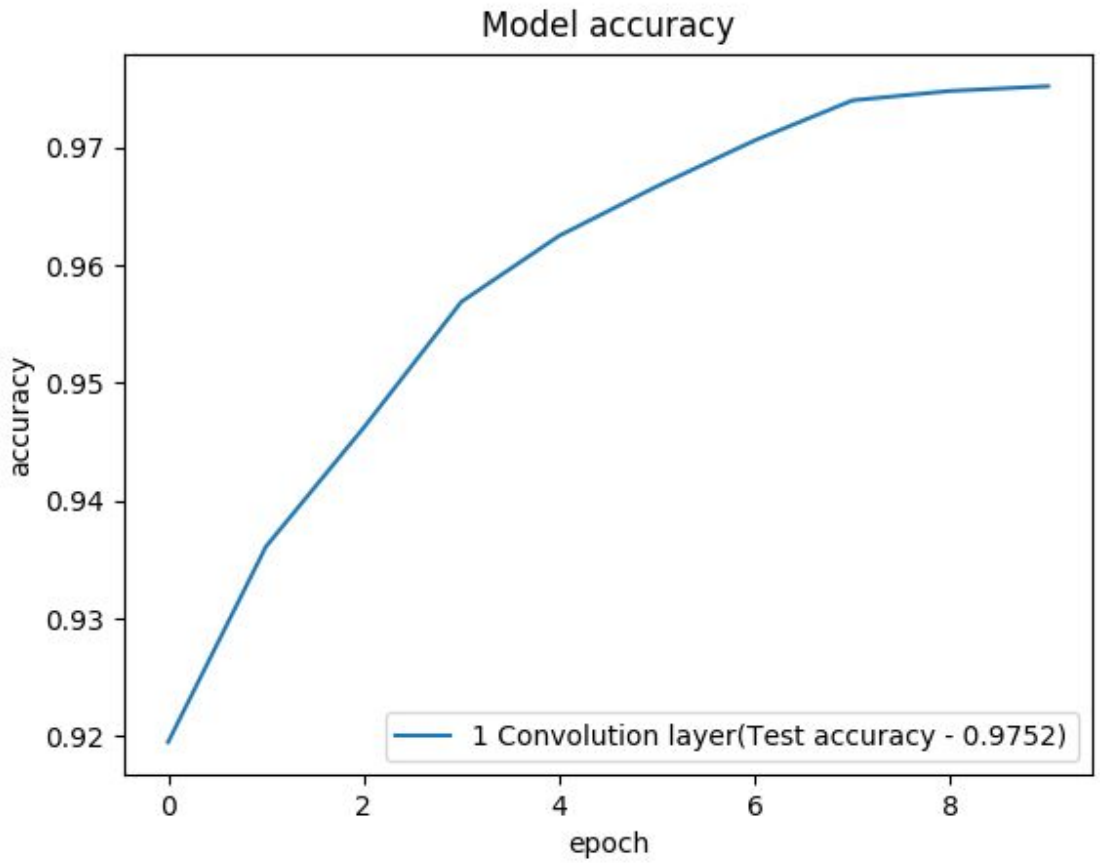


Using regularization or dropout reduces the loss in test data which is clearly visible in the above plot. Higher loss means higher level of overfitting.

Softmax regression with convolutional layers

Our initial model has the following properties -

- Neural network with a single hidden convolutional layer with kernel size 3x3 and 32 number of kernels or feature maps.
- We use softmax as the activation function of the output layer for the same reason as in the previous model. ReLU activation function is used for hidden layers.
- We also add a 2D max pooling layer of 2x2 over our convolutional layer.
- Batch size of 128 is used and model is run for 10 epochs.

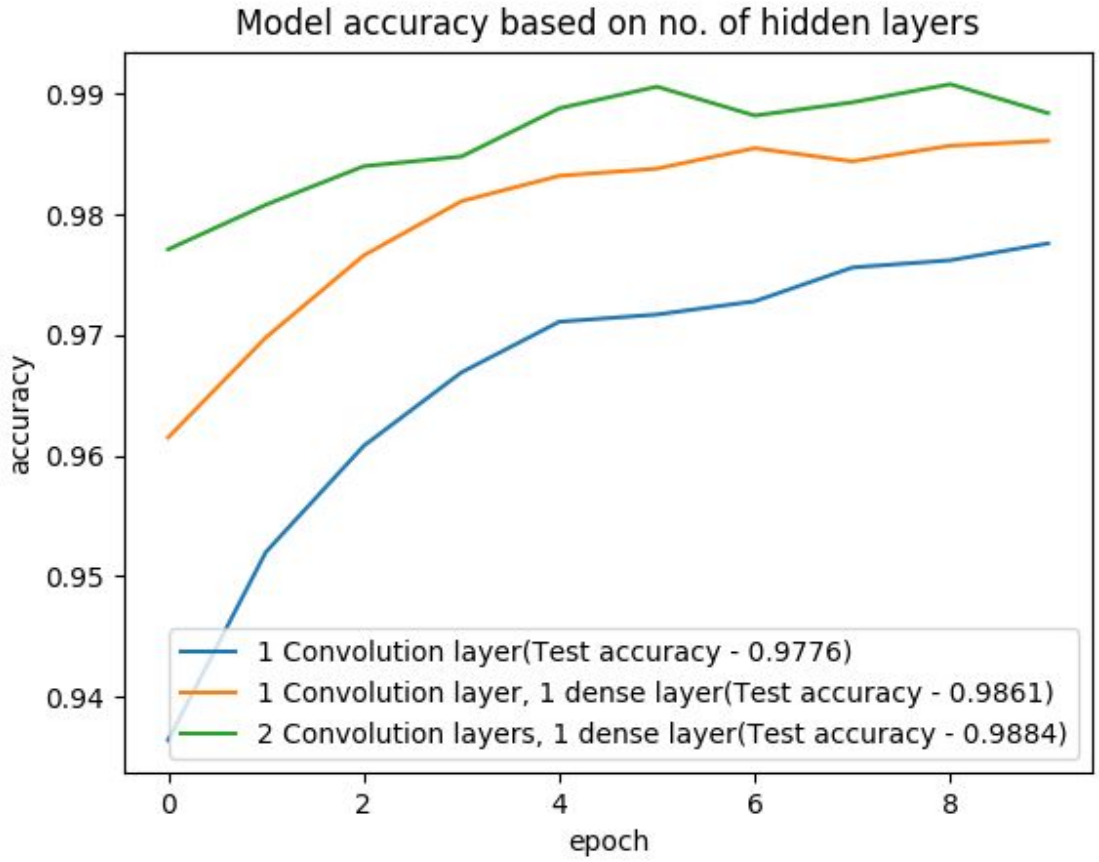


We get a fairly good accuracy of 97.52% on the test data. Now, we test out different parameter types and values in a hope to get better results.

Number of layers

We try with three different combination of convolutional and dense layers -

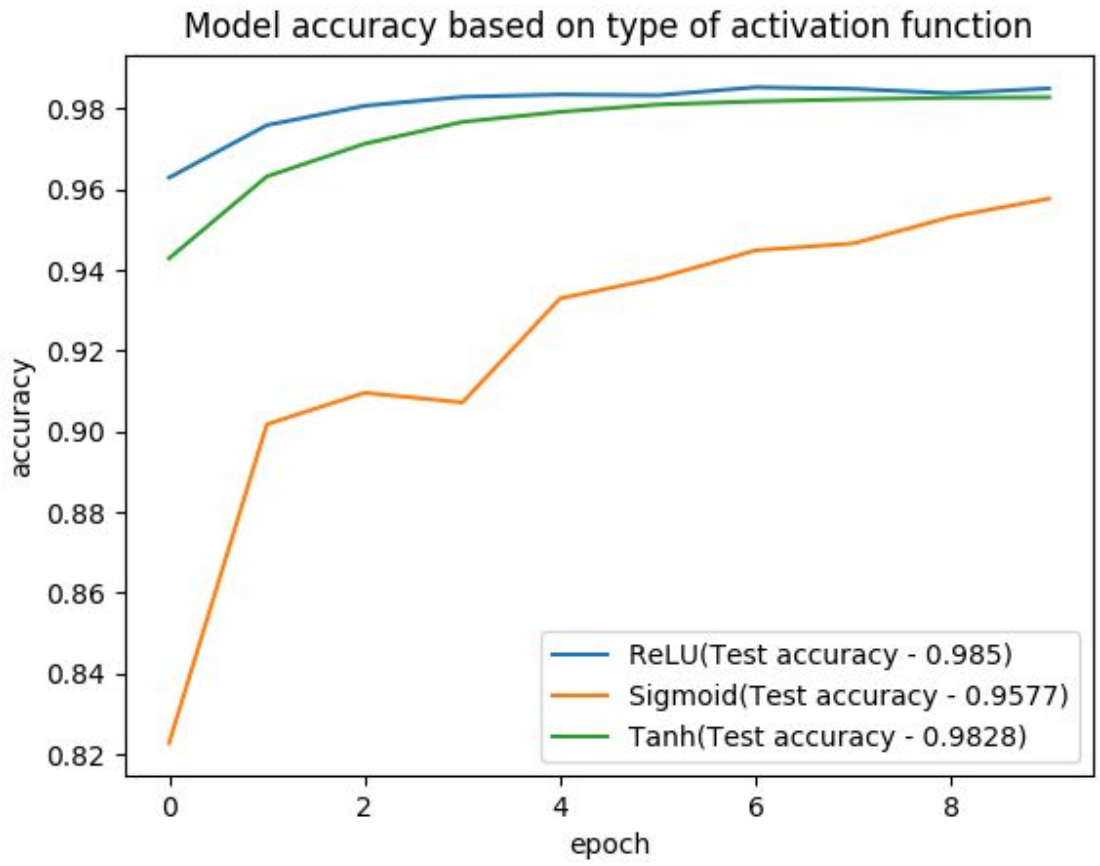
- Single convolutional layers
- Single convolutional layer (32 kernels of size 3x3) with a dense layer having 128 neurons.
- 2 convolutional layers (second one with 64 kernels of size 3x3) and a dense layer similar to previous case.



As expected, the third model performed better due to increase in convolutional layers.

Type of activation function

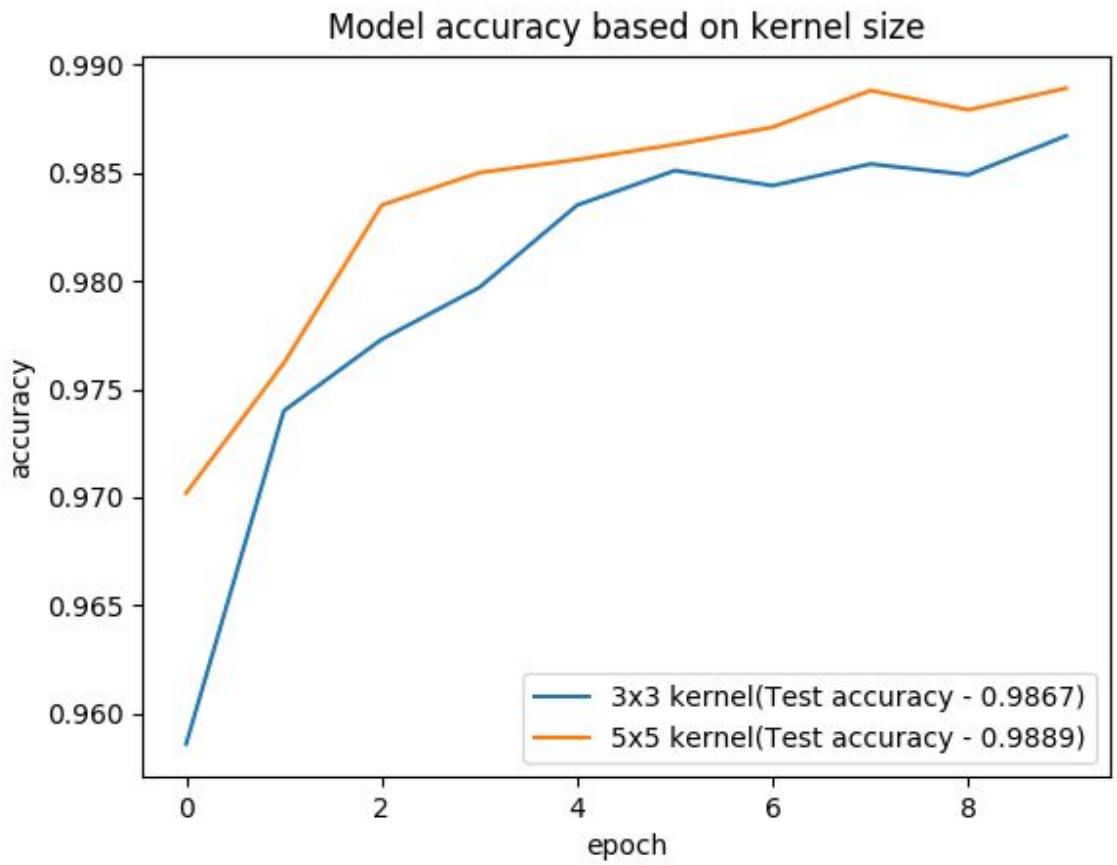
Just like the first model without convolutional layer, here we again study different activation functions in hidden layers with softmax being the one present at the output layer.



Here again, Sigmoid performs the worst and is a slow learner.

Kernel size

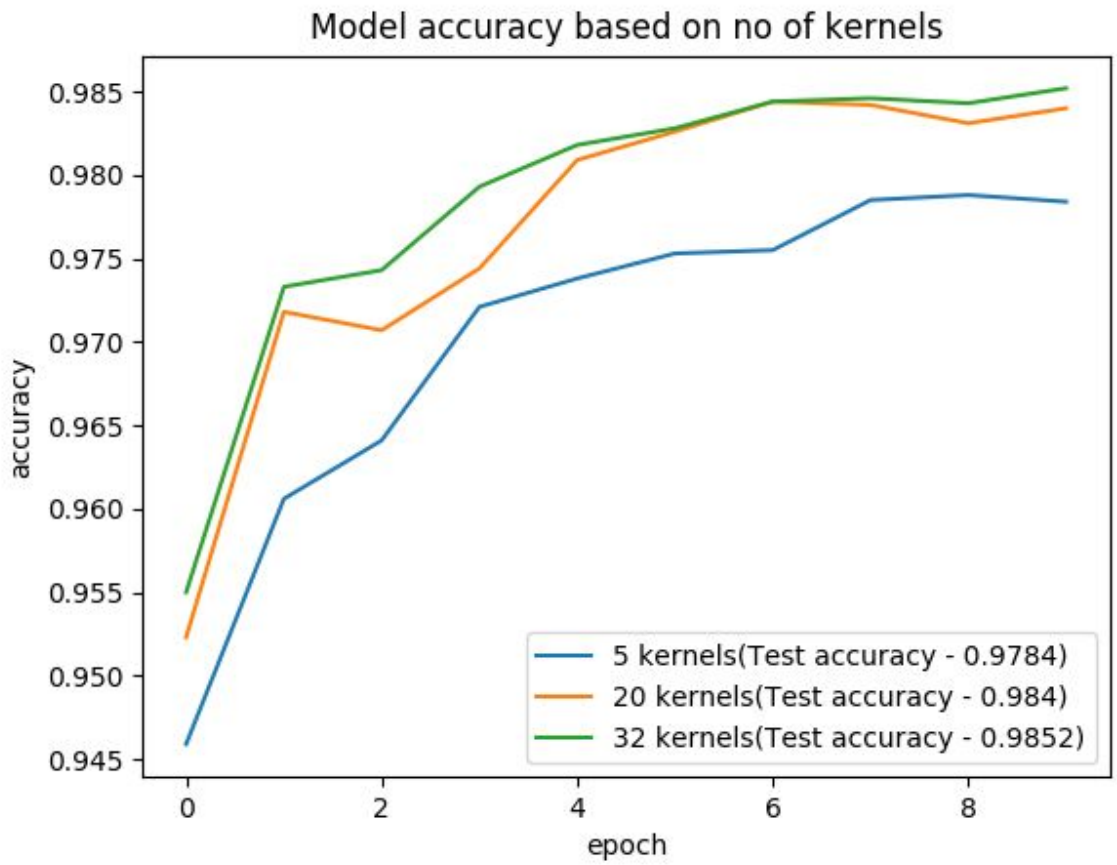
We experiment on 2 different kernel sizes, one being 3x3 and the other being 5x5.



We observe that accuracy for 5x5 kernel is better. Size of the filters play an important role in finding the key features. A larger size kernel can overlook at the features and could skip the essential details in the images whereas a smaller size kernel could provide more information leading to more confusion. Thus there is a need to determine the most suitable size of the kernel/filter. For the given data set of images (28x28) in size, 5x5 seems to be the most suitable size in effectively determining the details.

Number of kernels

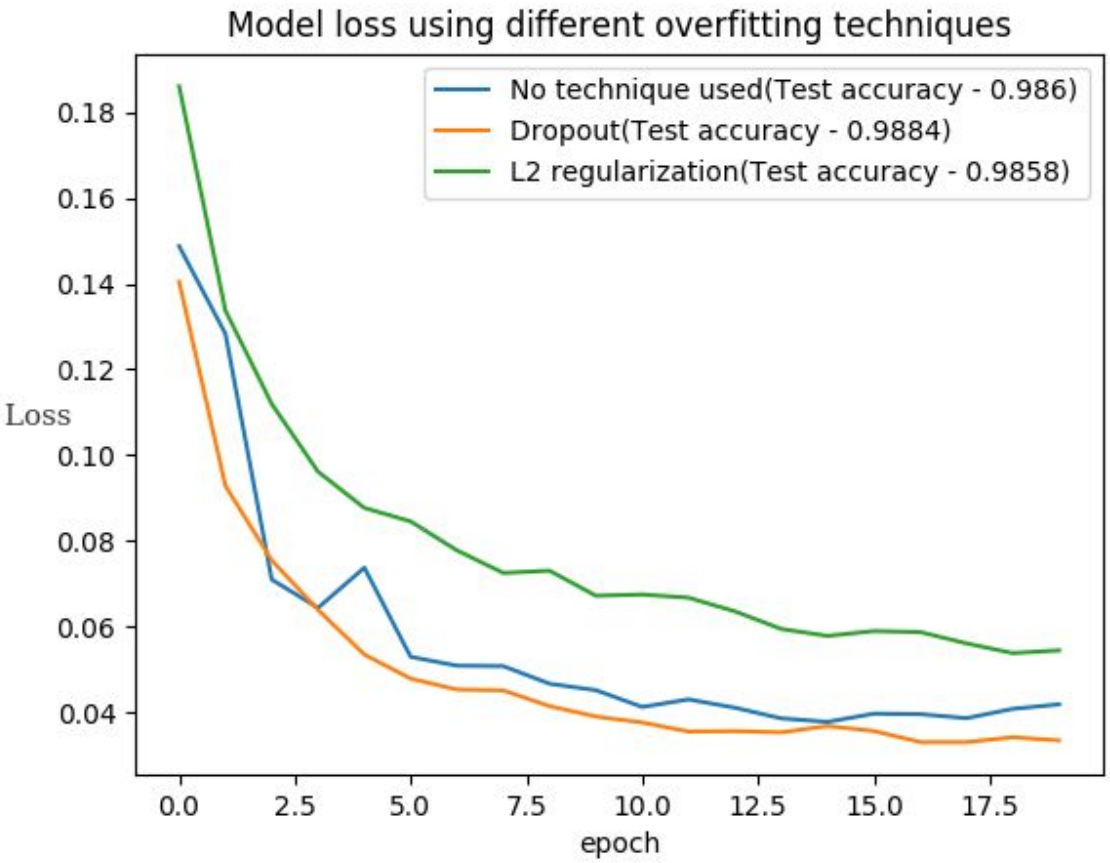
We experiment with 5, 20 and 32 feature maps / kernels in our model.



The more complex the dataset is, it is expected that networks with more kernels perform better. Intuitively, number of kernel at layer layer expected to bigger in the previous layers, as number of possible combination grow. That is why, in general, first layer kernels are less than mid-high-level ones. As can be seen from the plot, 5 kernels are way too less for our data and the accuracy increases on using higher number of kernels as expected.

Overfitting

Overfitting problem might arise in CNNs too, because of more or less the same reasons. We again try the 2 techniques to reduce overfitting - Dropout and L2 regularisation. Note that we apply dropout on only the dense layer of the network and not convolutional layers. This is because the shared weights in convolutions mean that convolutional filters are forced to learn from across the entire image. This makes them less likely to pick up on local idiosyncrasies in the training data.



Using dropout reduces the loss in test data which is clearly visible in the above plot. Higher loss means higher level of overfitting.

Conclusion

- As the training data is fed multiple times to the neural net, the model becomes more accurate as can be seen from the fact that accuracy improves with each epoch. Thus, there is a trade-off between time and accuracy.
- Convolutional Neural Network perform better as they take into consideration the features in image and make use of common patterns across it.
- Tuning of hyper parameters should be done on validation data.
- Since a lot of techniques and models are heuristic in nature, models and parameters are very much dependent on the type of problem.