EE569 Project 5

Name: Shreyas Dinesh Patil
Email: shreyasp@usc.edu

Tools used: Pytorch, Google Colaboratory
Problem 1: CNN Training and Its Application to the MNIST Dataset

Motivation:

Convolutional neural network is a special kind of neural network for processing data with a grid like topology. Image data is one of the examples of grid like data. CNN use special kind of linear operation called convolution instead of general matrix multiplication in each layer of the network. In traditional artificial neural network, the input to the network will be a feature vector and not directly raw data. In CNN the image features are automatically calculated by convolution operation with learnable parameters and hence there is no need to separately calculate data features. The input to CNN is directly raw data instead of feature vector. This also allowed to extract global features from image which was not the case earlier. This is one of the motivations why CNN were invented.

1(a) CNN Architecture and Training

1) CNN consists of fully connected layer, convolutional layer, max pooling layer, activation function and sometimes softmax function.

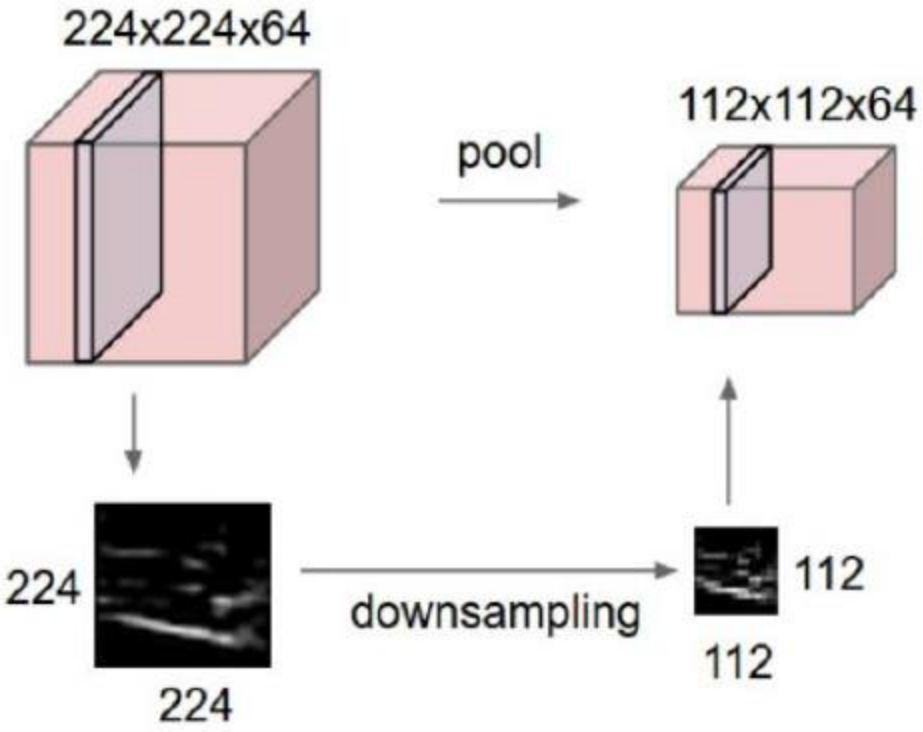
a) Fully connected layer

Fully connected layer connects all the preceding neurons to the succeeding neurons. Here every neuron of one layer is connected to every other neuron in the next layer.

b) Convolutional layer

Convolutional layer is different from fully connected layer in the sense that every neuron in one layer is not connected to every other neuron in the next layer. The neurons in subsequent layer will only be connected to neurons in small region of the previous layer called the receptive field. Convolutional layer has neurons arranged in 3 dimensions: width, height and depth. The parameters of convolutional net are set of learnable filters. Each filter is spatially small but extends through the depth of the input volume. In the forward pass each filter is convolved across width and height of the input volume. In this way it generates 2D activation map which is the response of filter convolved at every pixel position. The size of filters is a hyperparameter.

c) Pooling layer



Pooling layer performs downsampling of output of previous layer along the spatial dimensions. This is done to reduce the number of parameters and computation in the network. It is useful for extracting dominant features which are rotation and position invariant. There are 2 types of pooling techniques generally used. Max pooling returns the maximum value from the patch covered by the kernel. Average pooling gives the average of pixel values in the patch covered by kernel. Max pooling performs denoising along with dimensionality reduction, while average pooling only performs dimensionality reduction. Therefore max pooling is preferred over average pooling.

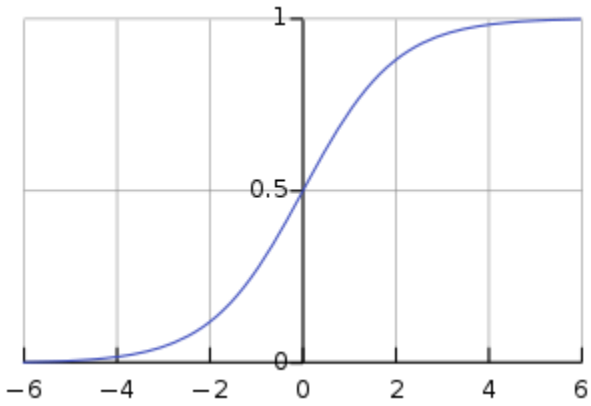
d) Activation functions:

Activation function is used for getting nonlinear outputs from linear combination of inputs to layer. The commonly used activation functions are Relu, sigmoid, softmax, tanh, leaky Relu etc. The output of each layer neuron in CNN is passed through an activation function defined for that particular layer.

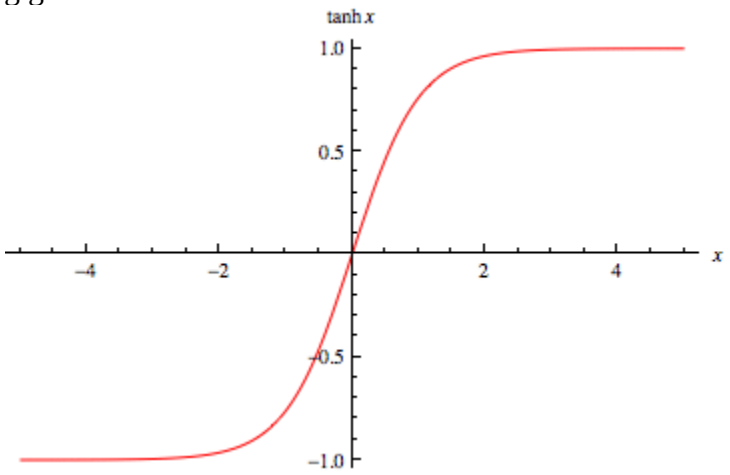
1) Sigmoid function:

   Any input value to this function is converted in the range 0-1. This activation function can cause the problem of vanishing gradients because higher input values reach saturation.

$$f(x) = \frac{1}{1 + e^{-x}}$$

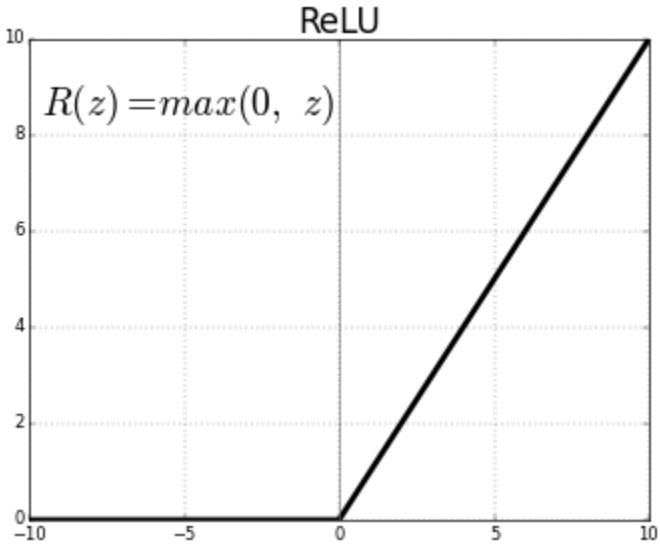


2) Tanh function:
This is similar function like sigmoid but a scaled version of it. The output values of this function are in the range of -1 to 1. This function also faces the problem of vanishing gradient.

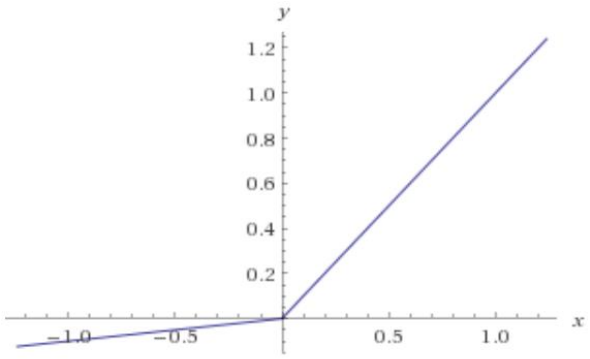


$$tanh(x) = 2\,sigmoid(2x) - 1$$

3) ReLu function:
This is the most commonly used activation function in deep neural networks. It is a good approximator (any function can be approximated by combinations of ReLu. ReLu has the advantage of making activations sparse and efficient. That means only fewer neurons in the network will fire for any input, this is because ReLu will cause all negative inputs to be 0. But this can be disadvantage also. For negative inputs the gradient will be 0 for few neurons which means the neurons will not respond to variations in error. This is called dying ReLu problem. This problem causes several neurons to die in network causing them to be passive.



4) Leaky ReLu:
To counter the dying ReLu problem, this function can be used. Now negative inputs have some small negative outputs. So negative inputs will have some gradient value instead of 0.



e) Softmax function:

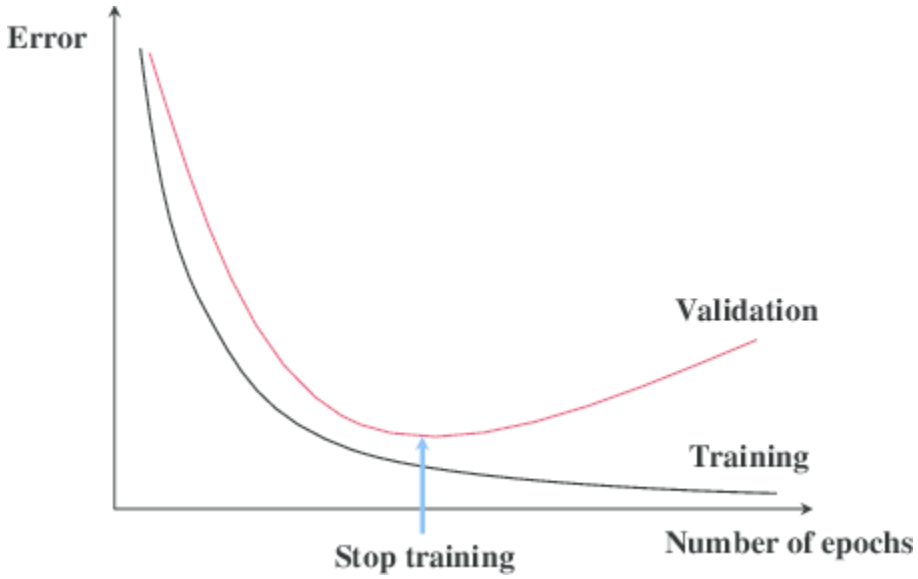
It is an activation function specially used in the last layer of CNN in classification tasks. Each neuron in the last layer of CNN is passed separately through this activation function. The output of this function is in the range of [0,1]. It means we get the probabilities for each class associated with a particular input. The formula for softmax function is given as,

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}$$

Here, yi is the input and S(yi) gives the probability value for that input.

2) Overfitting refers to a model that models the training data very well. Poor performance on test set can be a sign of model overfitting the training data.
Technique to avoid overfitting

a) Early Stopping



Early stopping is a regularization method when training a network with iterative method. In early stopping the training data is split into training and validation sets and a graph of error vs iterations is plot for the model on both training and validation sets. By observing the plot, the iteration after which the performance on training improves but validation goes down is noted. The model is then trained only until the optimal number of iterations. This process ensures generalization of the model to unseen data.

3) CNN is used in computer vision applications because through its architecture it can solve complex problems in field of object recognition, object detection, image classification etc..

Traditional pattern recognition systems were divided into 2 main parts. The first part was feature extraction module and last one was classifier. The first module transforms the input patterns into low dimensional vectors and are robust to translation, rotation and other distortions. Feature extraction requires prior domain knowledge and is problem specific. That means for every new problem new features need to be designed. One of the main problems with this approach was that the performance of classifier largely depended on the designer who designs the features.

CNN offers automatic feature extraction and classification all together in one architecture. CNN only needs large amount of labeled data to learn features from images. Using optimization techniques it learns the parameters of the architecture that help in extracting features and classifying images from those features.

Advantages of CNN over traditional methods:

a) The performance accuracy obtained by CNN is far more than traditional methods.
b) Sparse connectivity of weights in CNN means we need to store fewer parameters which reduces memory requirements for model and improves statistical efficiency.
c) Parameter sharing in CNN causes to learn on a set of parameters (filter weights) rather than leaning a separate set of parameters at every location. This reduces storage requirements of model parameters.
d) CNN is naturally invariant to translation because convolution operation.

4) Machine learns using loss functions. It evaluates how well your learning algorithms model the data. If the predictions deviate very much from the desired labels the loss function value increases and vice versa. Loss function are used in optimizations algorithms. Optimizations algorithms try to reduce the loss function value so that the algorithm models the data closely. Optimization algorithms minimize or maximize some objective function.
Backpropagation is a way of computing gradients of functions through recursive application of chain rule. In NN Backpropagation algorithm allows information from the loss function to flow backwards through the network to compute gradients of objective function with respect to weights and biases of the neural network. In backpropagation error is propagated backwards through respective neurons and every neurons contribution to the final output is found by calculating gradient of loss function w.r.t. weights in respective layers. One of the main criteria for backpropagation to work is that the activation functions in the neural network are differentiable. Backpropagation calculates the gradients in the network using chain rule of differentiability.

1(b) Train LeNet-5 on MNIST Dataset

I. Abstract and motivation

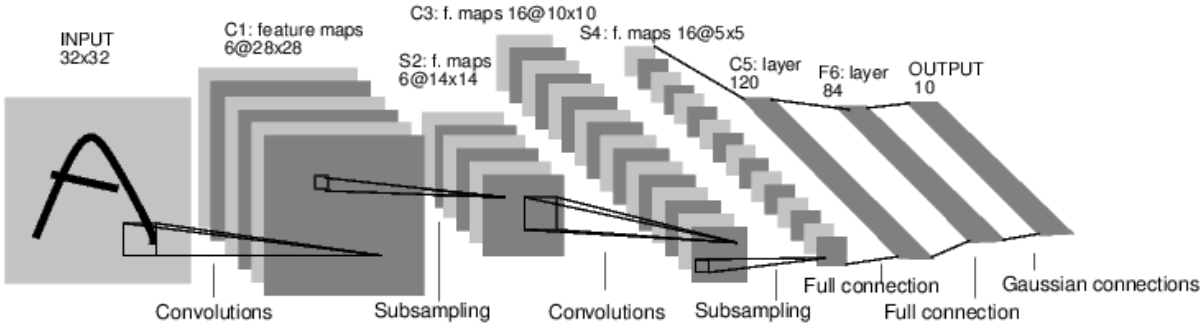


Figure1: LeNET5 architecture

LeNET5 is on of the first CNN that propelled the field of deep learning. This architecture is based on the insight that image features are spatially distributed across entire image. This architecture provided way of extracting these features using convolution operation with learnable parameters applied at multiple locations of image.

II. Approach:

The network architecture has 2 convolutional layers, 2 subsampling layers and 3 fully connected layers.

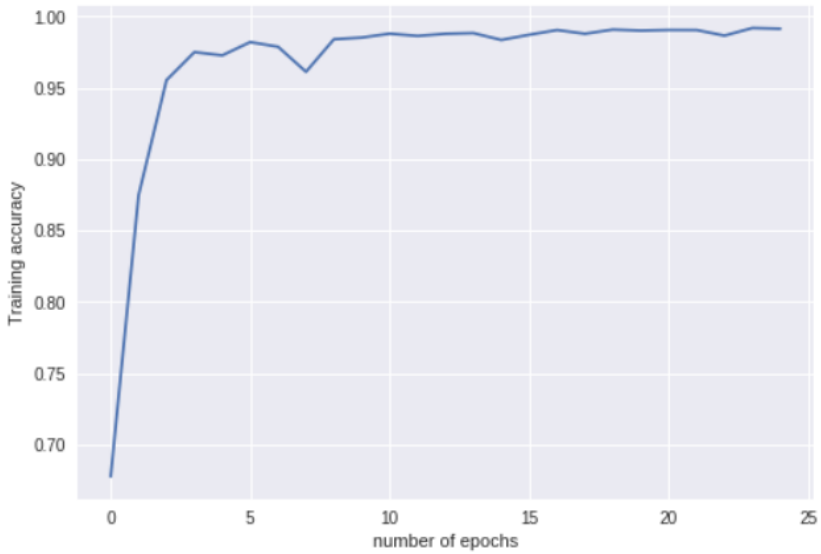
Input layer: This layer contains grayscale image of size 28x28.
Convolutional layer 1: This layer is made of 6 feature maps. Each of these are obtained by convolving 5x5 filters with the input image to produce feature maps of size 24x24. These feature maps are padded with 2 on each side to get 28x28 feature maps.
Subsampling layer 1: In order to reduce the dimensions of feature maps obtained using convolutional layer 1 max pooling layer with kernel size 2x2 is used. This subsamples the input image into half the size along spatial dimension. The output of max pooling is passed through ReLu activation function.
Convolutional layer 2: This layer has 16 feature maps generated by 5x5 size filters. This layer takes input of 6*14*14 and produces 16*10*10 output.
Subsampling layer 2: This layer takes input of size 10*10 and downsamples it to 5*5 size spatially. The output of max pooling is passed through ReLu activation function.
Fully connected layer 1: Input to this layer is 16*5*5 flattened vector and each neuron is connected to 120 neurons. Relu activation function is used for all 120 neurons.
Fully connected layer 2: Each of 120 neurons is connected to each of 84 neurons in the next layer. Relu activation function is used by all neurons.
Output layer: 84 neurons are each connected to each of 10 output neurons. The output neurons have softmax activation function which converts the input to output class propabilities.

Procedure:

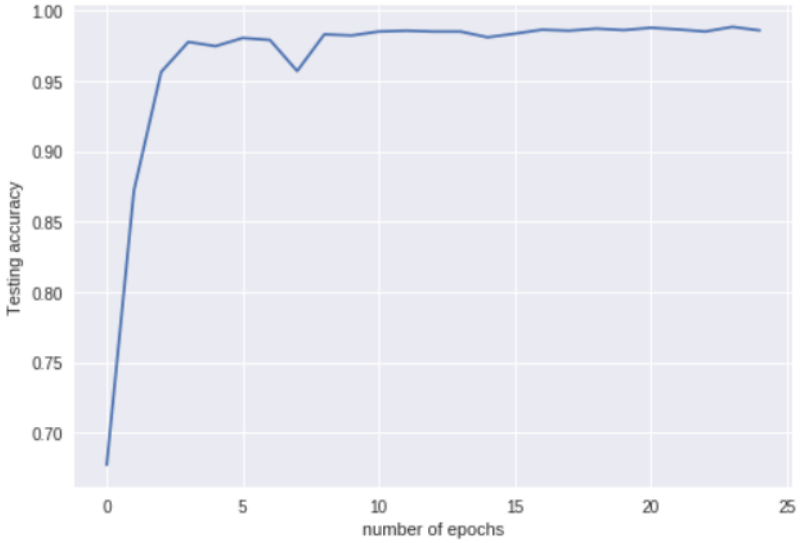
1) Load the MNIST dataset.
2) Define the neural network
3) Define the loss function and optimizer to be used for weight updates.
4) For each epoch: (25)
    I) For each batch (25) in all training batches:
        1) Train the network on batch.
    Predict on the entire training data.
    Calculate the accuracy on entire training data and append it to a list.
    II) Predict on the test data
    Calculate the accuracy on entire testing data and append it to a list.
5) Plot training accuracy vs number of epochs.
6) Plot testing accuracy vs number of epochs.
7) Append the accuracy on test and train data for 25^{th} epoch for each of the 5 hyperparameter setting in a list.
8) Calculate the mean and variance of 5 settings.
9) Plot the training vs number of epochs and testing vs number of epochs for the setting that gives highest accuracy on test set.

III. Experimental results

Momentum: 0.9 Learning rate: 0.01 Epochs: 25 Batch size: 25 Optimizer: SGD

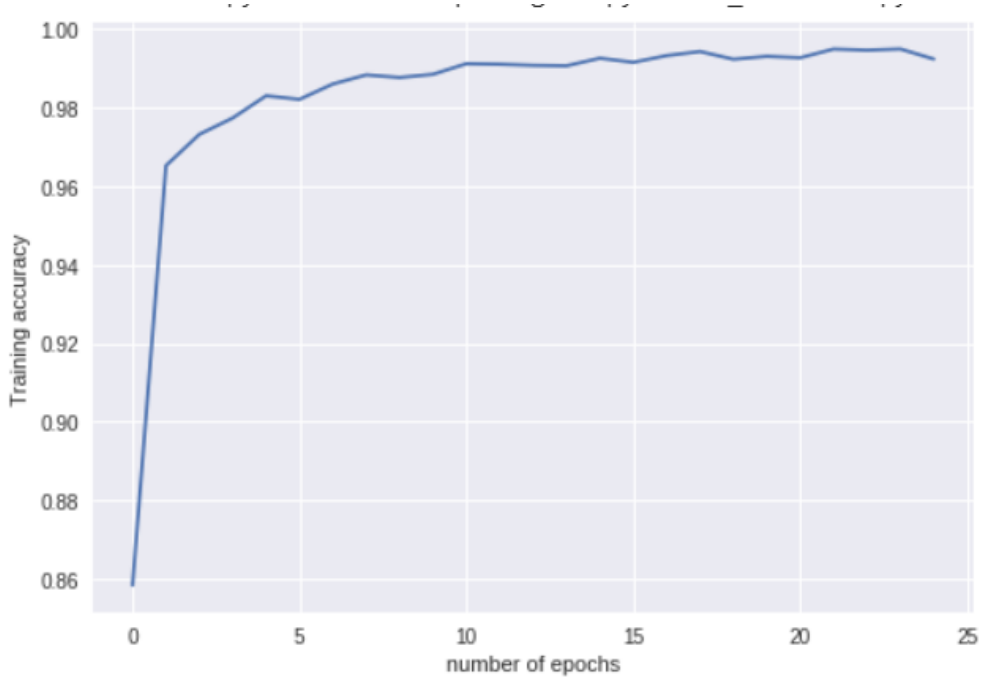


Train accuracy = 0.99276667

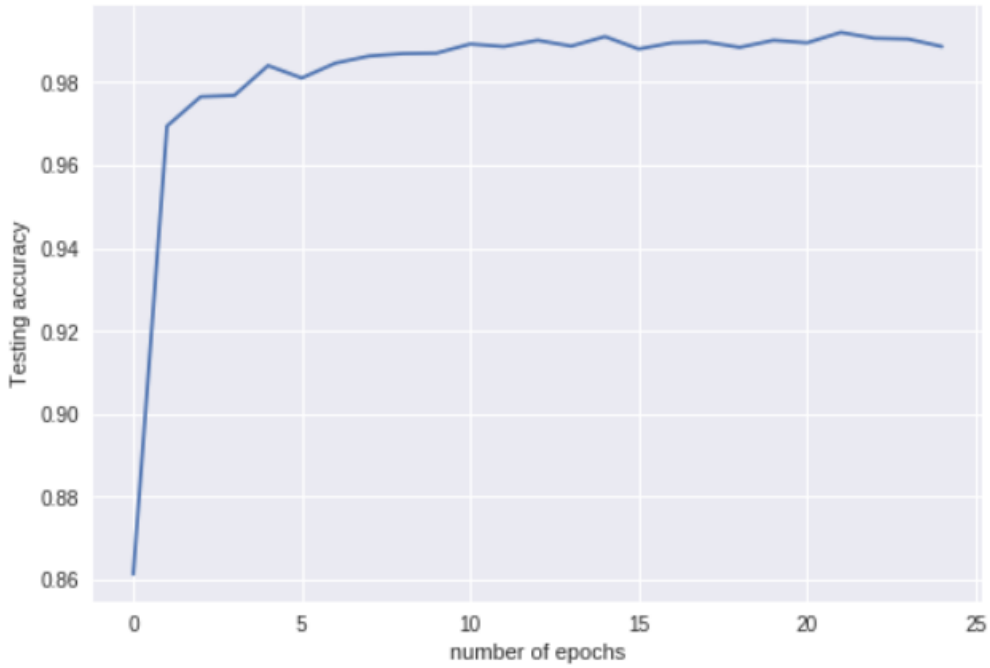


Test accuracy = 0.9861

Momentum: 0.9 Learning rate: 0.01 Epochs: 25 Batch size: 25 Optimizer: SGD
Number of filters for first convolutional layer: 16

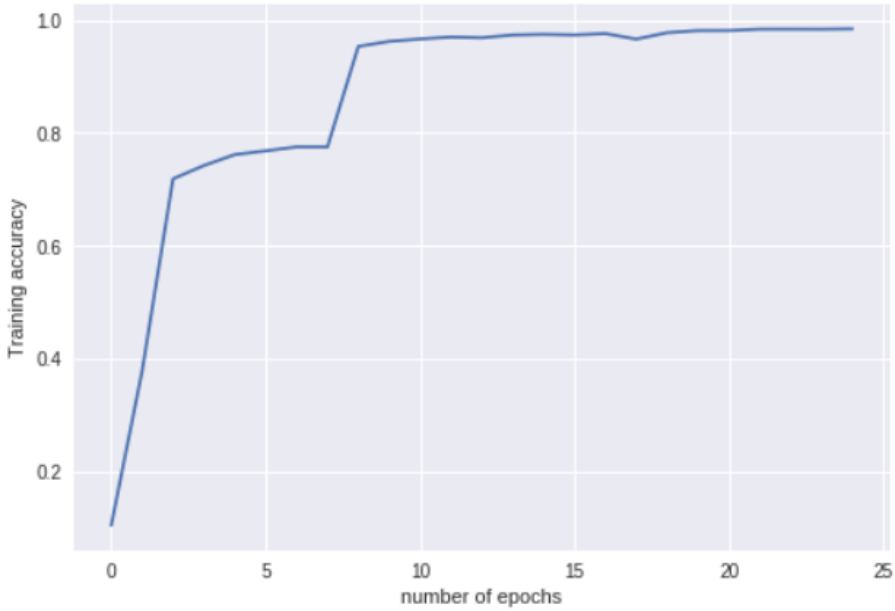


Train accuracy = 0.99453333

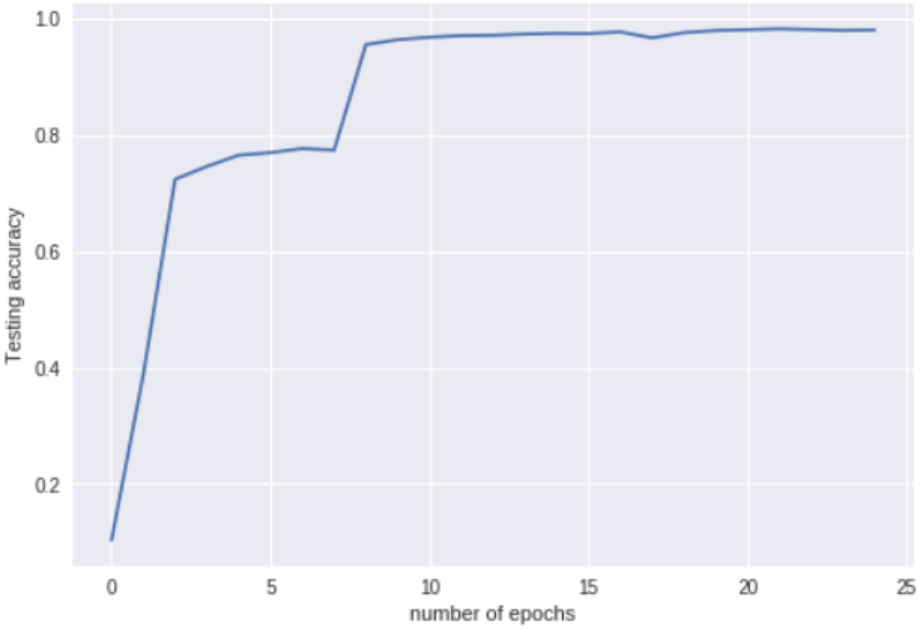


Test accuracy = 0.9886

Momentum: 0.9 Learning rate: 0.01 Epochs: 25 Batch size: 160 Optimizer: SGD

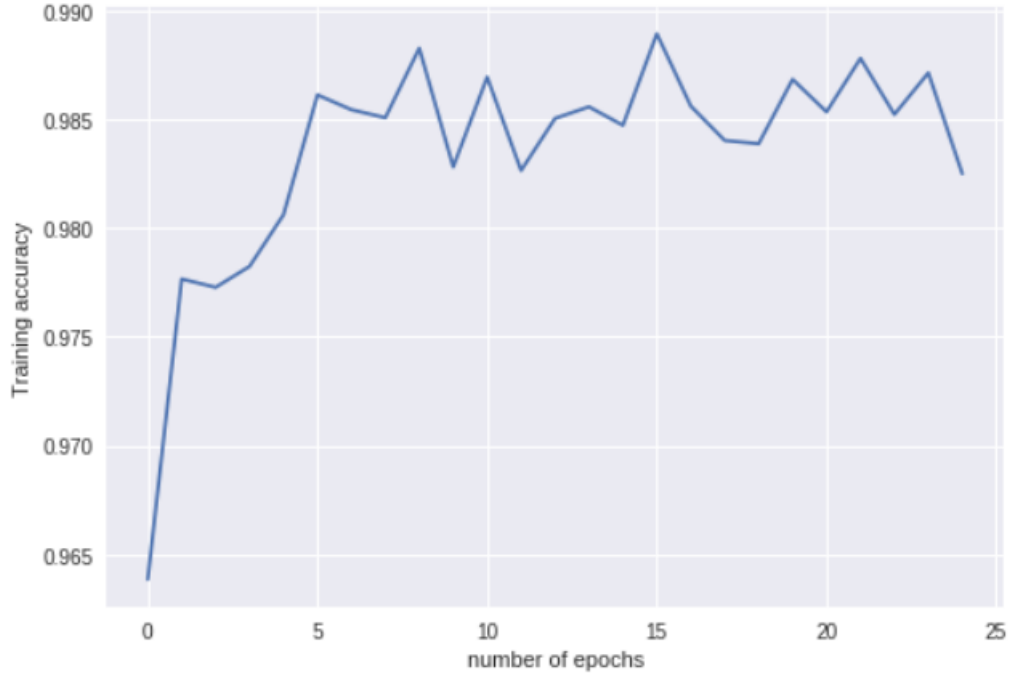


Train accuracy = 0.98395

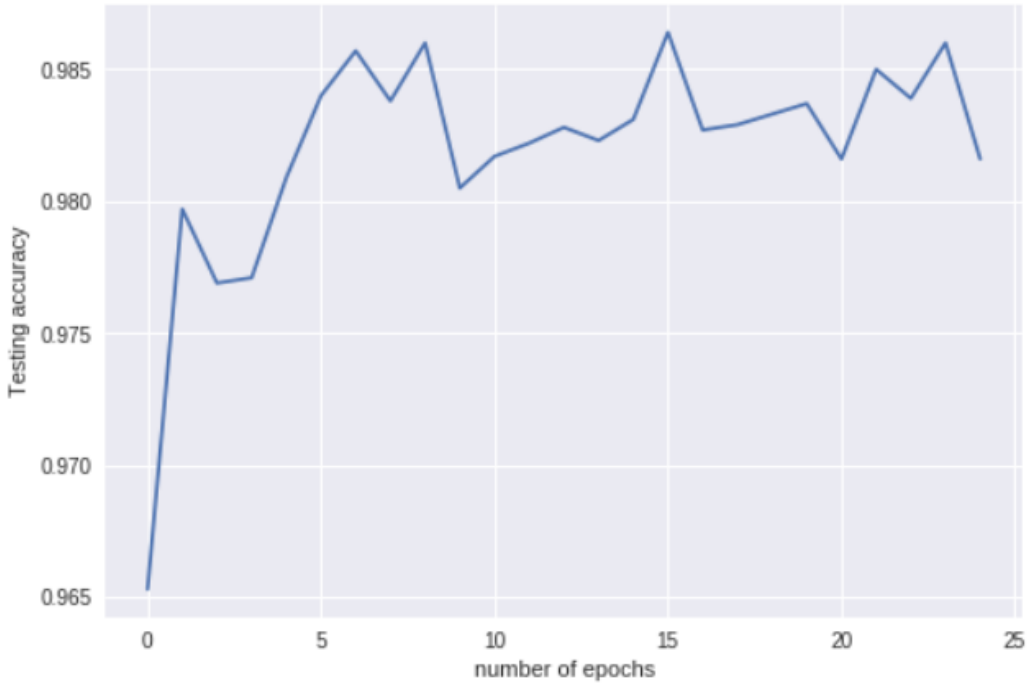


Test accuracy = 0.9805

Momentum: 0.9 Learning rate: 0.01 Epochs: 25 Batch size: 25
Optimizer used: Adam

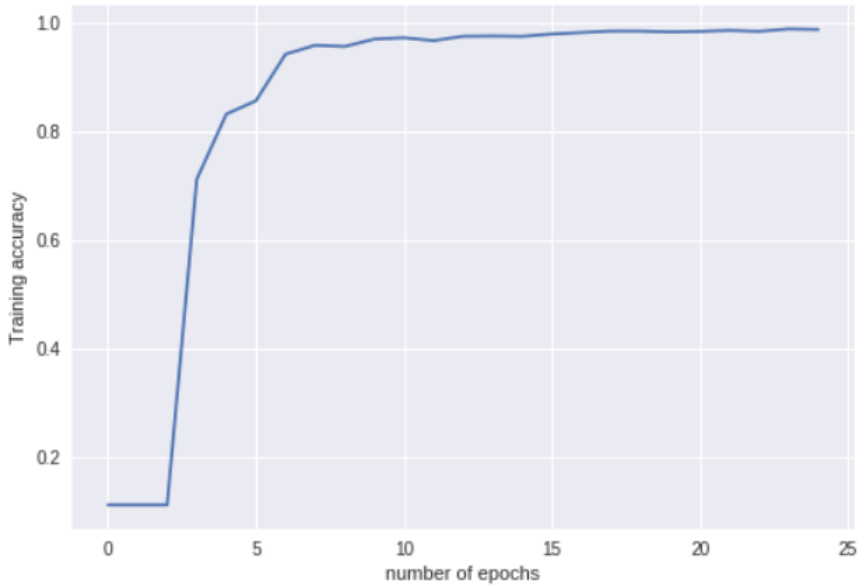


Train accuracy = 0.98296667

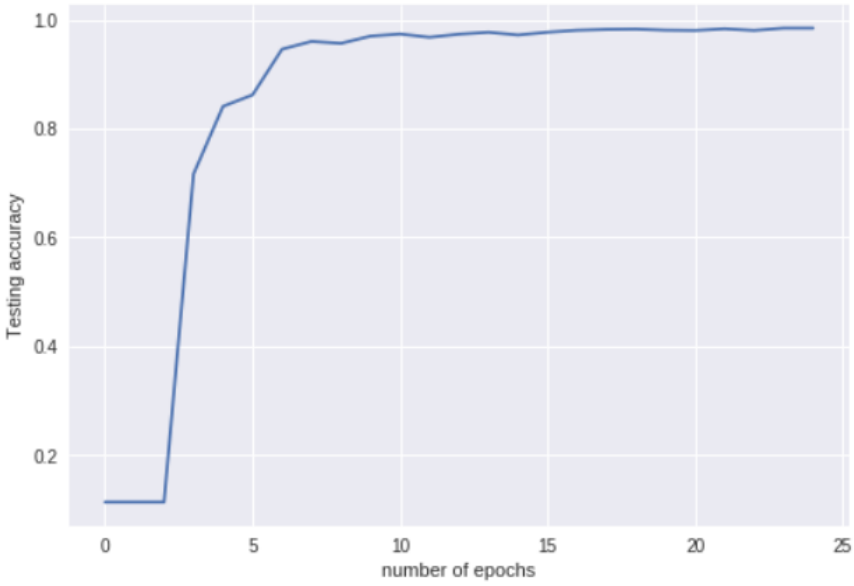


Test accuracy = 0.9816

Momentum: 0.9 Learning rate: 0.01 Epochs: 25 Batch size: 25 Optimizer: SGD
Filter size in both convolutional layers changed to 3x3



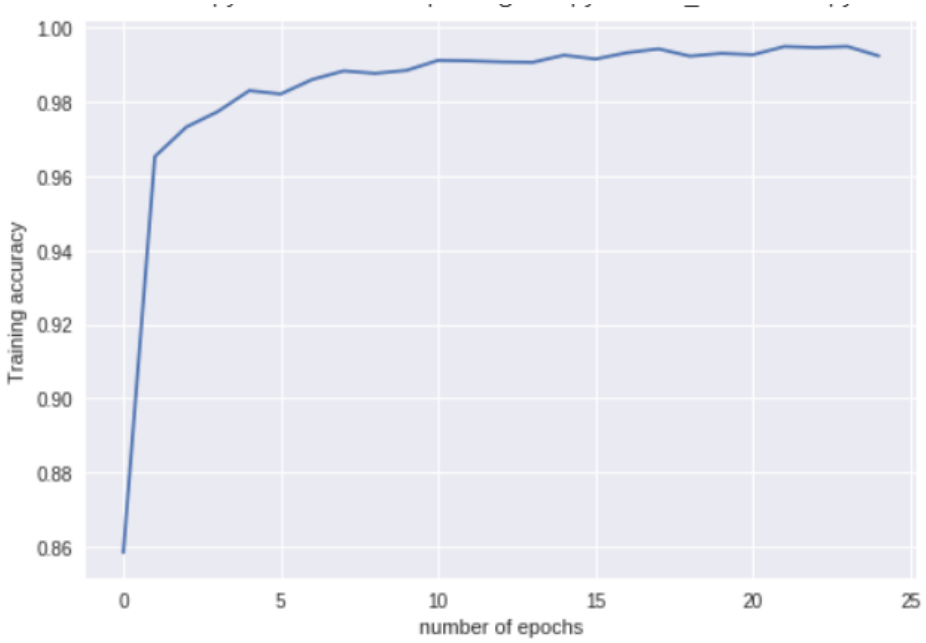
Train accuracy = 0.98741667



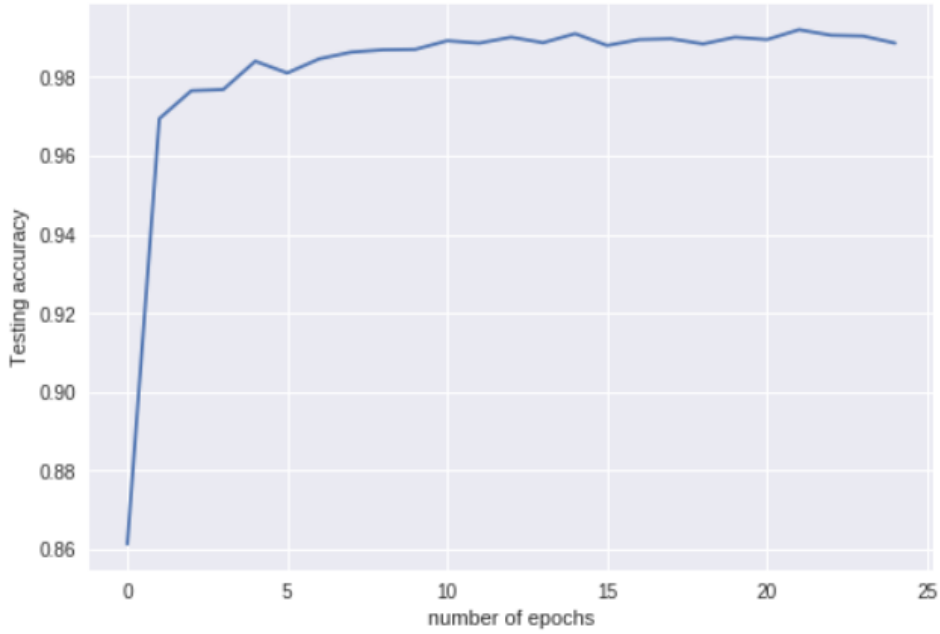
Test accuracy = 0.9849

Best parameter setting
Momentum: 0.9 Learning rate: 0.01 Epochs: 25 Batch size: 25 Optimizer: SGD
Number of filters for first convolutional layer: 16

Performance curves



Train accuracy = 0.99453333



Best accuracy on test data: 0.9886

Train mean accuracy for 5 settings: 0.9876966666666667
Test mean accuracy for 5 settings: 0.9843399999999999

Train variance for 5 settings: 1.4117599999999744e-05
Test variance for 5 settings: 8.762399999999952e-06

IV.Discussion

- Mean accuracy on test data gives us the approximate accuracy under different hyperparameter settings. Variance value tells us about the range in which the test accuracy is changing in different hyperparameter settings. From results we can see that the variance of test set on 5 settings is less compared to train set. This shows that the results are stable on unseen data.
- **Preprocessing:**
  The input 28x28 images have global mean subtracted from them and are zero centered. I have normalized the images in the range of 0-1.
- Learning rate decided how fast the network learns the weights. If learning rate is too small then the network will take long to converge and if it is high network will converge soon and the accuracy can be oscillating about some point. Momentum accumulates the gradients of past steps to guide the direction of the descent in SGD. Coefficient of momentum in weight update equation of SGD decides the percent of gradient retained in each iteration.
- Epochs denote the number of times the network is trained on the entire training data.
- In setting 1, I have used Batch size of 25. Batch size decides how many training examples are used in 1 iteration. It decides how computation time of SGD algorithm. If batch size is more it will take less computations because less iterations will be involved and vice versa. Typically, batch size has not much effect on accuracy.
- In setting 2 I have increased number of filters to 16 in 1^{st} convolutional layer. This increased the accuracy. This is because increase in filter numbers causes different features to be extracted from the same local region of the image. Ultimately more feature maps of the image are generated from the input image and more features are helpful in classification.
- Setting 3 has batch size increased to 160 which does not affect accuracy compared to batch size 25. This is expected result as discussed for setting 1. The sharp rise can be due to the fact that higher batch size training converges fast towards local minima of the cost function because the gradient descent steps involved will be less comparatively.
- For setting 4 I have used Adam optimizer in place of Stochastic Gradient descent. From train and test plots we observe that the accuracy is oscillating after 15 epochs about 95%. Adam optimizer is an extension of SGD with applications in computer vision and natural language processing. These oscillations are due to having separate learning rates for each weight parameter in the network. Individual learning rate parameters are computed from estimates of first and second moments of the gradients.
- In setting 5 I have changed the receptive field from 5x5 to 3x3 for both convolutional layers in the network. This did not affect the accuracy of the network much. The effect of the size of the filter will depend on the type of visual feature each filter is extracting. The filter in 1^{st} layer is responsible for extracting elementary visual features and 2^{nd} layer extracts higher order visual features.

1(C) Apply trained network to negative images

I. Abstract:

In this problem we test LeNET5 trained on original MNIST dataset on negative MNIST dataset to demonstrate that the model is not robust to illumination changes. The value of negative image at each pixel location is (255 – original pixel value). I have also designed a new network that can recognize both original and negative images from the MNIST test set.

II. Approach:1

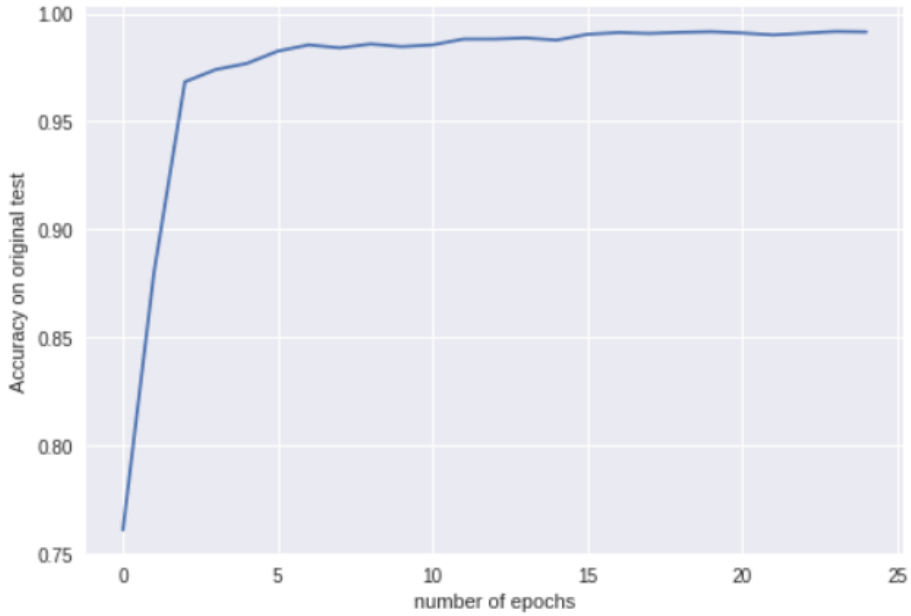
1) Load the MNIST dataset.
2) Define the best neural network from 1(b).
3) Define the loss function and optimizer to be used for weight updates.
4) For each epoch: (25)
   I) For each batch (25) in all training batches:
      2) Train the network on batch.
5) Calculate accuracy on the original test data.
6) Calculate the accuracy on negative images test data.
7) Plot original test data vs number of epochs.
8) Plot negative test data vs number of epochs.

Approach:2

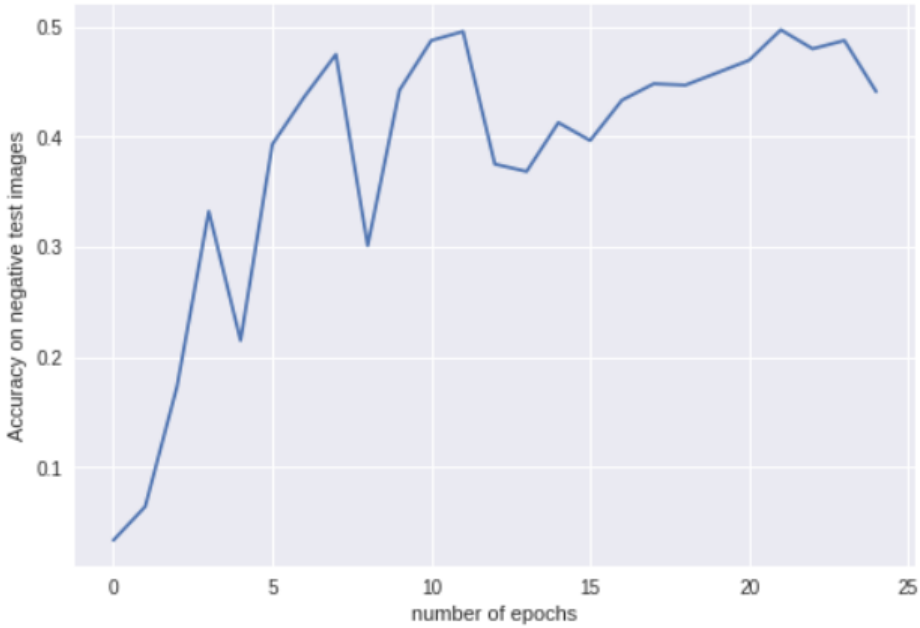
1) Load the MNIST dataset and negative of MNIST dataset.
2) Define new neural network.
3) Define the loss function and optimizer to be used for weight updates.
4) For each epoch: (25)
   I) For each batch (25) in all training batches:
      3) Train the network on batch.
5) Predict on the original test and negative test images and calculate accuracy in each case.
6) Plot original testing accuracy vs number of epochs.
7) Plot negative images testing accuracy vs number of epochs.

III. Experimental results

C(1)

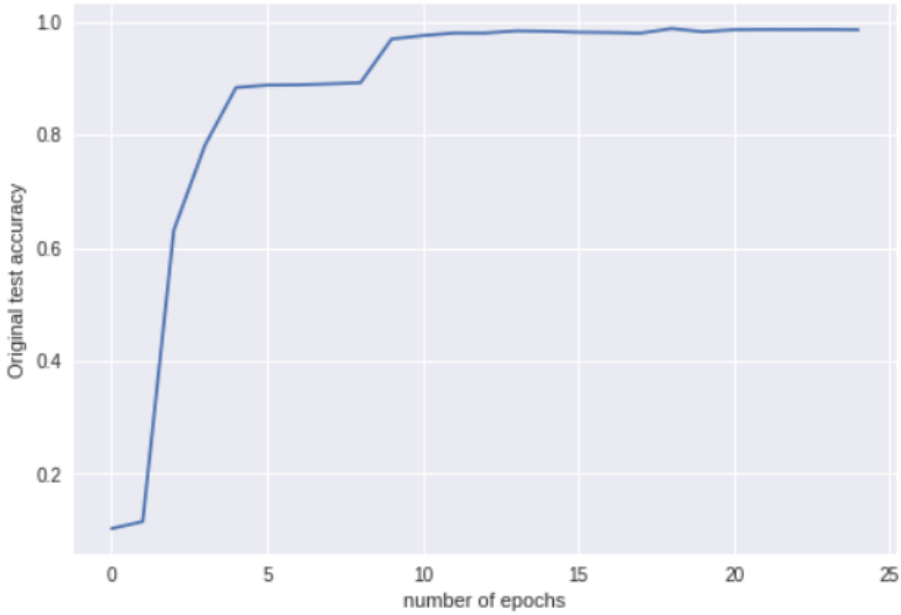


Test accuracy on original images: 0.9914

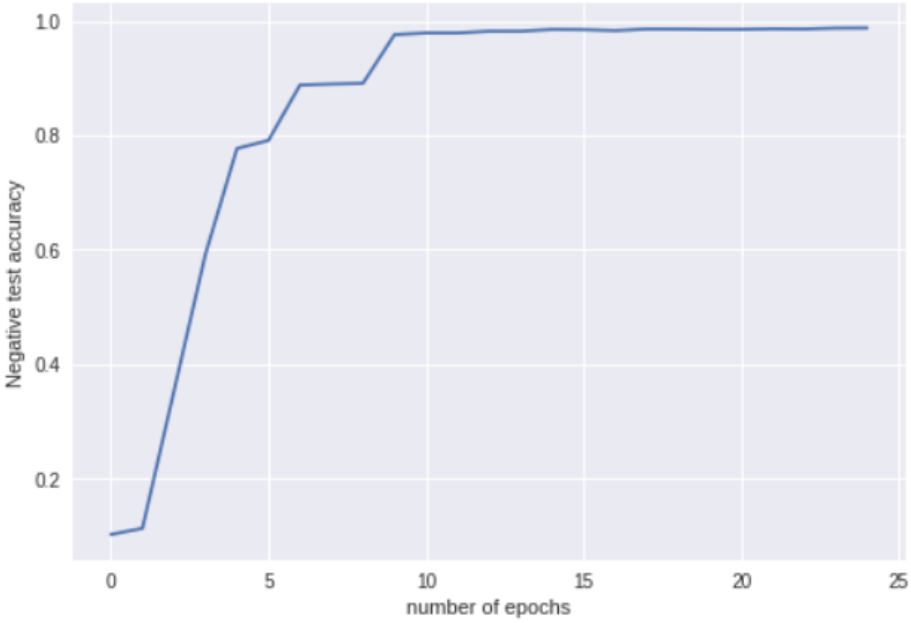


Test accuracy on negative images = 0.441

C(2)



Test accuracy on original test images = 0.9851



Test accuracy on negative test images = 0.9854

IV. Discussion

- CNN is naturally invariant to translation because convolution operation is equivariant to translation but is not equivariant to some other transformations. This can be verified from accuracy results of network on negative test images. This happens because the network has not generalized well and is not robust to illumination changes.

- The performance of the architecture is improved using data augmentation. In data augmentation we create more data from available data if additional labeled data is not available. Here I have augmented original images with negative of those images to create dataset. Training part of this data was used to train the network. This improved the accuracy of the network on negative images. Deep neural networks generalize well and their performance is increased on providing more training data. Therefore this data augmentation technique gave good results on negative images as well.

References:

1) http://cs231n.github.io/convolutional-networks/
2) https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23
3) dataconomy.com/2017/04/history-neural-networks/
4) https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
5) http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
6) https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
7) https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html#sphx-glr-beginner-blitz-cifar10-tutorial-py