Analysis of Convolutional Neural Networks and Multilayer Perceptron for MNIST Dataset

By
Rishab Goel,
Electrical and Computer Engineering Department,
University of Florida

***Abstract*— This document describes the comparative analysis of convolutional neural networks with traditional neural networks, multi-layer perceptron, for MNIST handwritten digits dataset. MNIST is a dataset of 70000 images with 60000 train images and 10000 test images. Convolutional Neural Network are neural networks which recently gained much popularity for solving problems in many image recognition areas. They are currently venturing into even to solve speech recognition problems. They have been in existence in theory for a while, but the machines reached the computation power to train them recently only. Moreover, their success in solving many recognition problems with high accuracy as well as on a large scale make them a hot research as well as industry area. In this document, we analyze various configuration parameters for convolutional neural network ranging from layer and kernel sizes, overfitting corrective measures, weights initializations and optimizers. We use the analysis of various choices for each configuration parameters to arrive to a convolutional network to finally achieve accuracy closer to human error for MNIST test dataset. We further compare their training behavior and their accuracy with the multilayer perceptron network from the results obtained for the MNIST test dataset.**

***Index Terms*—convolutional neural network, multi-Layer Perceptron, MNIST Data Set.**

I. Introduction

The handwritten digit recognition though may sound simple in today's scenario, but to achieve the accuracy closer to human error by machine learning methods is not a cakewalk. MNIST dataset is such dataset of 70000 handwritten digit images among which 60000 are train images while remaining 10000 are test images. The MNIST training and testing set images are taken from NIST train dataset and NIST test data. It is a widely used database of images to test and experiment with multiple machine earning methods.

The convolutional neural networks are not much different from the regular neural networks in terms of learnable weights and biases. Instead of working on a single dimension input structure like regular neural networks, they work of 3-D input structures (C * W *H). The convolutional neural network have a special property to work on inputs as images (even 3 channel RGB or YUV images. This makes them suitable for complex visual-recognition processing tasks.

The convolutional neural networks have basically four layers: convolutional layer, pooling layer, activation layer and fully connected layer followed by a loss function layer. Each layer is implemented 3D network of, connected with the number of neurons depending on the kernel sizes and input dimensions. Due to the number of parameters drastically increasing with input dimensions, they are relatively expensive in terms of computation and memory requirements for large resolution data sets

The convolutional neural network is not a black box instead of careful experimental analysis along with intuition is required to train it effectively. The convolutional neural networks are actually also derived from MLP network, and are modified to suit the image processing tasks. It makes sense of compare the convolutional neural networks with the multi-layer perceptron to show the effectiveness of the convolutional neural networks. In this paper further, we apply multiple hyperparameter modifications to train the convolutional network and discusses the effective of each modifications. These performance tuning parameters allow the convolutional neural network to achieve better generalization. MNIST dataset is used for experiments for better generalization of convolutional networks, and also for its comparison with multilayer perceptron. The dataset contains grayscale images of 28X28 size with a single handwritten digit being an image.

For our analysis we use Keras (Theano) documentation to construct a convolutional neural network and multilayer perceptron network. We tried training the network on my personal machine and also AWS server, but the time taken for the intensive Convolutional Network is huge. So for our training, we configured the keras library to work on my GPU (Nvidia GeForce 940M) which drastically reduces the execution time and also allows us ton increase the extent of the experiments. The paper further is organized into four sections further. In the section II, we discuss the convolutional neural network architecture used for our experiments and also describe how we reduce overfitting, the choice of optimizer and networks' weights initializations for the MNIST dataset. The section III explains, that the training of multilayer perceptron network for MNIST dataset and how we obtain the model with higher test accuracy. In section IV we compare and analyze the results and experiments that are generated after the implementation of the suggested experiments in section II and section III. In section V we compare the network configuration of convolutional neural network and multilayer perceptron with highest test accuracy. Finally the conclusion (VI) concludes the analysis with lessons learnt and scope for further optimization.

II. Convolutional Network Architecture

A. Description about the Network Layers

The Convolutional Neural Network [1] could be described as just a modification of the neural networks which made up of processing elements with learnable weights and biases. Each processing elements gets an input from the previous layer and generates an output using the dot product of the link weights and inputs which may or may not be passed through an activation functions. The weights and biases updating also takes place using the similar methods of backpropagation algorithm. They inherently assumed that the inputs to be images i.e. a 3-D collection of neurons in which each inputs corresponds to the pixel intensity. Each input pixels is not at all connected to all hidden layers instead a local receptive field of a particular size is created for computing the values for a hidden neurons. This local receptive field could be termed as actually a convolution kernel of a set window size which scrolls through all the inputs pixels or values. The kernel traces over the image plane could be overlapping or strided access. It is to be noted that each kernel over a single layer in convolutional network shares weights and biases. This makes sense as each layer will act as a feature extractor and will record a similar type of feature all over the input image. This helps the convolutional neural network to preserve the translational invariance over the images.

Note that for results and analysis we indicate first convolutional + ReLU layer by Conv1, second convolutional + ReLU layer by Conv2, first fully connected layer by FCC1and second fully connected layer by FCC2 and finally, the max pooling layer are indicated by Pool1 and Pool2 for results, observations and experiments. The network architecture is modelled in Fig 1 below.

The MNIST dataset could be termed as kind of hello world code for deep learning architectures. The input image is a 1 channel 28 x28 input matrix each having a handwritten digit.The literature survey method from references [1, 2, 3, 4, 5] suggests 4 convolutional layer with initial two being convolutional kernel layer while the next two being fully connected layer. When dealing with images the convolutional layer works on a local spatial field instead of connecting all neurons to everyone, to capture the local spatial patter. The fully connected layer is actually convolutional layer which has all processing elements connected to all inputs. It is beneficial at a later stage of the training as each convolutional layer captures a feature in the input and it seems slightly more intuitive to capture the whole pattern over the input space before final prediction. It is to be noted that it both convolutional layer we have 32 kernels each and whose window size is experimented with 3 and 5 values. The 5 windows size is experimented as it is expected to capture a larger spatial field, because MNIST is a dataset of handwritten digits so there is not much information in a lower spatial fields. The fully connected layer size of the last layer is fixed at 10, while we experiment with the sizes of 128 and 256 for the first fully connected layer. The convolutional network also have another layer named pooling layer which is suggested to be followed after each convolutional layer output to reduce overfitting. The pooling layer can be max or average pooling with a window size taken to be 2 with a stride of 2. We experiment with the insertion of the Max pooling layer at the first stage of the convolutional network. We choose a batch size of 128 on the basis of the literature survey.

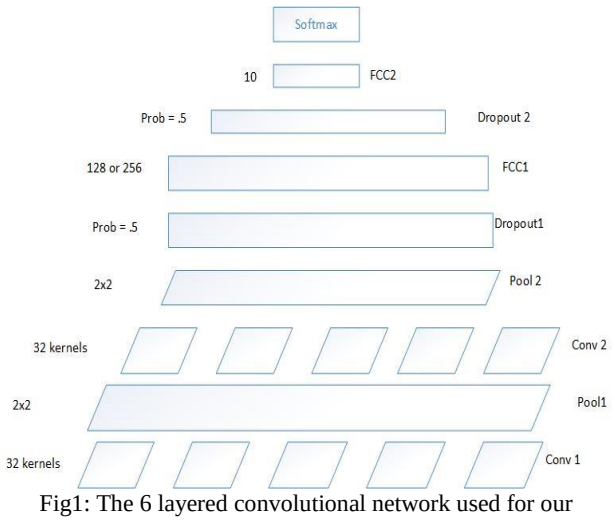


Fig1: The 6 layered convolutional network used for our experiments. The parallelogram suggests 2 input space while rectangle suggests 1D input space. Dropout layer adds additional two layer to the 6 layered convolutional network.

There are many activation functions in literature for the hidden layer outputs like sigmoid, tanh and Rectified Linear Unit. Though from experiments on the deep convolutional networks it is suggested to use ReLU to be used as the preferred activation functions for the hidden layer.[1] The advantage of ReLU is backed by two observations :

- faster convergence for Stochastic Gradient descent compared to sigmoid and tanh activation functions, and
- Simple thresholding operation function instead of computation intensive exponential functions.

$F(x) = \max(0,x)$

The ReLU activation functions is applied to the output of each convolutional layer. Though it suffers the problem of dying down of the processing elements due to large gradient inflow. The output classifier is a binary SoftMax layer to ensure binary classification and computes loss by categorically cross-entropy function [4]:

$C=-1n\sum x[y\ln(a)+(1-y)\ln(1-a)]$

; where **a** is the output of the activation function.

The cross entropy function is a usable cost function because its cost is non-negative value for C (> 0) and when the value predicted is actually close to the predicted value, then the loss function generates a value close to zero. But the property that makes it attractive for deep learning networks for classification is that it avoids the problem of learning slowdown in the least square cost function. This property exists because the derivative of the activation function vanishes, while the gradient remains the linear function of the error at the output. This property ensures that larger the error the faster the learning rate. This property not only reduces the time of training the network, but also minimizes to a higher accuracy.

Moreover, since we have 10 classes of number to predict so to take advantage binary classification property of the cost function. We change the outputs at the last layer to categories like 8 would be depicted {0,0,0,0,0,0,0,0,1,0} and 2 would be depicted {0,0,1,0,0,0,0,0,0,0}.

B. Overfitting corrections

The Dropout method is a recently introduced technique for training of deep neural networks to reduce overfitting. The concept of the dropout layer is to keep partial neurons of a layer only active inputs for the next layer as well as for backpropagation. This partial value is probability value so a probability of 0.5 means at any time in the training only half of the processing elements of the layer would be active. This method could be attributed to actually sampling the convolutional layer outputs. This technique process effective in testing by marginalizing the noise analytically. This method is proved very effective on various datasets. We also experiment with its effectiveness for MNIST data set by introducing at various stages of the network. The dropout layer is experimented at output of either and both the second convolutional layer and first fully connected layer or without it. It is suggested that for machine learning averaging out the results for different training architectures with large amount of datasets would give a more generalized picture of the learning model. We do these experiment on a 3x3 kernel convolutional network with a six layer architecture in figure 1.

Another important technique to reduce overfitting which is carried on from the training of multilayer perceptron network is early stopping. We separate the training set into two parts training set and validation data set with 50000 samples for training and 10000 samples for validation data set. The trained model on training set is validated after each epoch to compute the loss and accuracy on the validation dataset. When it is found that validation loss starts to increase the training is stopped. It is to be noted that putting an abrupt stop to the training may result to a lower test accuracy as it may happen after few iterations the validation loss starts to decrease again. Keeping to the above possibility in mind we introduce a patience factor is early stopping which allows the validation loss a headroom to start decreasing again.

We also enable the shuffling of the training dataset at each epoch to induce better generalization.

C. Optimizers

The learning method or optimizers decide how the weights and biases connecting the processing elements gets updated. We experiment with mainly two optimizers: Stochastic Gradient descent (SGD) and Root Mean Square (RMS) proportion. The Stochastic gradient descent method is the widely known learning method for neural networks [6]:

$wij(n+1) = wij(n) + \eta * \epsilon * \partial E/\partial w(t)$.

;where η is the learning rate.

This update method solely depends on weight trend in the current batch output, and there is no impact of the past portion of updates. [6] Therefore, it tends to have unstable update steps each iteration and leading to a slower convergence with an additional tendency to get stuck in the local minima. We can reduce these problems by introducing a momentum term which keeps the history of the optimization maintaining stable update steps and conforming to the optimization history[6]:

$wij(n+1) = wij(n) + \eta * \epsilon * \partial E/\partial w(t) + m * wij(n-1)$.

The momentum soothes the wild SGD learning curve but it may lead up to a sudden sign change 0.001 to -0.009 in the gradient value. This kind of behavior leads to loosing the gradient information. [10,6] To augment this behavior G. Hinton in the Coursera course suggests RMS proportion learning method. It keeps the running average of its recent gradient magnitudes and loosely normalizes the gradient values by dividing the next gradient by this average.[6]

$MeanSquare(w,t) = 0.9 * MeanSquare(w,t-1) + 0.1 * (\partial E/\partial w(t))^2$

$\triangle w(t) = \epsilon \partial E/\partial w(t)/(MeanSquare(w,t)+\mu)^{1/2}$

where μ (mu) is the smoothing value

For our training we combine the momentum with mu with RMS proportion [6].

For our training we experiment with different learning rate (0.1, 0.2 and 0.3) for SGD and SGD with momentum as well as for RMS prop. We maintain adaptive learning rate with a decay rate of 1*10^ (-6) as suggested by keras documentation. We also experiment with momentum values (0.5, 0.7 and 0.9) for SGDM. The learning rate of RMS proportion is kept petty lower 0.001 with mu value being 0.9 or 0.7 with a decay rate of 1*10^ (-6) ..

D. Weights Initializations

We ran the tests with varied weights initializations suggested by Glorot[7] and He [8]. Glorot Weights Initializations is modified to glorot uniform into the following formula*:*

sqrt(6. / (fan_in + fan_out)) ;where fan in and fan out is the dimension of the input and output processing elements. This is based on compromise and an equivalent analysis of the backpropagation gradients. Glorot uniform is most popular weight initializations, though, the success in paper [8] with ReLu activation function suggests to experiment with the idea for MNIST dataset as well.He [8] suggests that weights initializations suited especially for ReLu activation layer with a variance of neurons being he uniform by the formula:

sqrt(6. / (fan_in)).

We experiment with four configuration for weights initialization. Glorot uniform for both convolutional and fully connected layer, either Glorot uniform and He for convolutional and fully connected layer and finally He for both convolutional layer and fully connected layer.

III. MLP NETWORK

A. Input Data Preprocessing

The MLP network is a hidden layer with 256 neurons with RELU activation functions and followed by Dropout layer of 0.5.The MLP network is actually equivalent to fully connected layer used in the Convolutional Network which doesn't preserve any spatial information specifically for images, so it is suggested to modify the input data set. The input preprocessing helps in increasing the accuracy of the network of ANNs for

many learning patterns, so maybe it could help to preserve the spatial correlation between pixels. Note that we convert the two dimensional images (28x28) to single dimensional arrays of 728.For input preprocessing we try subsampling and principal component analysis as two methods and compare it with performance of single hidden layer MLP network. We tried input subsampling of 2 and 4 and dimensionality reduction in PCA to 80% of the original dimensions.

B. MLP Network Layers

We experiment with the number of hidden layers in the MLP network with purpose to arrive at a configuration for MLP network with best accuracy. The MLP single hidden layer network was brought down from 1024 to 256 as higher PEs were not effecting the accuracy much in fact took longer time to train, but we operate on the complete input data not without any preprocessing as indicated in the previous section that preprocessing reduces the test accuracy a lot. The convolutional network is considered specialized for images, so we attempt to achieve the MLP layer configuration with highest MNIST test data accuracy. We plan to use the dropout layer and ReLU activation functions with SoftMax classifier are used for MLP as well.

IV. Results and Analysis

A. Overfitting corrections

Fig2 Indicates the Effect of Dropout layer on the learning curves. No dropout layer has no dropout layer, Intenal1 dropout means dropout layer of 0.5 probability added after Conv2, Internal2 dropout layer is dropout of 0.5 probability after FCC1 and Dropout means two dropout layer of probability of 0.5 at Conv2 and FCC1. The test accuracy of Dropout, Intenal1 dropout, Internal2 Dropout and No Dropout is 99.23%, 99.09%, 99.10% and 98,97%. It is clearly observed from the training and validation learning curve that no dropout layer having low training loss and comparatively much higher validation loss than others. The 2 dropout configurations suffers the least overfitting and is smoother as per the learning curves and this hypothesis is strengthened further as its test accuracy is also higher.

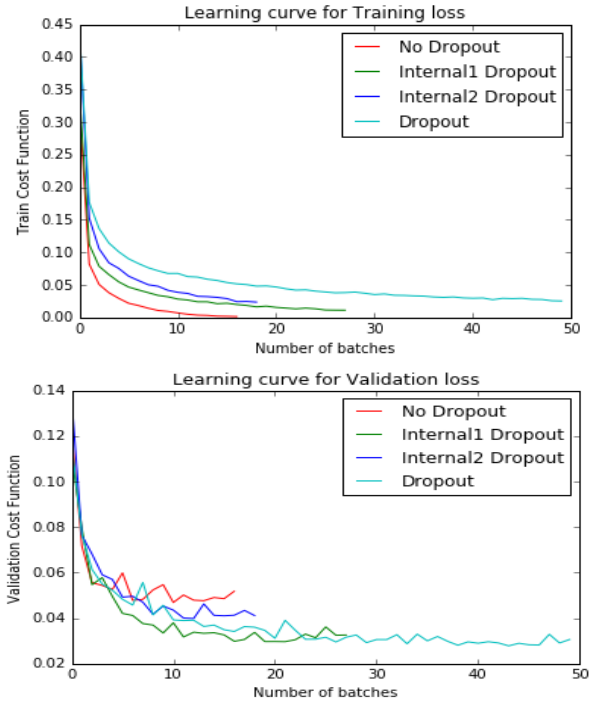


Fig2 Effect of Droput layer on Training and Validation curve of 6 layer ConvNets. The optimizer is SGDM with glorot initializations and convolutional kernel is 3x3 with fully connected layer of 128.

For early stopping we experiment 3x3 convolutional kernel network with SGD optimizer with learning rate of 0.1 and momentum 0.7 with a batch size of 100. The early stopping patience value was changed from 5, 10 and 0. The convolutional network with 5 patience value attained highest accuracy of 99.23% while the network with no validation stop has a test accuracy of 98.4%. The patience value also effects the test accuracy for 10 and 0 the test accuracy are 99.15% and 98.9 % respectively. It could be safe to assume that it is better to have a patience value for early stopping and choice the correct patience value is an experimental parameter. The introduction of Pool1 with dropout increased the test accuracy from 99.12% to 99.23%. On the basis of these experiments the Pool1, 2Dropout layer and early stopping with patience value 5 are made constant for further experiments.

B. Weights Initializations

We experimented with glorot uniform weight initializations for both the convolutional layer as well as fully connected layer as it is in general suggested to be a better method for weights initializations for neural networks. From the Table I it seems look He[8] weight initializations do perform slightly better than glorot uniform for the MNIST test dataset. Glorot-Glorot means glorot uniform weight initializations for both

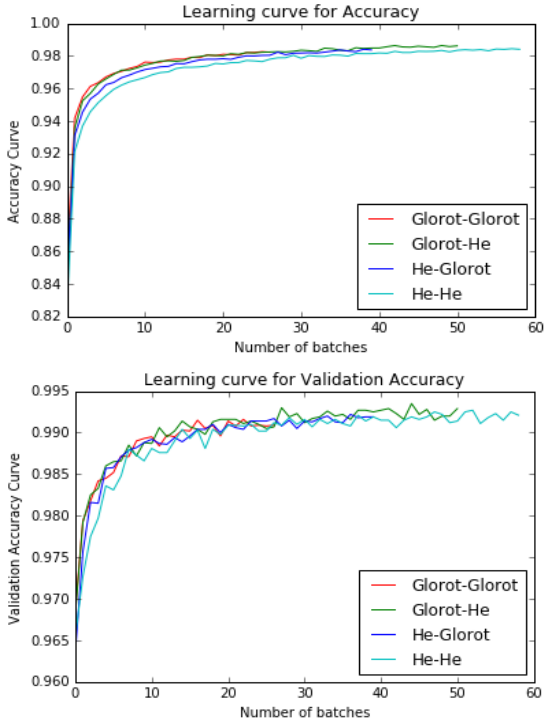


Fig3 : The accuracy curves for weights initializations for 6 layer ConvNets with SGDM=0.5 and convolutional kernel is 3x3 with fully connected layer of 128.

convolutional and FCC layers have Glorot weights initializations, while He–He means he uniform weight initializations for the both convolutional and FCC layers. The paper suggests that he uniform initializations tend to takes into account of ReLU activation function while its derivation which may enhance the performance of the network for the designated test set. From figure3 it is observed that the accuracy curves for all initializations are very close, for validation accuracy He – He curve slightly smoother which allows the training to continue slightly longer. This slight increase in epoch allows the model to achieve a better test accuracy. Note that this convolutional network of 3x3 kernels which as we see later is not actually our best configuration so we also try to experiment He weight initializations with our best configuration. This experiment will help us decide whether He weight initializations are in general better than glorot for deep nets for MNIST datasets or there is specific configuration only it could perform better.

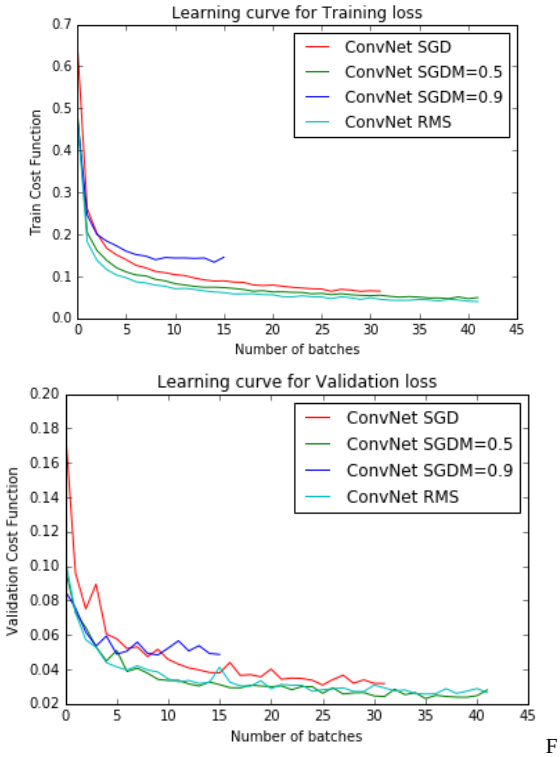
TABLE I. EFFECT OF WEIGHTS INITIALIZATION ON TEST ACCURACY OF THE MNIST DATASET

	Epochs	Test accuracy
Glorot -Glorot	26	99.21%
Glorot -He	50	99.29%
He- Glorot	39	99.27%
He- He	58	99.37%

C. Optimizers

The SGD momentum with 0.5 with learning rate 0.1 performed better than learning rate of 0.2 and 0.5 as indicated by Table II. So the learning rate value is fixed to 0.1 to compare SGDM=0.5 with other optimizers configuration. We experiment further with momentum parameter of SGM as 0,5 and 0.9 and observe a higher test accuracy for the given network configuration on MNIST. Choosing the optimizer as well as its best parameter configurations is most crucial choice in a neural network. The optimizer actually updates weights and biases of a network to learn the desired pattern.

Therefore, we experiment further with optimizers in Fig 4 and observe their learning curves for training and validation to observe their impact on network losses. We use the convolutional kernel with 3x3 size and 128 FCC1 for the results in Fig. RMS prop learning curves seems to be smoothest curve of all the optimizers, as suggested in section 2. Though surprisingly, SGD with momentum 0.9 have the least smooth curve even more than SGD without momentum which is actually expected to be smooth. SGDM 0.5 and RMS have a close learning curve for both validation and training set and don't seem to have any overfitting From Table II it is indicated that RMS have a lower test accuracy than SGDM 0.5 of about 0.15%, which considering the smooth curve of RMS seems unexpected. It seems like that smoothness of the training curve doesn't directly correlate to test accuracy, it may increase the accuracy for particular dataset and curve but not in generalization. Notice, SGD with momentum for different values have different learning curve smoothness and large variation in the test accuracy. This indicate that RMS prop in



F

Fig4: The learning curves for the optimizer for 6 layer convent with glorot initializations and convoltuional kernel 3x3 and fully connected layer of 128.

general gives smoother curve with better results. The configuration parameters RMS optimizer here for later stages of the paper remains as learning rate of 0.001 with mu value of 0.9 and decay rate 10^(-6) as changing this parameter don't improve the test accuracy for the network. Though for SGD with momentum there exists a particular configuration, which ensures smoother learning with higher test accuracy.

TABLE II. OPTIMIZERS' TEST ACCURACY FOR MNIST DATASET

	Epochs	Test accuracy
SGD l=0.1	31	99.11%
SGDM 0.5 l=0.1	41	99.23%
SGDM 0.5 l=0.2	25	99.13%
SGDM 0.5 l=0.5	21	99.03%
SGDM 0.9 l=0.1	13	98.49%
RMS mu=0.9 l=0.001	45	99.35%

D. Network layer sizes

Conv3FCC128 has convolutional layer with three kernels size and FCC1 with 128 PEs. Conv3FCC256 has convolutional layer with three kernels size and FCC1 with 256 PEs Conv5FCC128 has convolutional layer with five kernels size and FCC1 with 128 PEs. All the layer kernel sizes uses SGD optimizer with momentum=0.5. Conv3FC128 has convolutional layer with three kernel size and FCC1 with 128 PEs. From Table III Conv5FC256 has highest test accuracy closely followed by and next being Conv3FC256 though others two are not far behind. This gives a slight indication that

FCC256 is more effective in modulating the separation surface for MNIST test dataset. Moreover, the Conv5 kernel sizes are

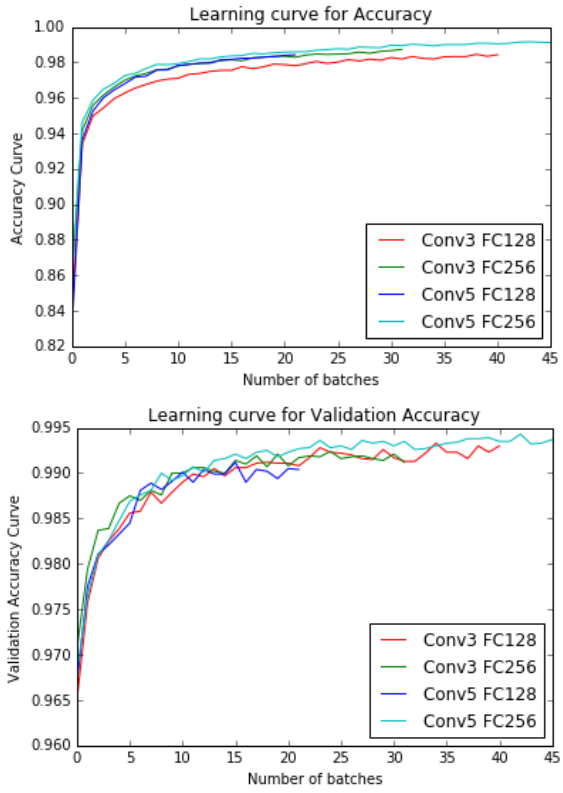


Fig 5: The accuracy curves on variation of the kernel sizes for 6 layer convent with glorot initializations and SGDM=05 l=0.1.

found to be more effective Conv3 kernel sizes which is expected due to the nature of the MNIST dataset. 5x5 kernel sizes tend to capture a better spatial detail than 3x3 for the input pattern than others. The test accuracy of approx. 0.2 % of Conv5FCC256 may seem minuscule in percentage, but numerically translates to 20 test images of 10000 test dataset. He uniform weights initializations doesn't perform better than glorot uniform for 5x5 convolutional kernel as seen for 3x3 convolutional kernel in the Table 1. It could be concluded about He-uniform initializations that it is a slightly different approach for weights initializations, which claims to be better with ReLU activation. On the basis of results, that he-uniform is not always better than glorat uniform initializations when using ReLU activation, but it depends on the combination of network parameters that may attain weights initiated with he uniform attain higher accuracy. From the accuracy curves in Fig5 it is clear that Conv5FCC256 is a better layer configuration for MNIST dataset compared to others as it not only performs much better than others, but also slightly smoother curves.. The patience value of early stopping limits the growth of other layer configurations as the validation loss grows for more 5 epochs for a lower accuracy value of the model. It could be concluded that smoothness of the accuracy or learning curves allow a better convergence in case of early stopping.

TABLE III. EFFECT OF CONVNETS LAYER SIZES ON TEST ACCURACY OF THE MNIST DATASET

	Epochs	Test accuracy
Conv3 FC128	42	99.23%
Conv3 FC256	31	99.30%
Conv5 FC128	21	99.25%
Conv5 FC256 Glorot	45	99.49%
Conv5 FC256 He	32	99.39%

E. Input Preprocessing in MLP

The Fig indicates the learning curves of the MLP network with different preprocessing techniques. The input data of 28x28 is flattened to 784 PEs and followed by preprocessing on the train, validation and test inputs. In MLP+PCA method, we do principal component analysis on the data to reduce the

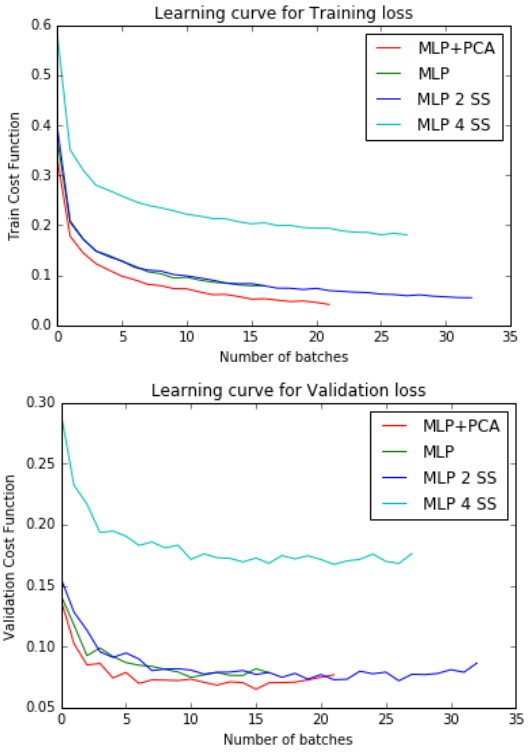


Fig6: The learning curves for the MLP single and two layer network with both SGDM and RMSprop optmizers.

dimensionality. We reduce the dimensionality for the learning curve to 80%. In MLP 2SS and MLP 4 SS method, we do subsampling of the input data for 2SS 1 value is preserved from two consecutive data value, and for 4SS 1 value is preserved from four consecutive data values. MLP 2SS and MLP 4SS reduces the input size by 2 and 4. While MLP configuration takes the input size as 784 itself. The learning curves in fig6 indicate that MLP and MLP 2SS perform very similar. MLP+PCA also doesn't suffers from any overfitting, but MLP 4SS seems to have stuck in a local minima. The test accuracy of MLP, MLP 2SS, MLP 4SS and MLP+PCA 98.14% 97.40%, 94.63% and 85.24% respectively. This test accuracy results indicate though MLP +PCA doesn't overfit, but it doesn't learn the desired features of the inputs. This could be attributed to the fact that the images have a spatial distribution and relation

while PCA has no such information as it works on a single dimensional input converts it two a different subspace. Surprisingly MLP 4SS performs much better than MLP PCA probably due to the preservation of spatial relation between pixels. MLP performs better than MLP 2SS, though is slightly slower, but for comparison with Convolutional Networks we use our best MLP results. We can also afford some extra time as it is meagre due to availability of the GPUs for training.

F. MLP Network Hidden layers

We experiment with multiple hidden layers in MLP network, the figure indicate the training and validation loss curves for the same. MLP SGDM indicate single hidden layer of 256 PEs

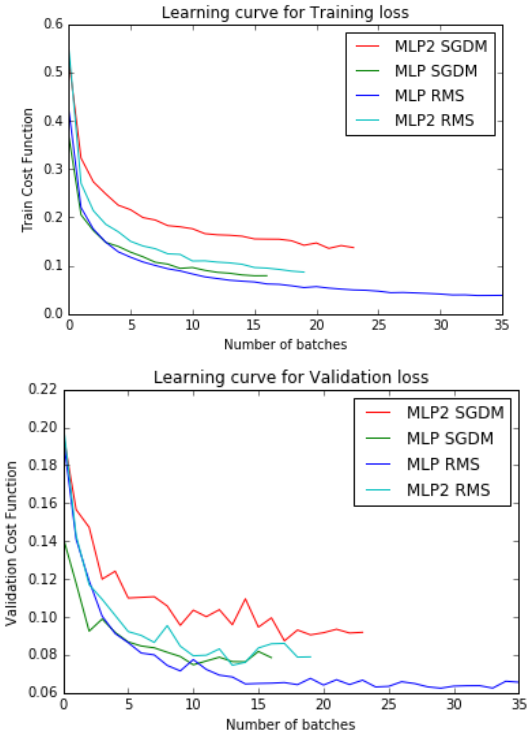


Fig 7: The learning curves for the MLP single and two layer network with both SGDM and RMSprop optmizers.

with SGD with momentum of 0.9 and MLP RMS indicate single hidden layer of 256 PEs with RMS optimizer of lr =0.001 mu=0.9.Two hidden layer network are indicated by MLP2 SGDM I and MLP2 RMS each optimizer having the same parameters as corresponding indicate single hidden layer. The two hidden layer network has 256 PEs in the first and 128 PEs in the second hidden layer each followed by a dropout layer.

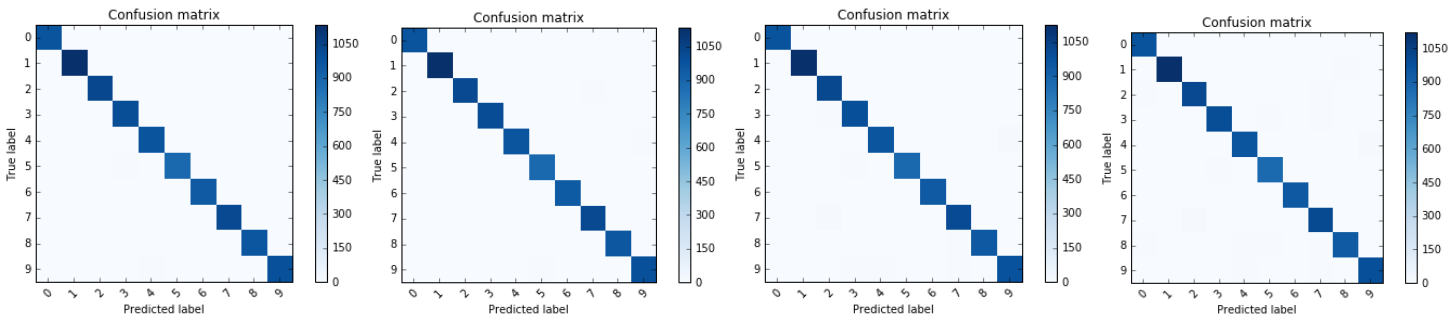
The learning curves in **Fig** for validation and training suggests that clearly 2 hidden layer network stop at much higher loss than single hidden layer network. This early stopping could be attributed to the highly zigzag learning curves for the 2 hidden layer, which induces early stopping to terminate the network early. This learning curve behavior gives us a fair indication that 2 hidden layer network won't perform better in test accuracy against a single hidden layer mode. The test accuracy for the network MLP SGDM, MLP2 SGDM, MLP RMS and MLP2 RMS are 98.14 %, 97.39 %, 98.15 % and 98.07 % respectively. This clearly proves our hypothesis that single hidden layer performs better than 2 hidden layer network and gives the best test accuracy for our MLP network experiments. This configuration is compared in the later section with the convolutional layer best configuration on the basis of test accuracy and learning as well as accuracy curves.

V. Comparative Analysis of ConvNets and MLP

We finally compare our best test accuracy configurations for convolutional networks and multilayer perceptron. In the Fig we compare both the network using both SGDM and RMS prop optimizers. The full network configuration of the convolutional network and multilayer perceptron network are mentioned in Table V and Table IV respectively. It is to be noted that convolutional network is deeper network but have lower parameters than MLP network. The higher number of parameters for MLP is due to the input itself each PE in the hidden layer is connected to the input element. It is to be noted that though ConvNet5 FCC256 have lower parameters than MLP network it takes much more time to train. The convolutional network designed achieves an accuracy with glorot uniform initializations and SGDM =0.5 optimizer attains a test data set accuracy of 99.49% and the same network with RMS optimizer achieves the test accuracy of 99.29%. It is to be noted that the multilayer perceptron with SGDM =0.9 and RMS prop achieves the test accuracy of 98.29% and 98.01% respectively. The best test accuracy of MLP network is around 1.2% (120 images) worse than best convolutional network test accuracy.

TABLE IV. MLP Network configuration that achieved accuracy of 98.29% on MNIST Test Dataset

Layers	Output shape	Param #
Dense	(None, 256)	161024
ReLU Activation	(None, 256)	0
Dropout	(None, 256)	0
Dense	(None, 10)	25632
SoftMax Activation	(None, 10)	0
*Total Network Parameters = 163594		



(a) Conv5 FCC 256 SGDM =0.5 (b) Conv5 FCC 256 RMSprop (c) MLP 256 PEs SGDM =0.5 (d) Conv5 FCC 256 RMSprop

Fig 8 (a), (b), (c) and (d) The confusion matrix for the 6 layered convolutional network and MLP networks

TABLE V. CONVOLUTIONAL NETWORK CONFIGURATION THAT ACHIEVED ACCURACY OF 99.49% ON MNIST TEST DATASET

Layers	Output shape	Param #
Convolution2D (32x5x5)	(None, 32, 24, 24)	852
ReLU Activation	(None, 32, 24, 24)	0
MaxPooling2D (2x2)	None, 32, 12, 12)	0
Convolution2D (32x5x5)	(None, 32, 8, 8)	25632
ReLU Activation	(None, 32, 8, 8)	0
MaxPooling2D (2x2)	(None, 32, 4, 4)	0
Dropout	(None, 32, 4, 4)	0
Flatten	(None, 512)	0
Dense	(None, 256)	131328
ReLU Activation	(None, 256)	0
Dropout	(None, 256)	0
Dense	(None, 10)	2570
SoftMax Activation	(None, 10)	0
*Total Network Parameters = 160362		

The comparative training plots of ConvNets and MLP of the learning curve.or training loss and validation loss as well as training and validation accuracy are presented in the Fig 9. seem to throw light on the reason behind this difference The loss and accuracy plot are closer to similar values for the training data for both the networks, but validation plots indicate a 2% lower accuracy and about 0.05 worse validation loss. The MLP network tends to have strong overfitting over the training data as compared to convolutional networks. It happens due to the fact that MLP layer trains on the individual gray level value of the input images instead capturing spatial pattern of the pixels. The convolutional layer in the convolutional neural networks actually captures the local spatial field which helps to learn the network the pattern of pixels over a local region. This property of the convolutional layer makes it effective for capturing patterns on images. It could be said each convolutional layer in the network captures a feature map from the previous input. As the depth of the network increases the complexity and details of this feature should increase.

The confusion matrix for these Conv5 FCC256 and MLP 256PEs for the SGDM and RMSprop optimizer are presented in the figure. The darker color indicates higher values of predictions in that cell. The confusion matrix due to color coding appears to be similar, for exact numbers please refer to plotconfusionmatrix.py for numerical values. It could be seen that number 1 have the highest correct prediction of over 1000 as it bears very less resemblance to other numbers. (6, 5, 8) and (4, 9) the two sets of numbers which suffer more mispredictions among them due to the similarity in their writing pattern.

VI. CONCLUSION

It could be concluded that experimental analysis of convolutional networks for MNIST dataset that they preserve the local spatial pattern between pixels which helps in

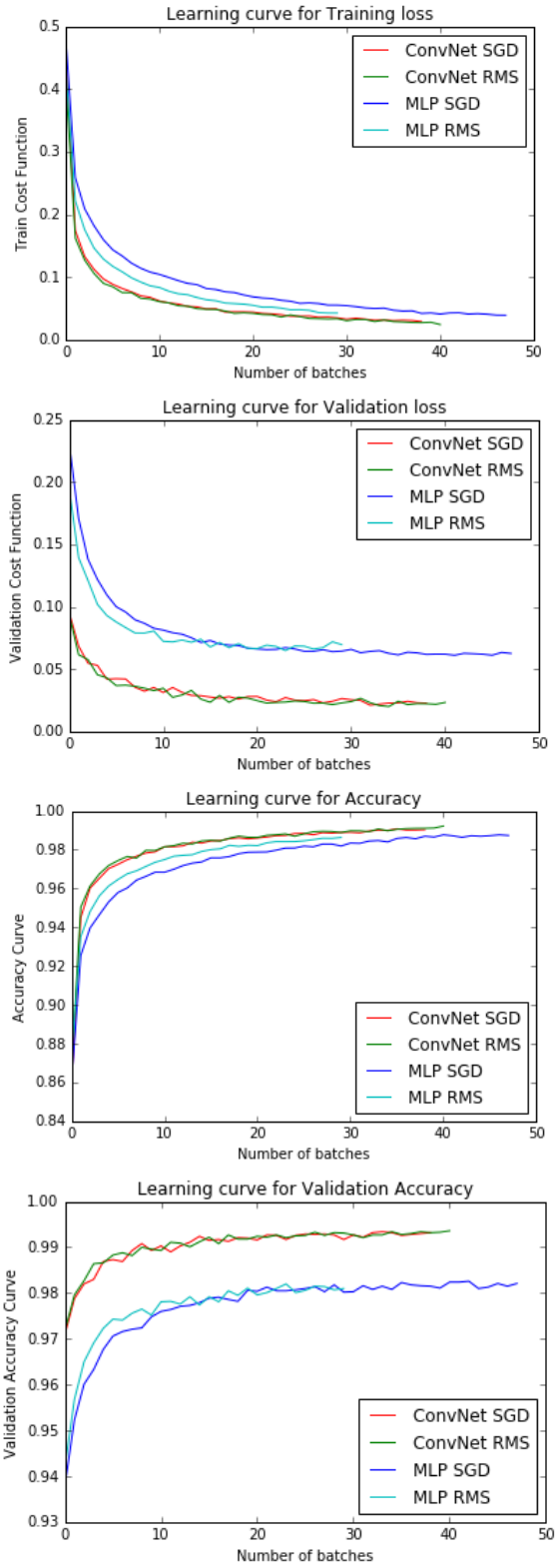


Fig8 The learning curves for loss functions and accuracy to compare the best Convolutional Network and MLP network configurations.

achieving better test accuracy (99.49%) than multilayer perceptron (98.20%). The best accuracy attained for MNIST dataset is 99.77% in literature using elasticity and drop connect. Our network configurations achieves is very close to the best test accuracy. Each layer of convolutional could be termed as feature detector/extractor, the accuracy of that features gets increases as the training progresses. The multilayer perceptron though faster in training, but their training on the basis of the gray value of the pixels rather than the spatial pattern.

Dropout and Early stopping for the Convolutional Networks (or for regular neural networks) gives better generalization and reduces overfitting , but also reduces time taken to training the convolutional neural network by huge margin. They should be always be initial for any deep learning network. SGDM and RMS prop are very close in test accuracy with SGDM beating RMS most of the time. RMS prop learning curves are much smoother than most SGDM. Though there exists SGDM parameter configuration that will outperform. The weight initializations by glorat unifom are more generalized and would normally give you better results. He uniform weight initializations claimed to give better results, but that will depend on the overall network parameter combination. Bigger kernel sizes for convolutional layer could be effective if the spatial variation has a wider spread, otherwise the smaller kernel sizes would be better to train and test the network.

MNIST dataset problem was just a beginning for convolutional networks, they are already extended to solve complex Imagenet challenges now. Convolutional Networks offer an interesting field to boost the learning methods for machines and with the high computation power currently training such networks is also not an issue.

REFERENCES

[1] http://cs231n.github.io/convolutional-networks/

[2] Dropout: A Simple Way to Prevent Neural Networks from Overfitting by Nitish Srivastava , Geofrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov , Department of Computer Science, University of Toronto, 10 Kings College Road, Rm 330.

[3] “Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis “ by Patrice Y. Simard, Dave Steinkraus, John C. Platt by Microsoft Research

[4] http://neuralnetworksanddeeplearning.com/chap6.html

[5] http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

[6] http://www.erogol.com/comparison-sgd-vs-momentum-vs-rmsprop-vs-momentumrmsprop/

[7] “Understanding the difficulty of training deep feedforward neural networks” by Xavier Glorot Yoshua Bengio

[8] “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification” by Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun , published on 6thbFb 2015.

[9] http://keras.io/

[10] http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf