EE 569: Homework #5: Image Recognition with CNN

Problem 1. CNN Training and Its Application to the MNIST Dataset

(a) CNN Architecture and Training

1.a.1 Abstract and Motivation

Convolutional neural network is widely used for image processing, computer vision, object recognition. Convolutional Neural Network combines both the feature extraction and classification problem into a single problem statement unlike typical image classification problem. Therefore, it is a self-learning process of extracting the features and classifying them automatically/ simultaneously. It involves lesser time to arrive at a solution for any type of classification/input in contrast to the typical image classification problem.

1.a.2 Theoretical Answers

Q1.

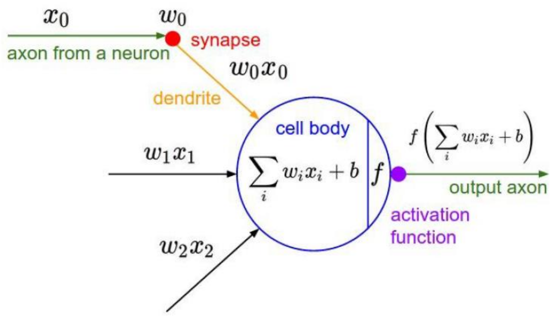
1. Fully connected Layer

- Each neuron in the output volume are connected to all the neurons in the previous layer and is activated by non-linear function – principle is same as multi-layer perceptron
- Let the input and output volume of fully connected layer be W_1 x H_1 x D_1 (width x height x depth) and W_2 x H_2 x D_2 (width x height x depth) respectively.
- Each pixel/node of W_2 x H_2 x D_2 is connected to corresponding each pixel/node in W_1 x H_1 x D_1 and each pixel/node of W_2 x H_2 x D_2 is obtained by adding 1 bias term to the corresponding connections of input volume. Therefore, fully connected layer has ($W_2 \cdot H_2 \cdot D_2 \cdot W_1 \cdot H_1 \cdot D_1 + W_2 \cdot H_2 \cdot D_2 \cdot 1$) trainable parameters and connections.
- Fully connected layers are essential for stretching the input volume and mapping it against the N dimensional output vector/layer (N- class problem).
- Fully connected layers is used for converting the spatial- spectral feature map into a feature dimension, adjust the anchor/feature vector with output pairing and provide the final classification decision as output

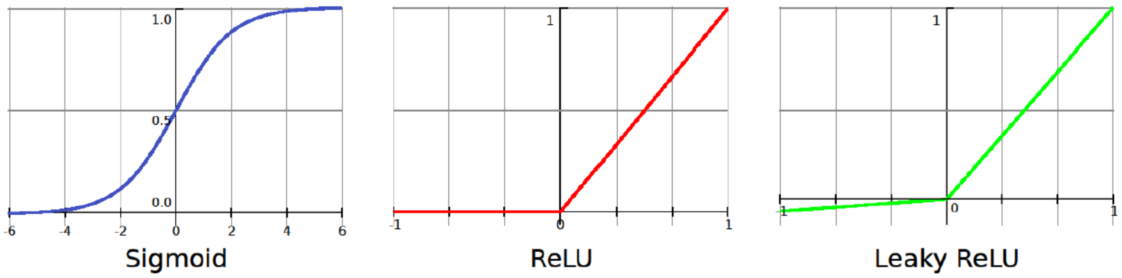
2. Convolution Layer

- Let the input and output volume of convolution layer be W_1 x H_1 x D_1 (width x height x depth) and W_2 x H_2 x D_2 (width x height x depth) respectively.
- Local receptive filter (kernel of convolution) of size F (ex: 5x5) is scanned/hovered over the image W_1 x H_1 x D_1, to obtain similar features across the entire image. Weights assigned to receptive filter are used to extract different features like edge, corners, end-points. Multiple Local receptive filters are used to slide over the input volume to form a stack of different feature extracted output layer.
- The term 'local receptive filter' is with analogy to neurons, the connections to the neuron are from several spatial local region of input volumes since it is impractical to connect neurons to all neurons in the previous volume which increases the complexity of computation.
- Convolution Layer preserves the spatial domain across all depth of input volume.

- Convolution operation can be interpreted as correlation / match filter operation. The bias term (b) is added to the inner product of F· F filter weights (w)and corresponding image F· F patch (x) as $\sum_i w(i)\, x(i)$. A non-linear function (f) is used to determine the activation of the neuron.



- Common Non-linear activation functions are sigmoid, ReLu (rectification linear), leaky ReLu.

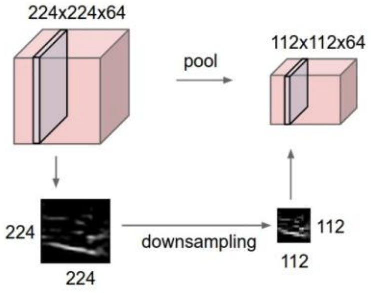


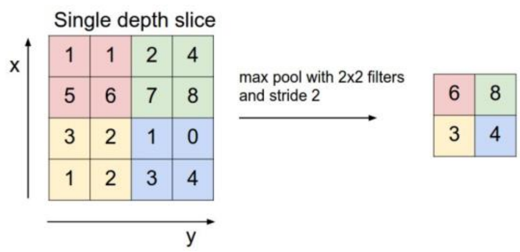
- Hyper parameters involved for Convolution Layer are:
  a. Number of filters (K): Determines the output volume depth (D_2) of convolution layer and the distinct features that can be extracted
  b. Spatial extent/filter size (F): Determines the output volume width(W_2) and height (H_2) of convolution layer.
  c. Stride (S): Distance to slide the centre of the filter bank (F)
  d. Amount/Degree of Zero Padding (P): Determines the output volume width(W_2) and height (H_2) of convolution layer
- Output of Convolution layer with K filters, F local receptive filter size, stride S, zero padding P is obtained by:
  $W_2 = \frac{(W1-F+2P)}{S+1}$ $H_2 = \frac{(H1-F+2P)}{S+1}$ $D_2 = K$
- Each pixel of $W_2 \times H_2 \times D_2$ is obtained using F· F receptive filter weights for each depth D_2 (or K) and D_2 (or K) bias terms. Therefore, convolution layer has
  (K (or D_2) · F · F + K (or D_2) · 1) trainable parameter.
- Each pixel of $W_2 \times H_2 \times D_2$ is connected to corresponding (F· F) neighbourhood in $W_1 \times H_1 \times D_1$ and each pixel of $W_2 \times H_2 \times D_2$ is obtained by adding 1 bias term to the corresponding connections of input volume. Therefore, convolution layer has
  ($W_2 \cdot H_2 \cdot D_2 \cdot F \cdot F + W_2 \cdot H_2 \cdot D_2 \cdot 1$)connections.

3. Max Pooling Layer

- Feature detection is more important and necessary than exact location, since the patterns need not be localized, it can be present at any position in the image. Therefore, relative position is important. In order to reduce these variations in identifying a pattern/object, the spatial resolution of an image is decreased so that the position variance can be reduced. Thus, reducing the output from shift and distortion variances.

- Convolution followed by dimension reduction is performed since, it provides greater degree of invariance to geometric transformations. It also reduces the parameters and computation in the network, and hence to also control overfitting.
- Hyper parameters involved for Max Pooling layer/ Sub Sampling Layer are:
    a. Spatial extent/filter size (F): Determines the output volume width(W_2) and height (H_2) of sub sampling layer/ max pooling layer
    b. Stride: Distance to slide the centre of the filter bank (F)
- Spatial extent/filter size for each depth of input volume is examined to obtain the maximum value and down-samples the input volume.

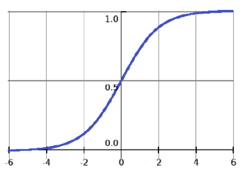




- Let the input and output volume of convolution layer be $W_1 \times H_1 \times D_1$ (width x height x depth) and $W_2 \times H_2 \times D_2$ (width x height x depth) respectively.
- Output of Max Pooling layer/ Sub Sampling Layer with F filter size, stride S is obtained by:
  $W_2 = \frac{(W1-F)}{S+1}$ $H_2 = \frac{(H1-F)}{S+1}$ $D_2 = D_1$
- Max Pooling layer/ Sub Sampling Layer has 0 trainable parameters if max pooling is adopted since maximum value is determined directly without any learning parameter.
- Each pixel of $W_2 \times H_2 \times D_2$ is connected to corresponding ($F \cdot F$) neighbourhood in $W_1 \times H_1 \times D_1$ and each pixel of $W_2 \times H_2 \times D_2$ is obtained by adding 1 bias term to the corresponding connections of input volume. Therefore, convolution layer has
  ($W_2 \cdot H_2 \cdot D_2 \cdot F \cdot F + W_2 \cdot H_2 \cdot D_2 \cdot 1$) connections.

4.Activation Function

- Non-Linear activation function is used which performs clipping operation.
- Activation function clips the negative value to 0 (sigmoid, ReLu) and to a small negative value in case of Leaky ReLu.
- This clipping is essential since convolutional neural network is a multi-layer network, the cascading effect is likely to interpret the correlations falsely and network fails to identify dissimilarities. Thus, negative correlation value at each layer is clipped to 0 to avoid error propagating to subsequent multi-layer (as explained using RECOS system).
- Different types of activation functions are:
    1. Sigmoid

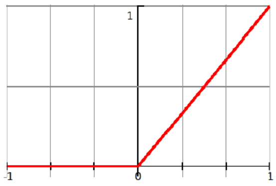


Sigmoid

- $\sigma(x) = \frac{1}{1+e^{-x}}$
- Limits the input to the range [0,1]
- Popular choice since it is interpreted as saturating “firing rate” of a neuron

- Cons:
    a. Saturated neurons can kill the gradient: The gradient flow through back propagation using chain rule becomes 0, so the weights are not updated.
    b. Sigmoid outputs are not zero centered: Weights of gradients are restricted to update in only 1 direction (all positive / all negatives) causing inefficiency.
    c. Exponential function is computationally expensive.

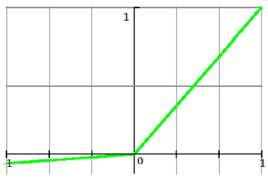
2. Rectified Linear Unit (ReLu)



ReLU

- ReLu(x)= max(0,x)
- Computationally efficient
- Converges faster than sigmoid/tanh activation functions about x6 times.
- Does not saturate in positive region
- Cons:
    a. ReLu outputs are not zero centered: Weights of gradients are restricted to update in only 1 direction (all positive / all negatives) causing inefficiency.

3. Leaky Rectified Linear Unit (Leaky ReLu)



Leaky ReLU

- Leaky ReLu (x) = max(0.01x,x) or max('small slope'x,x) = max(αx,x)
- Does not saturate
- Computationally efficient
- Converges faster than sigmoid/tanh activation functions about x6 times

4. Exponential Linear Unit (ELU)

5. Softmax Function/Classifier
- Softmax classifier are multinomial logistic regression
- It takes a vector of arbitrary real valued scores (s) and forces the output to be normalized in the interval (0,1)
- The scores obtained from cascade of convolution, max pool and fully connected layers are unnormalized log probabilities of the classes. Thus, exponential function is used while converting to probability.
- Score of class k (s_k) of total j classes is converted to probability using $\frac{e^{sk}}{\sum_j e^{sj}}$ = P(y=k|s_k)
- The output layer assigns the class corresponding to maximum likelihood or probability obtained among j classes to the input
- Softmax function/ classifier are widely used when dealing with categorical cross entropy.

Q2.

- Overfitting can be termed as the performance gap between training and testing accuracy with best performance for training data.
- The model parameters are fine tuned to a very granular level to obtain the best training accuracy. This leads to overfitting.
- Tuned in parameters fits best for only training data but performs poorly for testing data since it lacks generalization of data.
- When the number of training parameters are much larger than the training samples, it leads to overfitting.
- Pooling layer controls overfitting to a certain degree, since the number of parameters that needs to be updated are relatively less with the introduction of this layer.
- Dropout layer can be added in the convolutional neural network to overcome overfitting. Dropout consists of an hyperparameter p, which sets the p fraction of input neurons randomly to 0 (deactivated) at each update so that weights are not updated at iterations of training (weights are more evenly distributed) since when the number of training parameters are much larger than the training samples, it leads to overfitting.
- Other techniques to avoid overfitting are as follows:
  a. Cross Validation: Training samples are divided into train and validation set randomly for each iteration of fitting the model. The model parameters are fine tuned to fit train samples and are tested on validation set. This prevents overfitting since the randomness of sub division of train samples and validation samples lead to a generalized data of fit.
  b. Regularization: Penalty term/parameter can be introduced to the cost function so that the model is simple and avoids overfitting.
  c. More data can be used in comparison to number of training parameters. Data can be synthetically generated either by bootstrap or through data augmentation (flipping the images, rotation, or any other geometric modifications applied to image)

Q3. Why CNNs work much better than other traditional methods in many computer vision problems? You can use the image classification problem as an example to elaborate your points

- Typical image classification problem involves the steps of
  1. Pre-processing the data/image
  2. Feature extraction
  3. Classification
  4. Post processing of data/image
- Once the image is pre-processed, it has multiple features. Feature extraction includes the selection of optimal features to classify the input into different classes.
- The selection of the optimal feature is tedious and time consuming. It is often tried randomly.
- The number of features to be selected is also not well defined.
- The selected feature must have large discriminant power to classify the images into respective classes or groups.
- Classification involves techniques/methods like SVM (Support Vector Machine), linear classifier (minimum distance to class means classifier) which make use of the feature space.
- The optimization of loss function is used to update the weights of the classifier to correctly classify the data points/image.
- Typical image classification involves dealing with feature extraction and classification as two separate problem.

- Convolutional Neural Network combines both the feature extraction and classification problem into a single problem statement unlike typical image classification problem.
- The number of features to be extracted are defined by the number of filters used in the convolutional layers.
- Cascaded convolutional and max pooling layer will lead to a coarser extraction of features which is used to classify the images.
- The SoftMax classifier is used in Convolutional Neural Network. The optimization of loss function is used to update the weights of the filters and the weights of classifier for each iteration using back-propagation and chain rule.
- Therefore, it is a self-learning process of extracting the features and classifying them automatically/ simultaneously.
- It involves lesser time to arrive at a solution for any type of classification/input in contrast to the typical image classification problem.
- Performance/ classification accuracy is better in convolutional neural network compared to any other typical image classification problem.

Q4. Explain the loss function and the classical backpropagation (BP) optimization procedure to train such a convolutional neural network.

Loss function:

- Loss function measures the inconsistency between the predicted labels and the actual labels.
- Loss function is an important parameter to train data/network
- The loss function which can be used in convolutional neural network is SVM/softmax termed the cross-entropy loss which is of the form:

  $L_i = \log(\frac{e^{sk}}{\sum_j e^{sj}})$ where k denotes the kth class of score vector of dimension sx1
- SoftMax classifier weights and the filter (feature) weights of convolutional neural network are updated/adjusted each time so that the loss function is minimized
- The main objective is to minimize the loss function (cross-entropy) to obtain maximum gain/accuracy of training/testing image dataset.
- To minimize the cross-entropy loss, different types of optimization techniques like batch gradient descent, stochastic gradient descent, sequential gradient descent, adaptive moment estimation can be adopted.
- Model parameters are updated iteratively based on the evaluation of loss function on training data through back-propagation and chain rule. Weights are chosen such that they yield minimum loss (with margin).

Back-propagation:

- Convolutional Neural Network combines both the feature extraction and classification problem into a single problem statement unlike typical image classification problem. Therefore, it is a self-learning process of extracting the features and classifying them automatically/ simultaneously. It involves lesser time to arrive at a solution for any type of classification/input in contrast to the typical image classification problem.
- The key idea behind self-learning process is back-propagation and chain rule. It can also be termed as self-organizing property.
- Effective back-propagation training determines the performance of the system.

- For minimizing the loss function (non-convex optimization), there are multiple techniques like gradient descent, numerical, analytical method etc. But gradient descent is opted for its simple model and robustness.
- Back-propagation is essential since the filter weights must be updated/adjusted iteratively according to the gradient of loss function (minimizing loss function to improve performance)
- Initially, the kernel and the bias are initialized randomly, and loss function is calculated.
- Gradient of the loss function is calculated, and this must be propagated till the input, so that the intermediate weights are updated along the direction of negative gradient descent. This is achieved by chain rule.
- Each node is aware of local inputs and local outputs, from which the local gradients can also be computed. Local gradients are stored during the forward propagation of initial weights.
- During back-propagation, the gradient descent for each weight is obtained by simple multiplication of the incoming gradient with the local gradient and this is further passed on to nodes until the input is reached while propagating backwards from output. (General rule of thumb for chain rule: once a node/weight gets affected, then all its connections are as well affected)
- With the updated weights, the loss function is recomputed through forward propagation, the gradient descent of loss function is propagated backwards, and the weights are updated. The process continues till gradient descent of loss function converges.
- Back-propagation is used since the computational complexity is low/reduced. Since, local gradients are stored, multiplication with incoming gradient yields the gradient descent of weights accordingly. Hence, it saves memory and it can be applied to any sought of network (since the computational blocks can be easily cascaded (Chain rule).
- As and when the weights are updated, the system learns to recognize different features on its own (self-learning).

(b) Train LeNet-5 on MNIST Dataset

1.b.1 Abstract and Motivation

Convolutional neural network is widely used for image processing, computer vision, object recognition. Convolutional Neural Network combines both the feature extraction and classification problem into a single problem statement unlike typical image classification problem. Therefore, it is a self-learning process of extracting the features and classifying them automatically/ simultaneously. It involves lesser time to arrive at a solution for any type of classification/input in contrast to the typical image classification problem.

1.b.2 Approach and Procedure

[Ref 1]

Artificial neural network can be subclassified into:

a. Convolutional Neural Network (CNN): Widely used in image processing, computer vision, object recognition.
b. Recurrent Neural Network (RNN): Widely used in speech/ language processing.

Architecture of LeNet-5 (LeCun -Network 5) Convolutional Neural Network

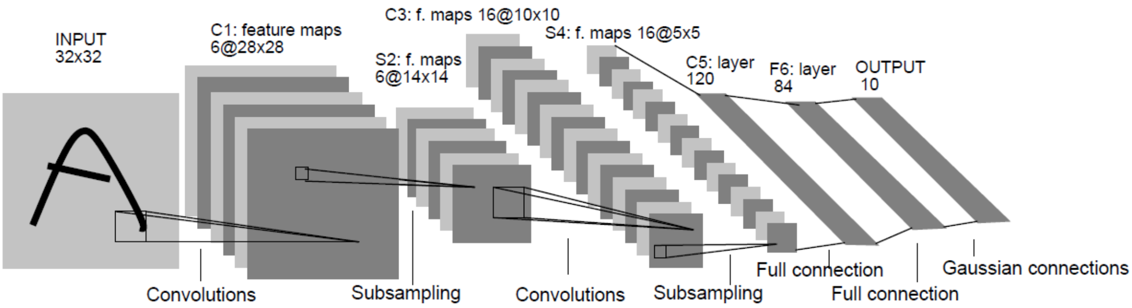


Fig: Architecture of LeNet-5 CNN

According to approximation theory, neural network is decomposed into following layers:

a. Input Layer
b. Hidden Layer (involves feature extraction and classification)
c. Output Layer

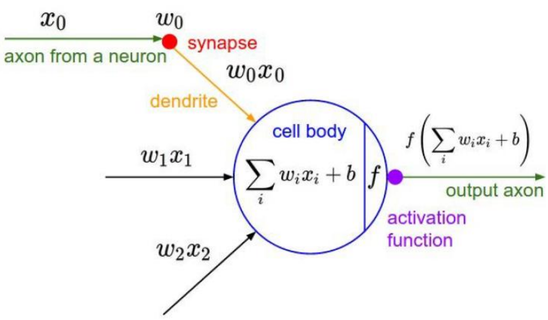
LeNet-5: consists of 7 layers

1. Input Layer:

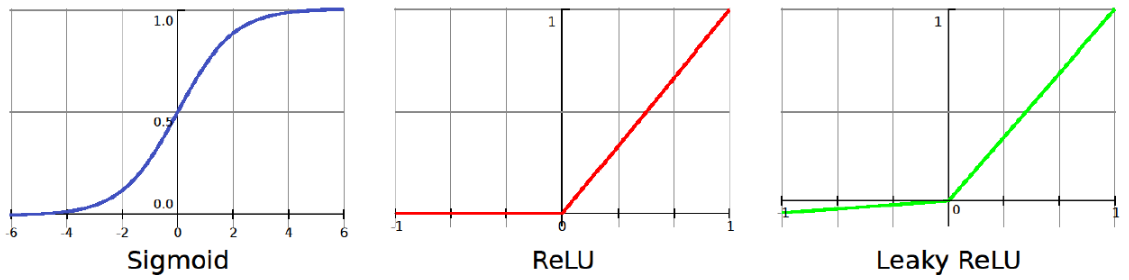
- 8-bit gray-scale image of 32x32 dimension.
- The input dimension is relatively higher since usually the features are present in 20x20 area. The reason being that potential features such as edges, corner points are localized at the centres for the highest-level feature detection.

2. C1- Convolutional layer:

- Local receptive filter (kernel of convolution) of size 5x5 is scanned/hovered over the image, to obtain similar features across the entire image. Weights assigned to receptive filter are used to extract different features like edge, corners, end-points.
- The term 'local receptive filter' is with analogy to neurons, the connections to the neuron are from several spatial local region of input volumes since it is impractical to connect neurons to all neurons in the previous volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron (filter size).
- 6 different 5x5 filter weights and 6 bias term are used to determine the feature map thus obtained as the output from the first stage. Convolution operation can be interpreted as correlation / match filter operation. The bias term (b) is added to the inner product of 5x5 filter weights and corresponding image 5x5 patch($\sum_i w(i)\ x(i)$). A non-linear function (f) is used to determine the activation of the neuron.



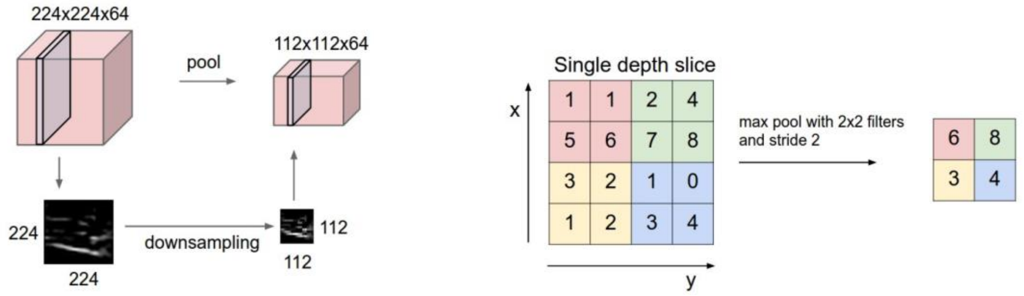
- Common Non-linear activation functions are sigmoid, ReLu (rectification linear), leaky ReLu.



- C1 layer has 156 trainable parameters: 6 filters *5*5 weights + 6 bias terms for each filter and 1,22,304 connections.
- Hyper parameters involved for Convolution Layer are:
    e. Number of filters: Determines the output volume of convolution layer and the distinct features that can be extracted
    f. Spatial extent/filter size: Determines the output volume of convolution layer
    g. Stride: Distance to slide the centre of the filter bank
    h. Amount/Degree of Zero Padding: Determines the output volume of convolution layer
- Output of C1 layer with 6 filters, 5x5 local receptive filter size, stride=1, 0 zero padding is 6 28x28 feature maps.

3. S2- Sub- sampling layer:

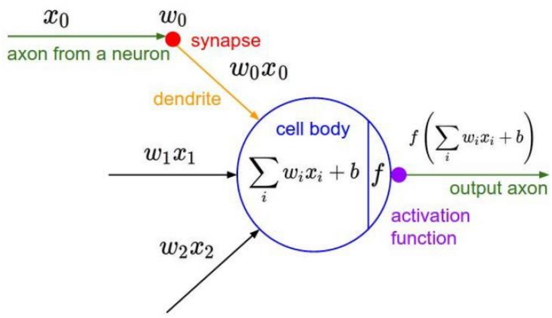
- Feature detection is more important and necessary than exact location, since the patterns need not be localized, it can be present at any position in the image. Therefore, relative position is important. In order to reduce these variations in identifying a pattern/object, the spatial resolution of an image is decreased so that the position variance can be reduced. Thus, reducing the output from shift and distortion variances.
- Convolution followed by dimension reduction is repeated since, it provides greater degree of invariance to geometric transformations. It also reduces the parameters and computation in the network, and hence to also control overfitting.
- Hyper parameters involved for Sub Sampling Layer are:
    c. Spatial extent/filter size: Determines the output volume of sub sampling layer
    d. Stride: Distance to slide the centre of the filter bank
- Spatial extent/filter size for each depth of output volume of C1 layer is examined to obtain the maximum value and down-samples the input volume.



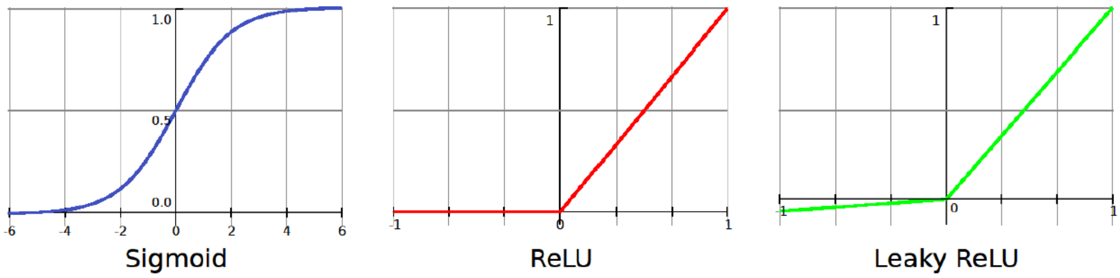
- S2 layer has 0 trainable parameters if max pooling is adopted and 5,880 connections
- Original paper uses 12 trainable parameters (4 units of 2x2 neighbourhood are added, multiplied by the trainable coefficient and trainable bias is added before the activation of non-linear function, since there are 6 filter/depth, with 2 trainable parameter each, there are 12 trainable parameters) and 5,880 connections.
- Output of S2 layer with 2x2 neighbourhood and stride =2 is 6 14x14 feature map.

4. C3- Convolutional layer:

- Local receptive filter (kernel of convolution) of size 16 5x5x6 is scanned/hovered over the image, to obtain similar features across the entire image. Weights assigned to receptive filter are used to extract different features like edge, corners, end-points.
- The term ‘local receptive filter’ is with analogy to neurons, the connections to the neuron are from several spatial local region of input volumes since it is impractical to connect neurons to all neurons in the previous volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron (filter size).
- 16 different 14x14x6 filter weights and 16 bias term are used to determine the feature map thus obtained as the output from the first stage. Convolution operation can be interpreted as correlation / match filter operation. The bias term (b) is added to the inner product of 14x14x6 filter weights and corresponding image 14x14x6 patch($\sum_i w(i)\ x(i)$). A non-linear function (f) is used to determine the activation of the neuron.



- Common Non-linear activation functions are sigmoid, ReLu (rectification linear), leaky ReLu.



- (Original paper) Output from S2 layer 14x14x6 layers are convolved differently to get 10x10x16 as follows:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

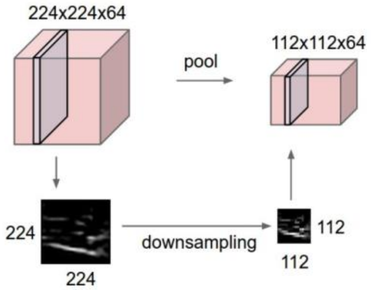
Fig : Each column indicates which feature map in S2 are combined by the units in a particular feature map of C3

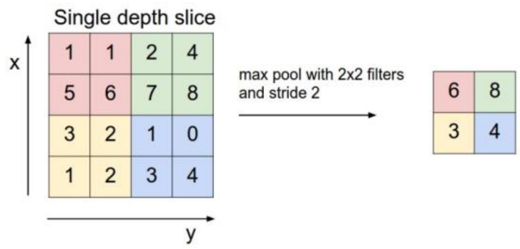
- (Original paper) The first six C3 feature maps takes inputs from every contiguous subset of three feature maps in S2. The next six C3 feature maps takes inputs from every contiguous subset of four feature maps in S2. The next three C3 feature map takes input from discontinuous subset of four. The last C3 feature map takes input from all feature maps in S2.
- (Original paper) Therefore C3 layer has (6 filters)* 5*5*(3 subsets) +(6 filters)* 5*5*(4 subset) + (3 filters)* 5*5*(4 discontinuous subsets)+ (1 filters)* 5*5*(6 subsets) = 1500 + 16 bias co-efficient (for 16 output filters) = 1516 training parameters with 1,51,600 connections.

- The reason for repeated convolution layers with non-complete connection scheme is to break the symmetry in convolutional network and different feature maps are combined to obtain different features from different sets of extracted feature maps.
- Hyper parameters involved for Convolution Layer are:
    i. Number of filters: Determines the output volume of convolution layer and the distinct features that can be extracted
    j. Spatial extent/filter size: Determines the output volume of convolution layer
    k. Stride: Distance to slide the centre of the filter bank
    l. Amount/Degree of Zero Padding: Determines the output volume of convolution layer
- Output of C1 layer with 16 filters, 5x5x6 local receptive filter size, stride=1, 0 zero padding is 16 10x10 feature maps.

5. S4- Sub- sampling layer:

- Feature detection is more important and necessary than exact location, since the patterns need not be localized, it can be present at any position in the image. Therefore, relative position is important. In order to reduce these variations in identifying a pattern/object, the spatial resolution of an image is decreased so that the position variance can be reduced. Thus, reducing the output from shift and distortion variances.
- Convolution followed by dimension reduction is repeated since, it provides greater degree of invariance to geometric transformations. It also reduces the parameters and computation in the network, and hence to also control overfitting.
- Hyper parameters involved for Sub Sampling Layer are:
    e. Spatial extent/filter size: Determines the output volume of sub sampling layer
    f. Stride: Distance to slide the centre of the filter bank
- Spatial extent/filter size for each depth of output volume of C3 layer is examined to obtain the maximum value and down-samples the input volume.

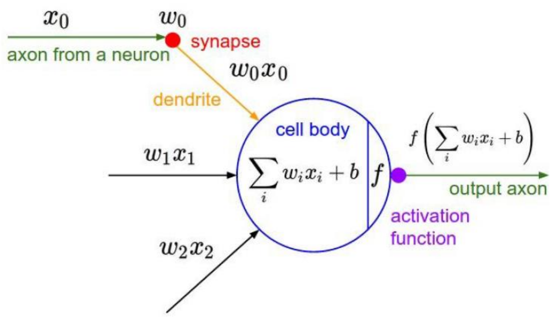




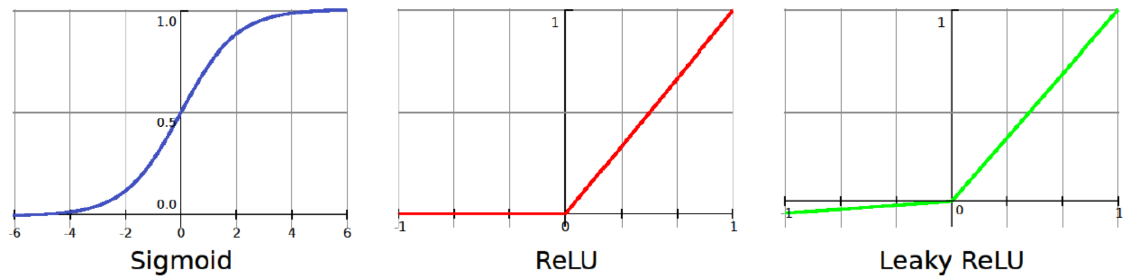
- S2 layer has 0 trainable parameters if max pooling is adopted and 5,880 connections
- Original paper uses 32 trainable parameters (4 units of 2x2 neighbourhood are added, multiplied by the trainable coefficient and trainable bias is added before the activation of non-linear function, since there are 16 filter/depth, with 2 trainable parameter each, there are 32 trainable parameters) and 2,000 connections.
- Output of S2 layer with 2x2 neighbourhood and stride =2 is 16 5x5 feature map.

6. C5- Convolutional Layer:

- Each output is connected to all 5x5 neighborhood of 16 feature maps, hence it is fully connected.
- 120 different 5x5x16 filter weights and 120 bias term are used to determine the feature map thus obtained as the output from the first stage. Convolution operation can be interpreted as correlation / match filter operation. The bias term (b) is added to the inner product of 5x5x16 filter weights and corresponding image 5x5x16 patch($\sum_i w(i)\ x(i)$). A non-linear function (f) is used to determine the activation of the neuron.



- Common Non-linear activation functions are sigmoid, ReLu (rectification linear), leaky ReLu.



- C5 layer is fully connected and thus have 48,120 trainable connections.
- Hyper parameters involved for Convolution Layer are:
  m. Number of filters: Determines the output volume of convolution layer and the distinct features that can be extracted
  n. Spatial extent/filter size: Determines the output volume of convolution layer
  o. Stride: Distance to slide the centre of the filter bank
  p. Amount/Degree of Zero Padding: Determines the output volume of convolution layer
- Output of C5 layer with 120 filters, 5x5x16 local receptive filter size, stride=1, 0 zero padding is 120x1 feature maps.

7. F6- Layer:

- F6 layer is fully connected (each neuron in the output volume are connected to all the neurons in the previous layer C5 and is activated by non-linear function – principle is same as multi-layer perceptron) and thus have 10,164 (each of 84 output neurons is connected to 120 previous layer neurons= 84*120 + 84 bias terms) trainable connections.
- Output of F6 layer is 84x1 feature maps.
- It is equivalent to dimensionality reduction

8. Output Layer:

- Output layer is fully connected (each neuron in the output volume are connected to all the neurons in the previous layer C5 and is activated by non-linear function – principle is same as multi-layer perceptron) and thus have 850 (each of 10 output neurons is connected to 84 previous layer neurons= 84*10 + 10 bias terms with softmax non-linear activation) trainable connections.
- Output of F6 layer is 10x1 with the probability of the input image belonging to each of the 10 classes. The input image belongs to the class corresponding to the maximum probability in the output vector.

Algorithm/ Implementation:

Step 1: Load the training and test images from the mnist database/set.
Step 2: The training and testing labels are of type int. Since we are using categorical entropy as our loss function, the output labels have to be converted to categorical label.
Step 3: The model of the network is to be created by adding subsequent layers of operation
Step 4: The model involves addition of sequential convolution layer with 6 filters of 5x5 with non-linear activation, max pooling layer of size 2x2, convolution layer with 16 filters of size 5x5 with non linear activation, max pooling layer of size 2x2, flatten the output, fully connected convolutional layer of size 120 with non-linear activation, fully connected layer of size 84 with non-linear activation, fully connected layer of size 10 with 'softmax' classifier.
Step 5: Model is compiled by specifying the loss function, optimizer and metric for training the data using back-propagation.
Step 6: Training data is passed to the model so that the data fits the model with specified loss function.
Step 7: Test data is used as a validation set while training for each epoch
Step 8: The model is run over 100 epochs
Step 9: Trained model is used to predict the labels of test data and corresponding loss and accuracy is calculated.
Step10: Plot of training and testing accuracy are obtained for different parameter setting.
Step11: Mean and variance of training and testing accuracies for 5 different parameter setting are calculated

1.b.3 Results

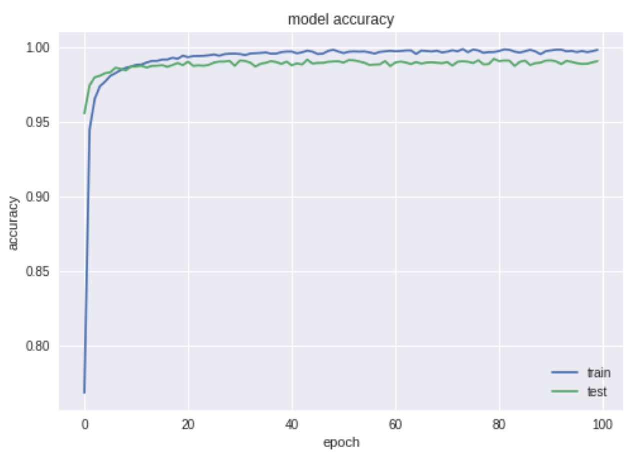
1.b.4 Discussion

Case	Optimizer	Parameter - Learning rate	Activation Function	Drop out	Kernel Initializer	Bias Initializer	Training Loss	Training Accuracy	Testing loss	Testing accuracy
1	Adam	0.001	relu	-	glorot_uniform	zeros	0.00803469318845	0.998983333	0.11644755	0.9904
2	Adam	0.001	relu	0.5	glorot_uniform	zeros	0.0033672182	0.999016	0.05667	0.9888
3	Adam	0.001	relu	0.7	glorot_uniform	zeros	0.017789484596780192	0.9965333333333334	0.0512579882632941	0.9882
4	SGD	0.01	relu	0.7	glorot_uniform	zeros	0.027027977332331164	0.9923	0.058032052010053306	0.985
5	SGD	0.01 M:0.1	relu	0.7	glorot_uniform	zeros	0.020035630772669143	0.9938	0.0678319830433668	0.9834
6	SGD	0.01	relu	0.7	Random_uniform	Random_uniform	0.007141029279905713	0.9976666666667	0.03559319313451499	0.9925
7	SGD	0.01	relu	0.7	glorot_normal	glorot_normal	0.021534922409752258	0.9935333333333334	0.052175959912111554	0.987

8	SGD	0.01	elu	0.7	glorot_normal	glorot_normal	0.02556776511237064	0.991816666666667	0.05702773319681	0.9812
9	Adam	0.001	relu	0.7	glorot_normal	glorot_normal	0.013018899046330383	0.996733333333334	0.04999525197438634	0.9886
10	Adam	0.001	sigmoid	0.7	glorot_normal	glorot_normal	0.011996125300594334	0.996166666666666	0.04150993278018432	0.9888
11	Adam	0.001	linear	0.7	glorot_normal	glorot_normal	14.471094508361816	0.1021833333333333	14.490167527770996	0.101
12	SGD	0.01	relu	-	glorot_uniform	zeros	0.024222296557192265	0.9930833333333333	0.058978204058179835	0.9845
13	Adam	0.001	relu	0.7	Random_uniform	Random_uniform	0.003408039682253108	0.9991333333333333	0.030509147875125563	0.9928

- **Case 1~13 are different parameter setting with case 1,2,6,10,13 being the 5-parameter different setting**
- **Case 13 is the best parameter setting to achieve highest accuracy on test set.**

Plot of model loss vs epoch and model accuracy vs epoch for all the cases listed above in the table



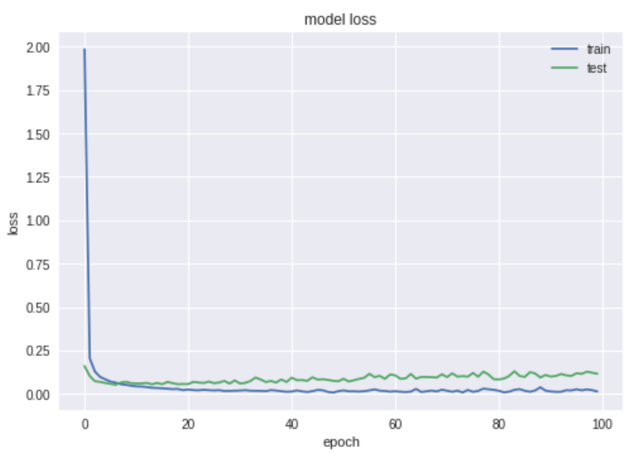


Fig 1.1: Plot of model loss vs epoch for Case 1 Fig 1.2: Plot of model accuracy vs epoch for Case 1

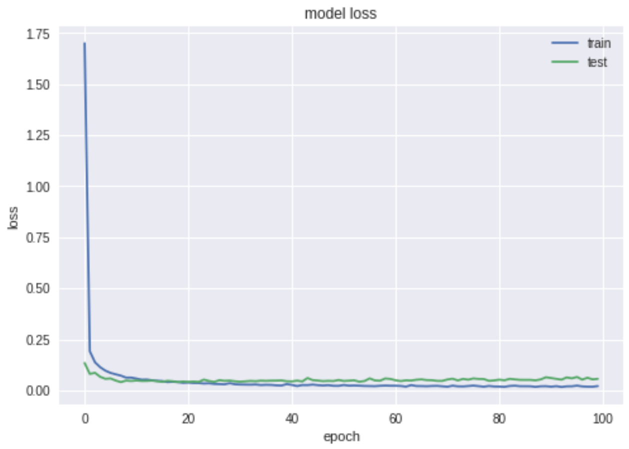


Fig 1.3: Plot of model loss vs epoch for Case 2

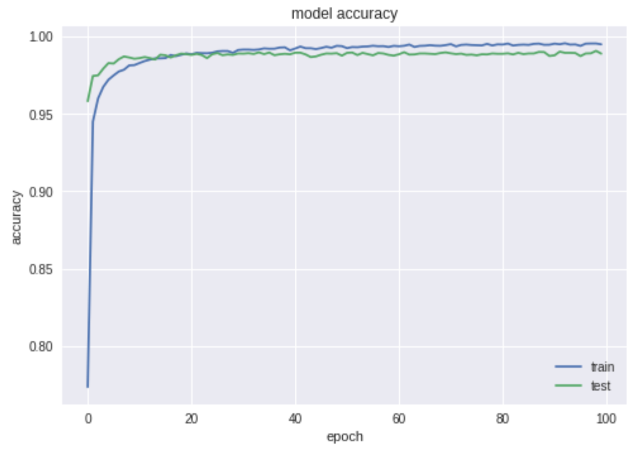


Fig 1.4: Plot of accuracy loss vs epoch for Case 2

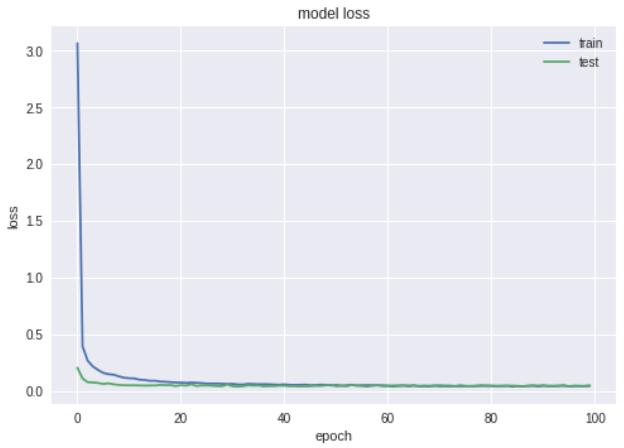


Fig 1.5: Plot of model loss vs epoch for Case 3

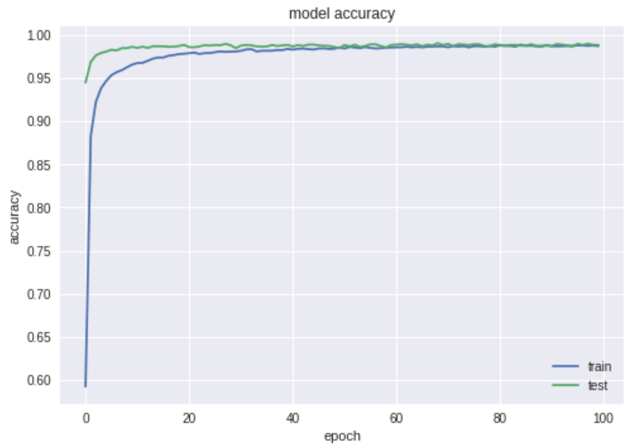


Fig 1.6: Plot of model accuracy vs epoch for Case 3

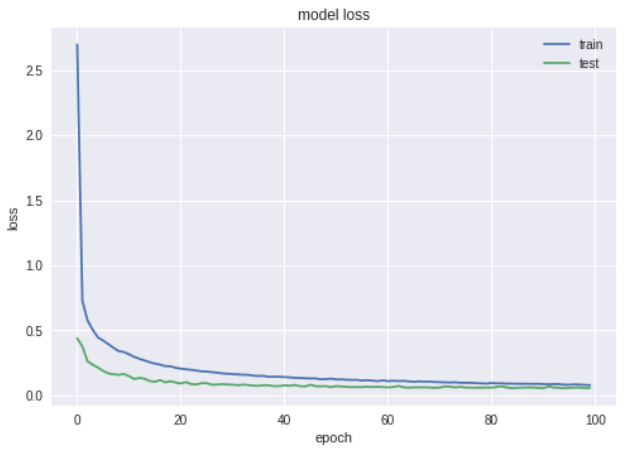


Fig 1.7: Plot of model loss vs epoch for Case 4

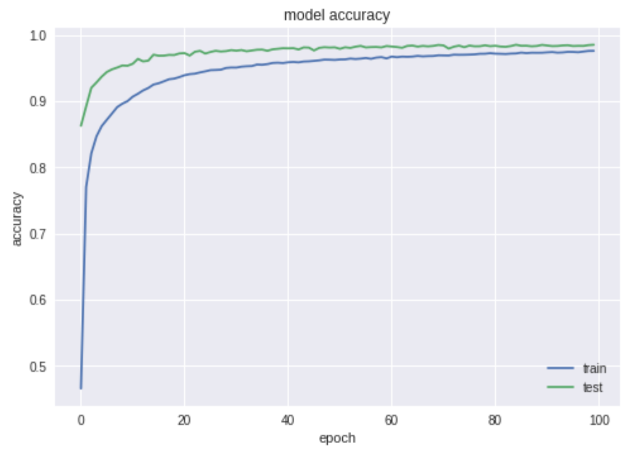


Fig 1.8: Plot of model accuracy vs epoch for Case 4

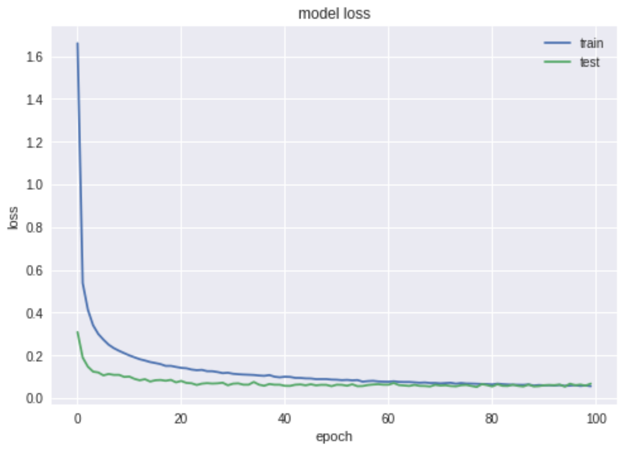


Fig 1.9: Plot of model loss vs epoch for Case 5

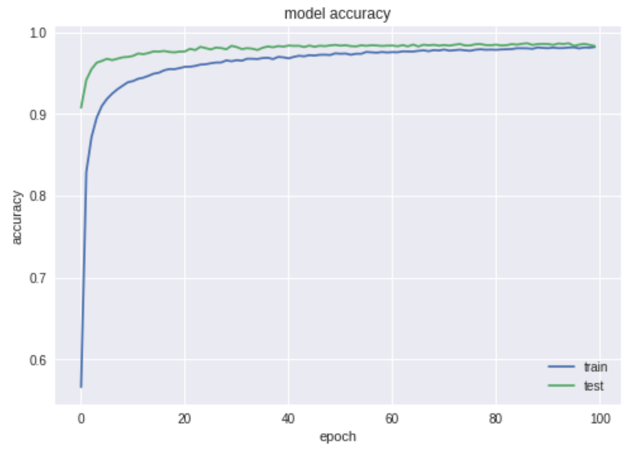


Fig 1.10: Plot of model accuracy vs epoch for Case 5

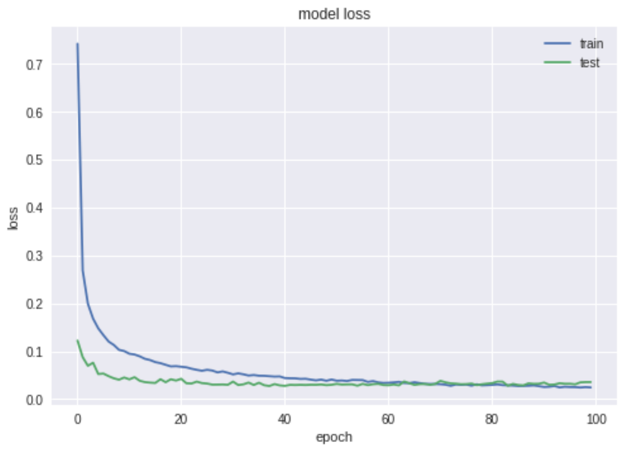


Fig 1.11: Plot of model loss vs epoch for Case 6

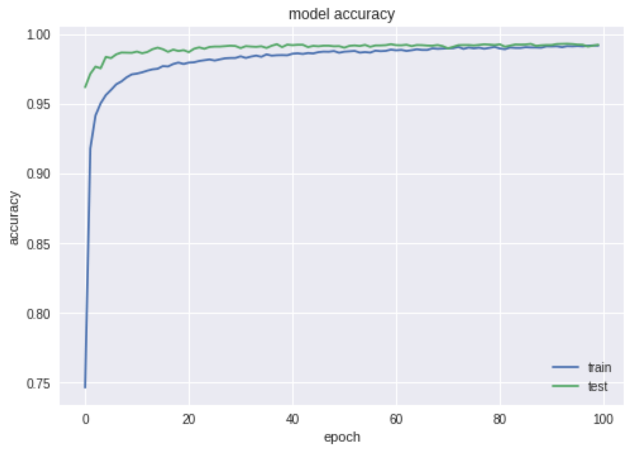


Fig 1.12: Plot of model accuracy vs epoch for Case 6

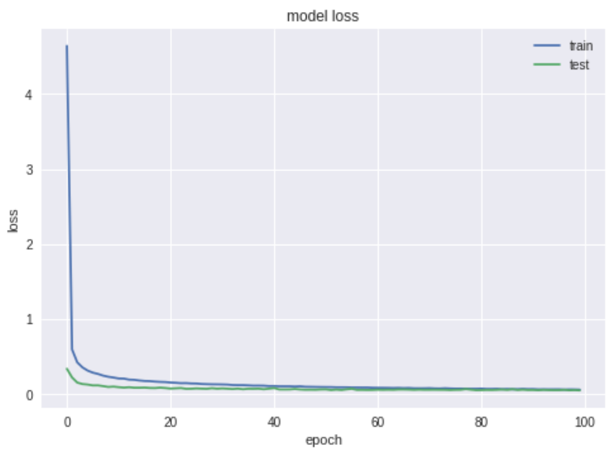


Fig 1.13: Plot of model loss vs epoch for Case 7

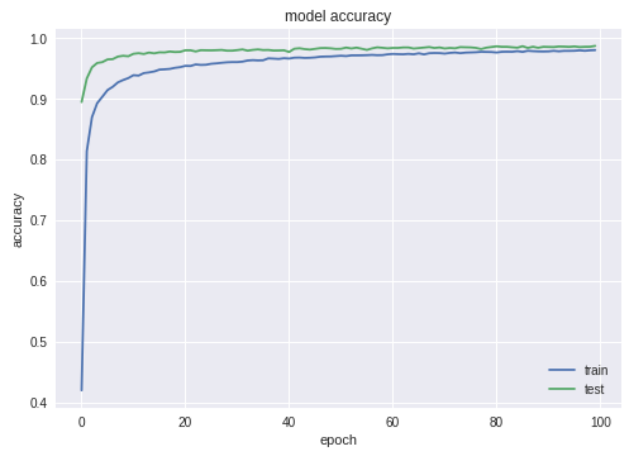


Fig 1.14: Plot of model accuracy vs epoch for Case 7

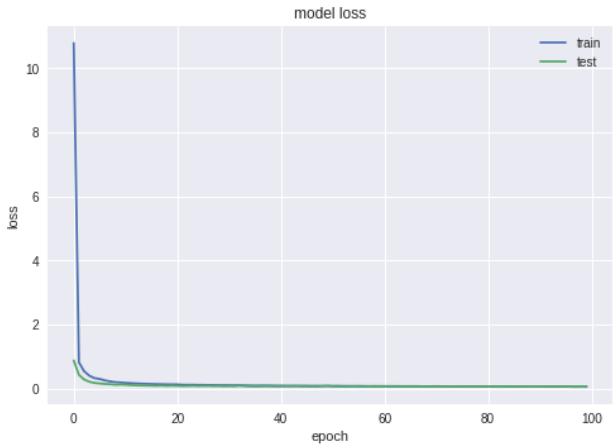


Fig 1.15: Plot of model loss vs epoch for Case 8

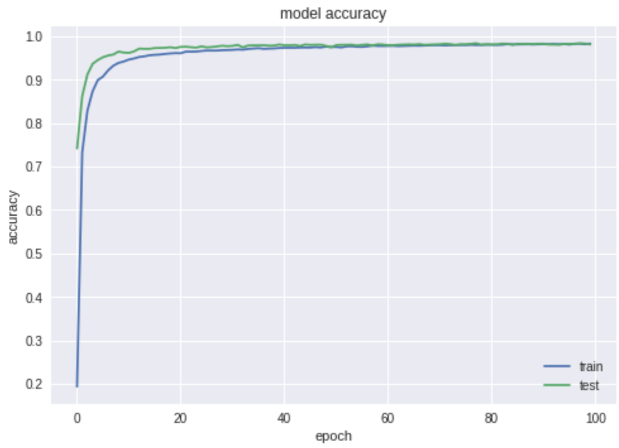


Fig 1.16: Plot of model accuracy vs epoch for Case 8

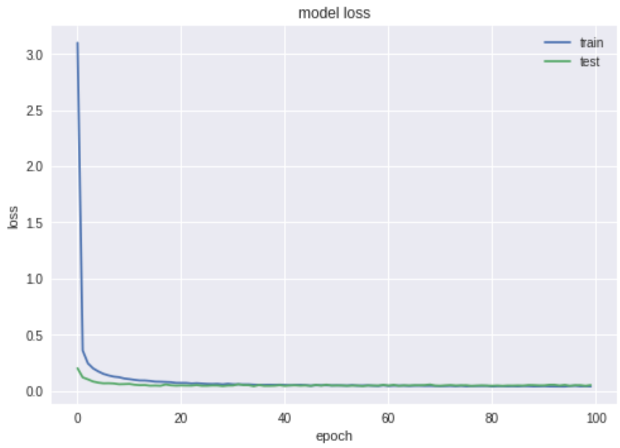


Fig 1.17: Plot of model loss vs epoch for Case 9

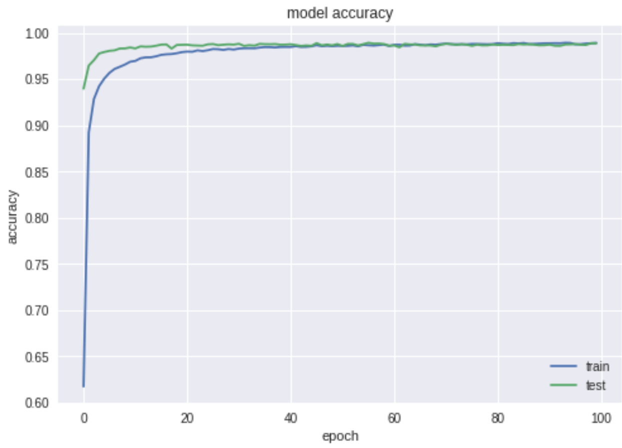


Fig 1.18: Plot of model accuracy vs epoch for Case 9

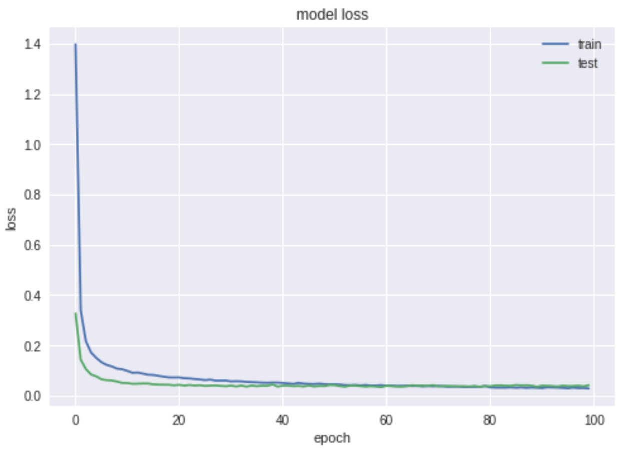


Fig 1.19: Plot of model loss vs epoch for Case 10

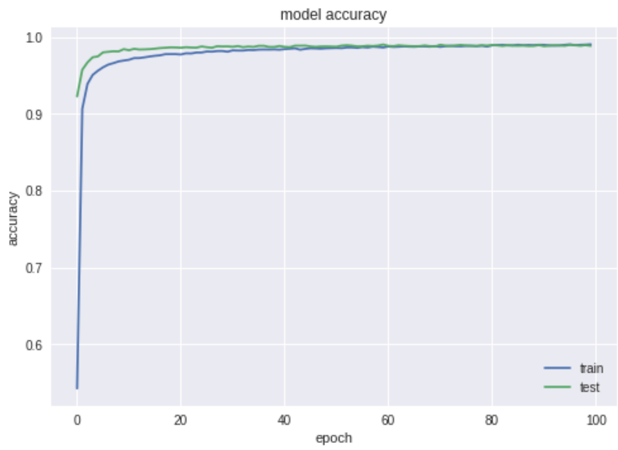


Fig 1.20: Plot of model accuracy vs epoch for Case 10

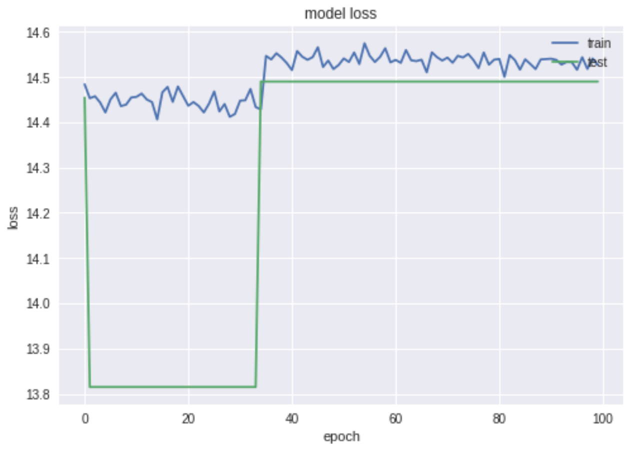


Fig 1.21: Plot of model loss vs epoch for Case 11

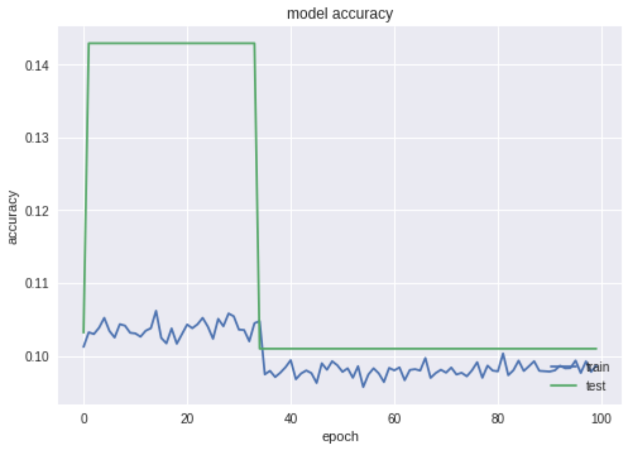


Fig 1.21: Plot of model accuracy vs epoch for Case 11

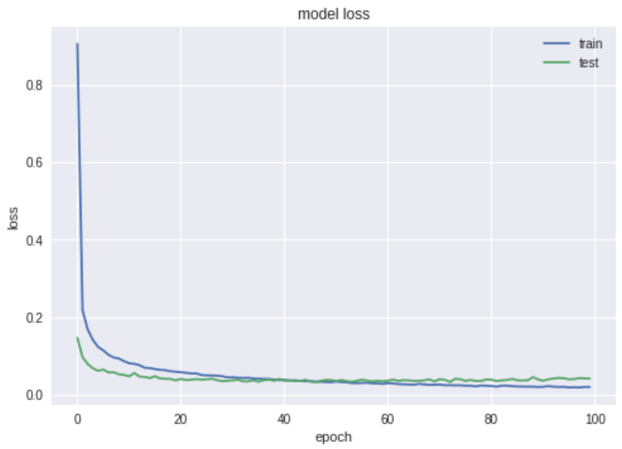


Fig 1.23: Plot of model loss vs epoch for Case 12

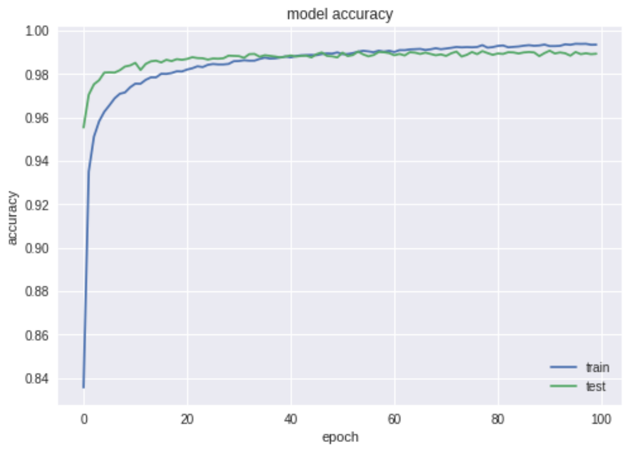


Fig 1.24: Plot of model accuracy vs epoch for Case 12

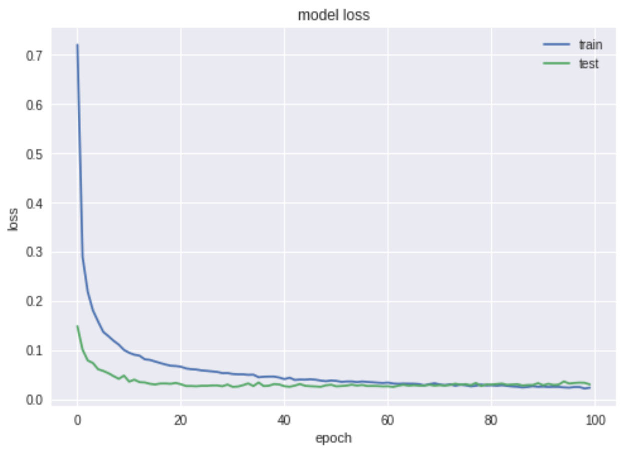


Fig 1.25: Plot of model loss vs epoch for Case 13

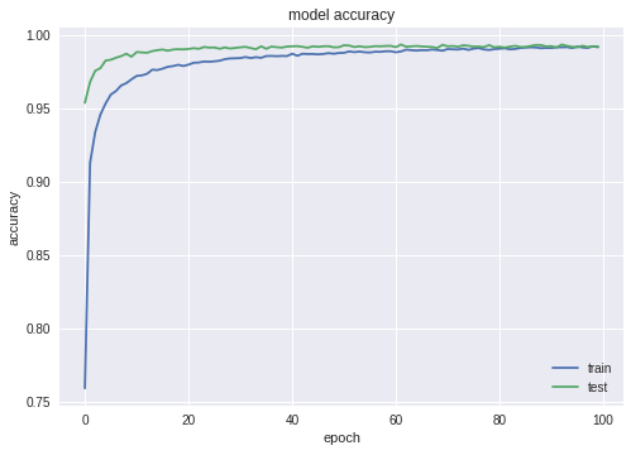


Fig 1.26: Plot of model loss vs epoch for Case 13

1.b.4 Discussion

Case 1: Overfitting

Table 1.b.4.1 Discussion about overfitting

Case	Optimizer	Parameter - Learning rate	Activation Function	Dropout	Kernel Initializer	Bias Initializer	Training Loss	Training Accuracy	Testing loss	Testing accuracy
1	Adam	0.001	relu	-	glorot_ uniform	zeros	0.0080	0.99898	0.1164	0.9904
2	Adam	0.001	relu	0.7	glorot_ uniform	zeros	0.0177	0.99653	0.0512	0.9882

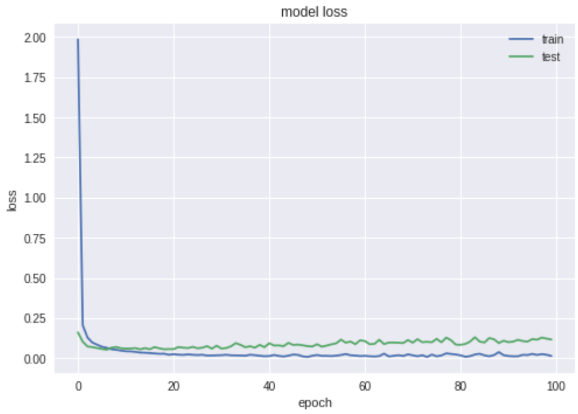


Fig1: Plot of model loss vs epoch for Case 1

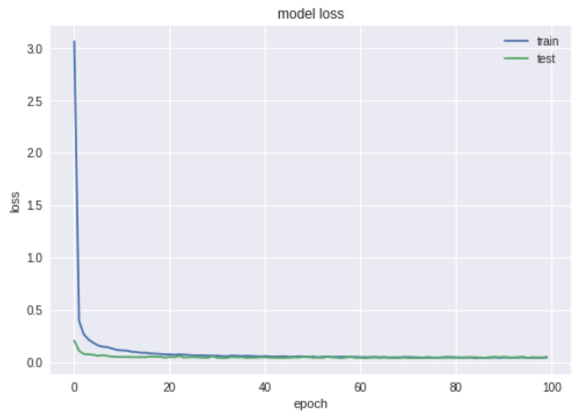


Fig2: Plot of model loss vs epoch for Case 2

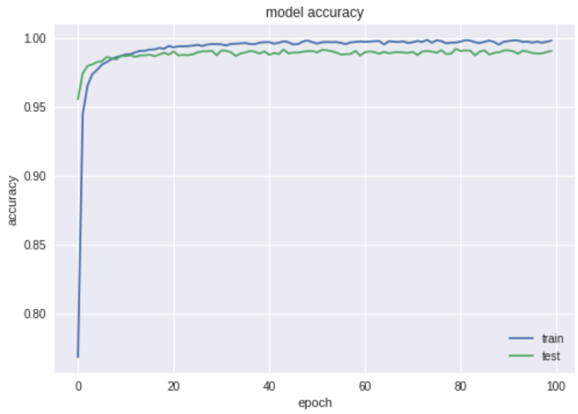


Fig3: Plot of model accuracy vs epoch for Case 1

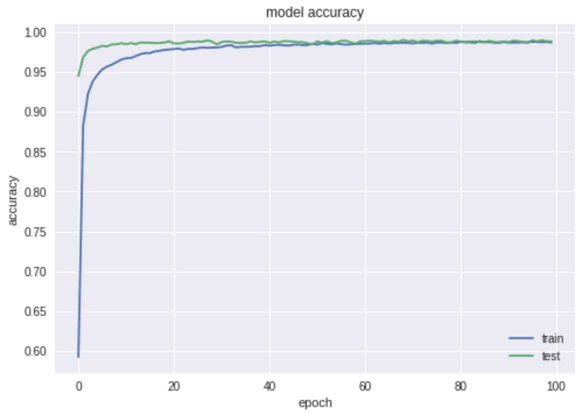


Fig4: Plot of model accuracy vs epoch for Case 2

From Fig 1 and 2, it can be observed that there exists a performance gap between testing and training data with increasing epochs. Test data loss reaches a certain minimum and then starts increasing. Thereby, the accuracy of test data drops while the training data has better accuracy. This is termed overfitting, which can be avoided by adding a dropout layer as in Case 2. Dropout consists of an hyperparameter p, which sets the p fraction of input neurons randomly to 0 (deactivated) at each update so that weights are not updated at iterations of training (weights are more evenly distributed) since when the number of training parameters are much larger than the training samples, it leads to overfitting. Thus, in case 2, there is no overfitting problem.

Case 2: Learning Rate

Table 1.b.4.2 Discussion about optimum choice of Learning rate

Case	Optimizer	Parameter - Learning rate	Activation Function	Dropout	Kernel Initializer	Bias Initializer	Training Loss	Training Accuracy	Testing loss	Testing accuracy
1	SGD	0.01	relu	-	glorot_uniform	zeros	0.0080982819848914	0.9977833333333334	0.05231403939670126	0.9863
2	SGD	0.001	relu	-	glorot_uniform	zeros	0.045217217237393685	0.9863	0.06469224581671879	0.9786
3	SGD	0.0001	relu	-	glorot_uniform	zeros	0.18376517725779365	0.9467333333333333	0.1752161616858095	0.9465
4	SGD	0.015	relu	-	glorot_uniform	zeros	0.0067517030866513245	0.99795	0.09693387291407563	0.9798

From Table 1.b.4.2, comparing case 1,2 and 3: it is evident that, as the learning rate is small (slow), the accuracy of both training and testing data decreases over the same epoch. As the learning rate is small, for each iteration the weights are updated by smaller increments, thus taking more time for gradient descent to converge. Unlike case 4, the accuracy of testing data has decreased compared to case 1 since taking larger increment steps of gradient descent for convergence might be misleading sometimes. Therefore, learning rate of 0.010 is optimum for SGD (Stochastic gradient descent) optimizer.

Case 3: Batch Size

Table 1.b.4.3 Discussion about optimum choice of Batch Size

Case	Optimizer	Parameter – Batch Size	Activation Function	Dropout	Kernel Initializer	Bias Initializer	Training Loss	Training Accuracy	Testing loss	Testing accuracy
1	SGD	100	relu	-	glorot_uniform	zeros	0.0080982819848914	0.9977833333333334	0.05231403939670126	0.9863
2	SGD	200	relu	-	glorot_uniform	zeros	0.01729264209665610 4	0.9947166666666667	0.06244659750371029	0.9826
3	SGD	1000	relu	-	glorot_uniform	zeros	0.06167840185218180 5	0.9807	0.0753722182	0.9756

									4915614	

From Table 1.b.4.3, comparing case 1,2 and 3: it is evident that, as the batch size increases the accuracy of both training and testing data decreases over the same epoch. As the batch size is small, for each iteration the weights are updated by frequently, thus taking less time for gradient descent to converge. Therefore, batch size of 100 is optimum for SGD (Stochastic gradient descent) optimizer.

Case 4: Optimizers

Table 1.b.4.4 Discussion about optimum choice of Optimizer

Case	Optimizer	Parameter - Learning rate	Activation Function	Dropout	Kernel Initializer	Bias Initializer	Training Loss	Training Accuracy	Testing loss	Testing accuracy
1	Adam	0.001	relu	-	glorot_uniform	zeros	0.00803469318845	0.998983333	0.11644755	0.9904
2	SGD	0.01	relu	-	glorot_uniform	zeros	0.024222296557192265	0.99308333333333333	0.058978204058179835	0.9845
3	Adam	0.001	relu	0.7	glorot_uniform	zeros	0.017789484596780192	0.99653333333333334	0.0512579882632941	0.9882
4	SGD	0.01	relu	0.7	glorot_uniform	zeros	0.027027977332331164	0.9923	0.058032052010053306	0.985
5	Adam	0.001	relu	0.7	glorot_normal	glorot_normal	0.013018899046330383	0.99673333333333334	0.04999525197438634	0.9886
6	SGD	0.01	relu	0.7	glorot_normal	glorot_normal	0.021534922409752258	0.99353333333333334	0.052175959912111554	0.987
7	Adam	0.001	relu	0.7	Random_uniform	Random_uniform	0.003408039682253108	0.99913333333333333	0.030509147875125563	0.9928
8	SGD	0.01	relu	0.7	Random_uniform	Random_uniform	0.007141029279905713	0.99766666666666667	0.03559319313451499	0.9925

Optimizer: Adam (Adaptive moment estimation) vs SGD (Stochastic Gradient descent)

From Table 1.b.4.4, comparing training and testing accuracies of all the cases it is evident that Adam optimizer performs better in comparison with SGD. Adam are used widely in deepnets while SGD operates better in shallow nets. Since, convolutional neural network is a deep network, Adam is an optimum choice for optimizer. Adam optimizer works well with little tuning of hyperparameter.

Case 5: Activation function

Table 1.b.4.5 Discussion about optimum choice of Activation function

Case	Optimizer	Parameter - Learning rate	Activation Function	Dropout	Kernel Initializer	Bias Initializer	Training Loss	Training Accuracy	Testing loss	Testing accuracy
1	SGD	0.01	elu	0.7	glorot_normal	glorot_normal	0.02556776511237064	0.9918166666666667	0.05702773319681	0.9812
2	Adam	0.001	relu	0.7	glorot_normal	glorot_normal	0.0130188990463303 83	0.9967333333333334	0.049995251974386 34	0.9886
3	Adam	0.001	sigmoid	0.7	glorot_normal	glorot_normal	0.0119961253005943 34	0.9961666666666666	0.041509932780184 32	0.9888
4	Adam	0.001	linear	0.7	glorot_normal	glorot_normal	14.4710945083618 16	0.10218333333333333	14.4901675277709 96	0.101

From Table 1.b.4.5, comparing all cases: it is evident that, a non-linear activation function is to be used from comparison of case 4 with rest cases. Among non-linear activation functions, relu (Rectified Linear Unit) performs better than elu (Exponential linear unit) and sigmoid. Therefore, relu (Rectified Linear Unit) is optimum for non-linear activation function.

Conclusion

Best performing parameters and output is given by:

Optimizer: Adam
Activation function: ReLu
Dropout: 0.7
Learning rate: 0.001
Kernel initializer: RandomUniform
Bias initializer: RandomUniform
Training Loss: 0.003408039682253108
Testing Loss: 0.030509147875125563
Training accuracy: 0.9991333333333333
Testing accuracy: 0.9928

5 different parameters setting:

Case	Optimizer	Parameter - Learning rate	Activation Function	Drop out	Kernel Initializer	Bias Initializer	Training Loss	Training Accuracy	Testing loss	Testing accuracy
1	Adam	0.001	relu	-	glorot_uniform	zeros	0.00803469318845	0.998983333	0.11644755	0.9904
2	Adam	0.001	relu	0.5	glorot_uniform	zeros	0.0033672182	0.999016	0.05667	0.9888
6	SGD	0.01	relu	0.7	Random_uniform	Random_uniform	0.007141029279905713	0.9976666666666667	0.03559319313451499	0.9925
10	Adam	0.001	sigmoid	0.7	glorot_normal	glorot_normal	0.011996125300594334	0.9961666666666666	0.04150993278018432	0.9888
13	Adam	0.001	relu	0.7	Random_uniform	Random_uniform	0.003408039682253108	0.9991333333333333	0.030509147875125563	0.9928

The mean and variance of training accuracies over 5 parameter setting are 0.998193 and 1.3138e-06 respectively.

The mean and variance of testing accuracies over 5 parameter setting are 0.99066 and 2.9904e-06 respectively.

The variance of training accuracies is less in comparison with variance of test accuracies.

(c) Apply trained network to negative images

1.c.1 Abstract and Motivation

Convolutional neural network is widely used for image processing, computer vision, object recognition. Convolutional Neural Network combines both the feature extraction and classification problem into a single problem statement unlike typical image classification problem. Therefore, it is a self-learning process of extracting the features and classifying them automatically/ simultaneously. It involves lesser time to arrive at a solution for any type of classification/input in contrast to the typical image classification problem.

1.c.2 Approach and Procedure

[Ref 1] as mentioned in 1.b.2

Algorithm/ Implementation:

Step 1: Load the training and test images (negation of training images) from the MNIST database/set.
Step 2: The training and testing labels are of type int. Since we are using categorical entropy as our loss function, the output labels have to be converted to categorical label.
Step 3: The model of the network is to be created by adding subsequent layers of operation
Step 4: The model involves addition of sequential convolution layer with 6 filters of 5x5 with non-linear activation, max pooling layer of size 2x2, convolution layer with 16 filters of size 5x5 with non linear activation, max pooling layer of size 2x2, flatten the output, fully connected convolutional layer of size 120 with non-linear activation, fully connected layer of size 84 with non-linear activation, fully connected layer of size 10 with 'softmax' classifier.
Step 5: Model is compiled by specifying the loss function, optimizer and metric for training the data using back-propagation.
Step 6: Training data is passed to the model so that the data fits the model with specified loss function and the model is run over 100 epochs
Step 7: Trained model is used to predict the labels of test data and corresponding loss and accuracy is calculated
Step 8: For designing my own network of original and negative images, training and test set is augmented by stacking both original and negative images training and test data and labels as a new training and testing data, label set respectively.
Step 9: Step 2~7 are repeated to obtain the loss and accuracy

1.c.3 Results

1.c.3.1

By training convolutional neural network with original training image and labels and testing it on negative test images and labels, the following results are obtained:

Table 1.c.3.1 Results for 1.c.3.1

Parameter	Value
Test Loss for negative image	3.4309286254882814
Test Accuracy for negative image	0.2217
Training Loss for original image	0.006409686019861859
Training Accuracy for original image	0.9987333333333334

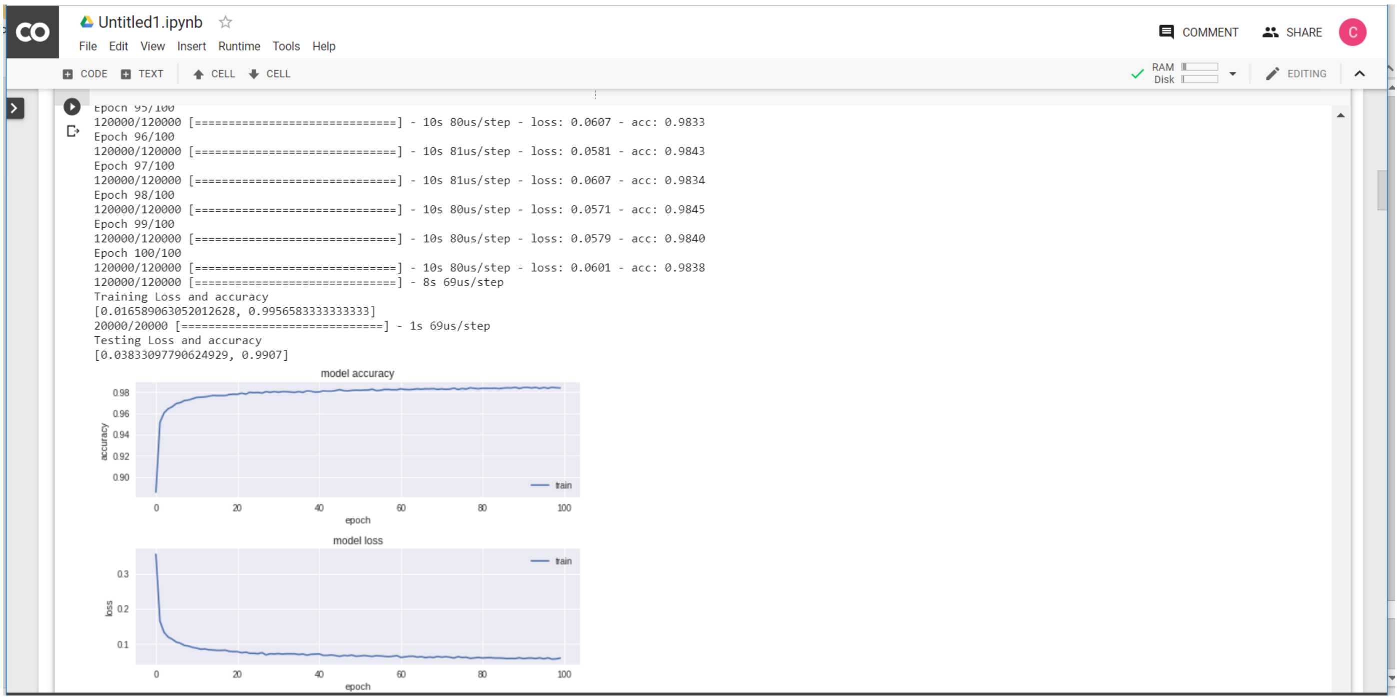
1.c.3.2

New Network:

By training convolutional neural network with original and negative training image and testing it on both original and negative test images and labels, the following results are obtained

Table 1.c.3.2 Results for 1.c.3.2

Parameter	**Value**
Test Loss for both original and negative image test images	0.03833097790624929
Test Accuracy for both original and negative image test images	0.9907
Training Loss for both original and negative image training images	0.016589063052012628
Training Accuracy for both original and negative image training images	0.9956583333333333



1.c.4 Discussion

From table 1.c.3.1, it can be inferred that testing accuracy is poor (or testing loss is high) in contrast to the training accuracy and loss.

Following are the reason:

- The model is trained for original (positive) image and the image is tested for negative image
- Even if the image is exactly similar to the original image, the activation function influences this poor performance.
- ReLu (Rectified linear ReLu(x)= max(0,x)) is a non-linear activation function, which clips any negative correlation with respect to the filter to 0. Thus, not allowing the feature to propagate further for classification.

- This clipping is essential since convolutional neural network is a multi-layer network, the cascading effect is likely to interpret the correlations falsely and network fails to identify dissimilarities. Thus, negative correlation value at each layer is clipped to 0 to avoid error propagating to subsequent multi-layer (as explained using RECOS system: a positive response at the first layer followed by a negative filter weight at the second layer; and (2) a negative response at the first layer followed by a positive filter weight at the second layer. For this reason, it is essential to set a negative correlation value (i.e. the response) at each layer to zero (or almost zero) to avoid confusion in a multi-layer RECOS system).
- Thus, the negative images (even if they are similar/ negatively correlated with original (positive) image) it is considered as an entirely new set of images. Hence, the system fails to identify the handwriting of digits producing low accuracy.
- This depicts that LeNet 5 is sensitive to the representation of image (it differentiates original and negative image as two separate image)

From table 1.c.3.2, it can be inferred that testing accuracy has improved.

Method1: Train the Le-Net 5 network with both original and negative training images, label and test it on original and negative testing image, label without changing the network architecture

Following are the reason:

- The original and negative images have the same 'label' irrespective of their data
- Since the categorical label are the same for both original and negative image, the categorical loss function must be minimized.
- To minimize categorical entropy for original and negative image inclusive, the weights of the kernel and bias filters are forced to learn both images by adjusting/updating their values through back-propagation and chain rule.
- The hyper-parameters used for obtaining the accuracies are:
  Optimizer: Adam
  Activation function: ReLu
  Dropout: 0.7
  Learning rate: 0.001
  Kernel initializer: RandomUniform
  Bias initializer: RandomUniform
- Other data processing and data augmentation techniques can be used to improve the existing performance.