Report of IDML730 CNN Assignment

BACHELOR IN TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted By :
Sandeep Singh:: IIT2015014
Raghav Saboo :: IIT2015042
Jayesh Chaudhari :: IIT2015044

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

Introduction

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars.

We are going to implement a CNN on **MNIST Dataset** with **1024 samples in the training set** containing almost **100 samples of each digit** and **256 samples in the testing set containing almost 25 samples of each digit.**

The model on which parameter tweaking was done was obtained from www.keras.io
The model was based on VGG. It is a VGG like ConvNet.

We will now tweak parameters such as -

1. Dropouts
2. Strides
3. Filter size
4. Activation function.
5. Optimization algorithm
6. Pooling layer size
7. DropConnect

And thus observe the effect of changing each of them in training and testing accuracy.

Architecture

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 329,962
Trainable params: 329,962
Non-trainable params: 0
_________________________________________________________________
None
```

Thus we have 4 convolution layers , 2 pooling layers , 3 fully connected layers. Stride is 1 for all the filters and zero padding.

Experiments

1 Dropout

Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units in a neural network.

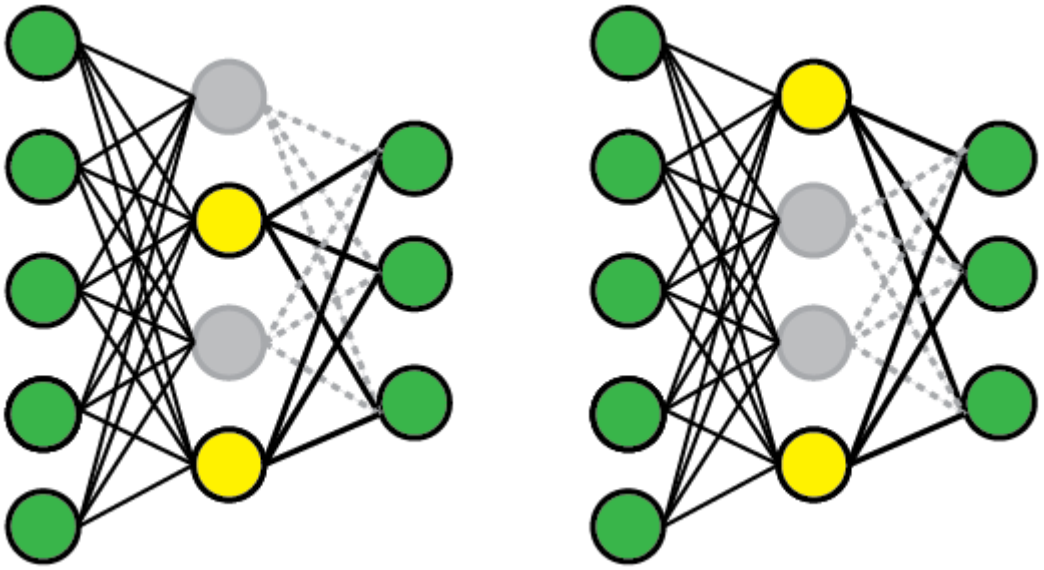
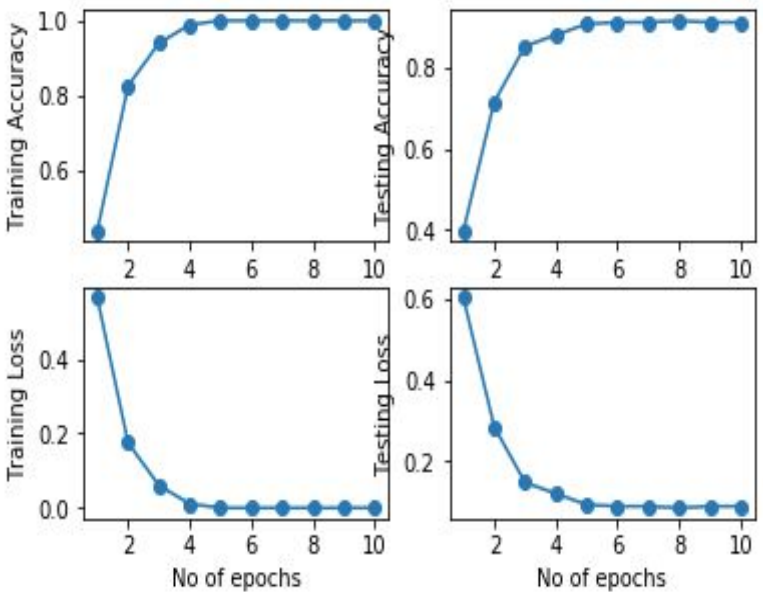
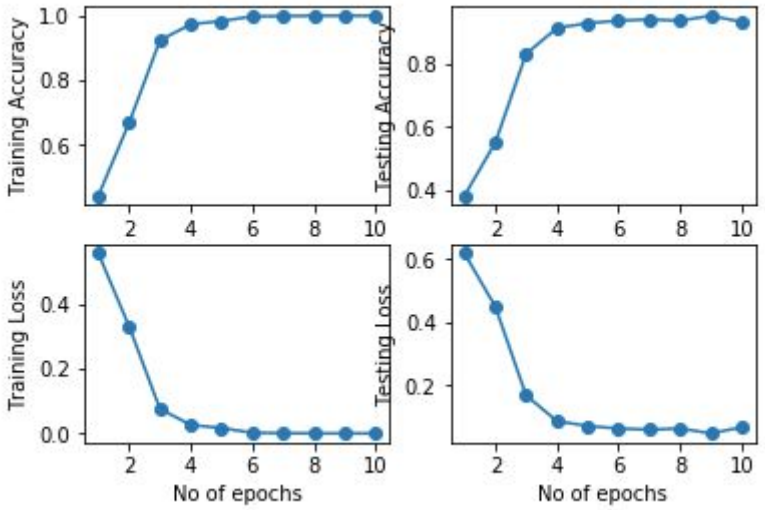


Fig 1. Visualization of Dropout

This is the graph of testing and training accuracy without using dropout.



After adding dropout in 3 layers with keep prob as 0.25 , 0.25 and 0.5 the results obtained are -



We have plotted the graph of testing and training accuracy with dropout for 10 epochs and we observed that with dropout overfitting was reduced and testing accuracy **improved by 2.2%**. Training accuracy was initially less in dropout but after a few epochs it increased gradually.

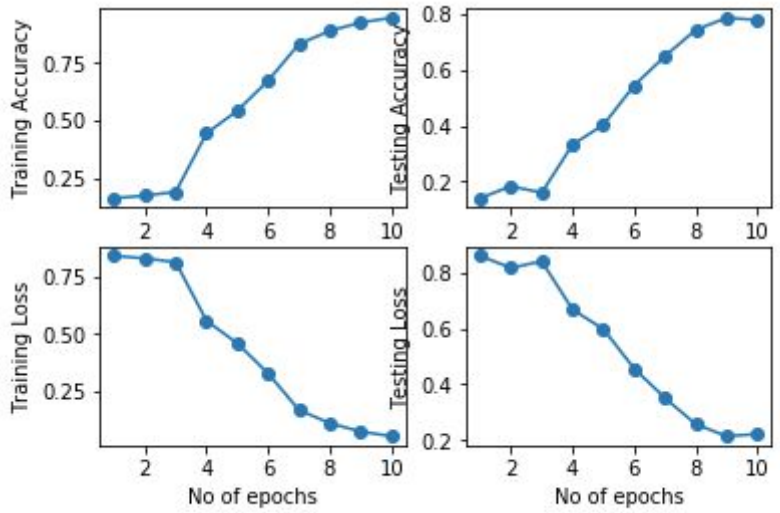
The testing accuracy was 91% without dropout and after using dropout it increased to 93%.

2 Strides

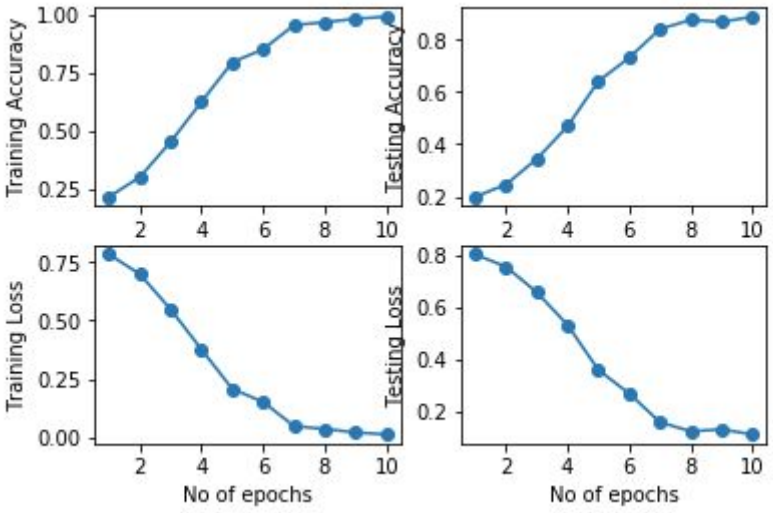
Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

We changed the strides of the first filter used from 2 to 4 and the results obtained were as follows-

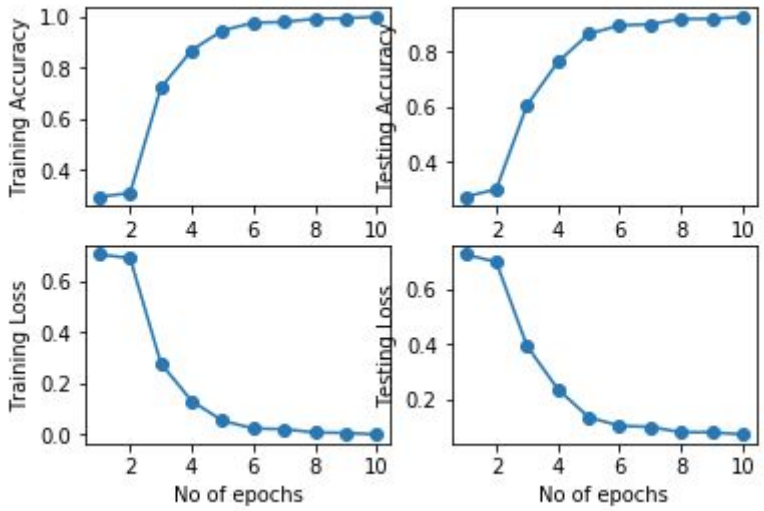
For Stride 4 -
Testing Accuracy - 77.73%
Training Accuracy - 94.62%



For stride 3 -
Testing Accuracy - 88.67%
Training Accuracy - 98.04%



For stride 2 -
Testing Accuracy - 92.57%
Training Accuracy - 99.41%



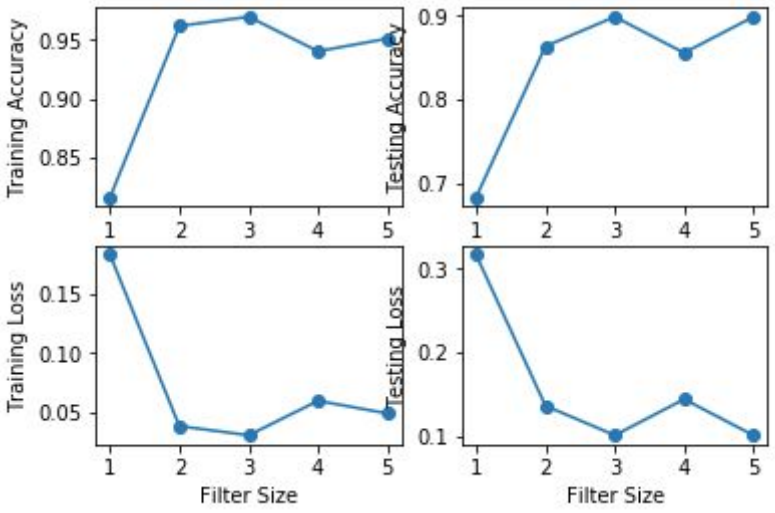
The testing accuracy gradually decreased as we increased the stride size as the input image was of size 28x28 and by increasing the stride size the shape of input for further layers decreased drastically. Thus stride size should be decided according to the shape of input.

3 Filter Size

A filter slides over the input image (convolution operation) to produce a feature map. In practice, a CNN learns the values of these filters on its own during the training process (although we still need to specify parameters such as number of filters, filter size, architecture of the network etc. before the training process). The more number of filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images.

We changed the size of filter from 1 to 5 and the results obtained were as follows -

Filter Size	1X1	2X2	3X3	4X4	5X5
Training Accuracy	81.64%	96.19%	96.97%	94.04%	95.11%
Testing Accuracy	68.35%	86.32%	89.84%	85.54%	89.84%



We got the best accuracy with filters of size 3x3 and least accuracy with filters of size 1x1.

4 Activation functions

Activation functions are really important for a Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable.They introduce non-linear properties to our Network.Their main purpose is to convert a input signal of a node in a A-NN to an output signal. That output signal now is used as a input in the next layer in the stack.

We have used 3 activation functions -

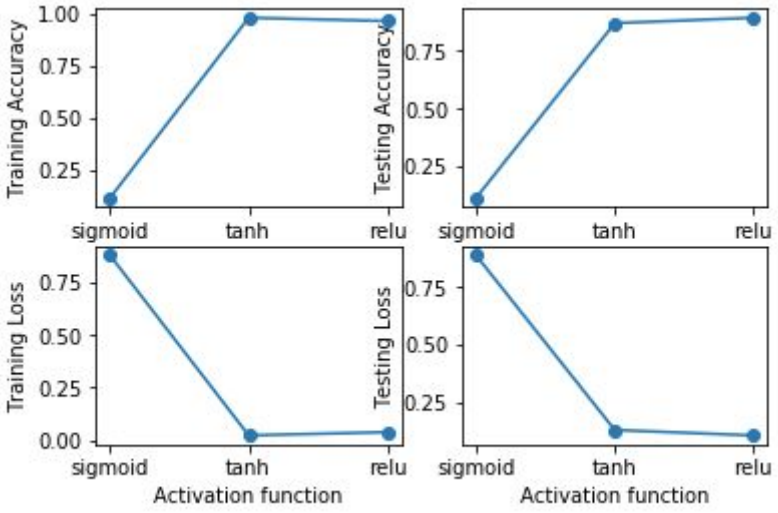
1. Sigmoid Activation function: It is an activation function of form f(x) = 1 / 1 + exp(-x) .Its Range is between 0 and 1. It has vanishing gradient problem , secondly , its output isn't zero centered. It makes the gradient updates go too far in different directions. 0 < output < 1 , and it makes optimization harder. Sigmoids saturate and kill gradients and Sigmoids have slow convergence.

2. Hyperbolic Tangent function- Tanh : It's mathematical formula is f(x) = 1 — exp(-2x) / 1 + exp(-2x). Now it's output is zero centered because its range in between -1 to 1 i.e -1 < output < 1 . Hence optimization is easier in this method hence in practice it is always preferred over Sigmoid function . But still it suffers from Vanishing gradient problem.

3. ReLu- Rectified Linear units : It has become very popular in the past couple of years. It was recently proved that it had 6 times improvement in convergence from Tanh function. It's just R(x) = max(0,x) i.e if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x.

The results obtained were as follows -

Activation Function	Sigmoid	TanH	ReLU
Training Accuracy	11.91%	97.75%	96.19%
Testing Accuracy	11.32%	87.10%	89.45%



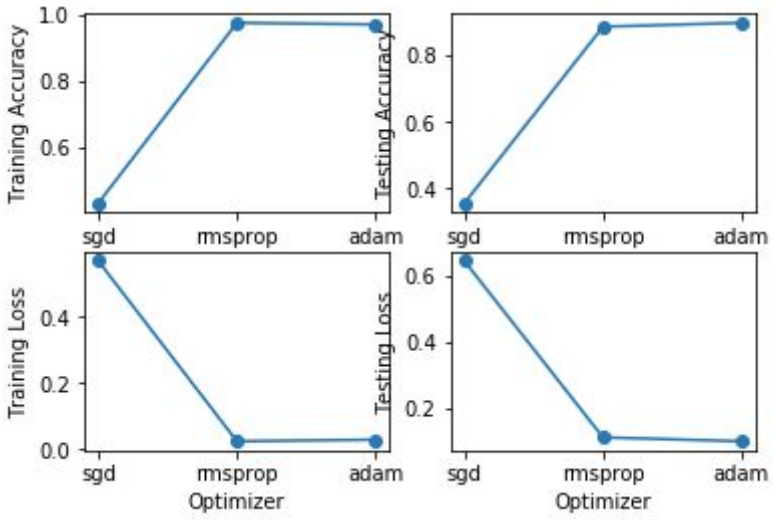
Thus tanh and relu are far better than sigmoid to be used as an activation function in neural networks. Relu is the best among the three and using it also makes the network converge faster.

5 Optimization Algorithm

Optimization algorithms helps us to minimize (or maximize) an Objective function (another name for Error function) E(x) which is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model. For example — we call the Weights(W) and the Bias(b) values of the neural network as its internal learnable parameters which are used in computing the output values and are learned and updated in the direction of optimal solution i.e minimizing the Loss by the network's training process and also play a major role in the training process of the Neural Network Model .

The results obtained were as follows -

Optimizers	SGD	RMSProp	Adam
Training Accuracy	43.06%	88.67%	89.84%
Testing Accuracy	35.54%	88.67%	89.84%



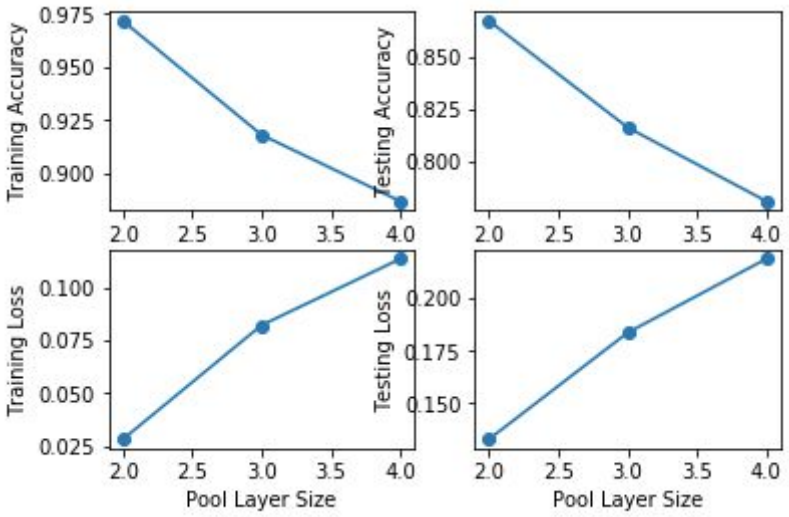
6 Pooling layer size

Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

We changed the size of pooling layer from 2 to 4 and we observed that on increasing the pooling layer size the shape of the input for further layers decreased drastically and thus the accuracy decreased.

The results obtained were as follows -

Pool Layer Size	**2**	**3**	**4**
Training Accuracy	97.16%	91.79%	88.67%
Testing Accuracy	86.71%	81.64%	78.12%



7 DropConnect

For Regularizing large fully connected layers within neural networls and preventing overfitting, we use DropConnect. DropConnect is a generalization of Dropout. When training with Dropout, a randomly selected subset of activations are set to zero within each layer. DropConnect instead sets a randomly selected subset of weights within the network to zero.

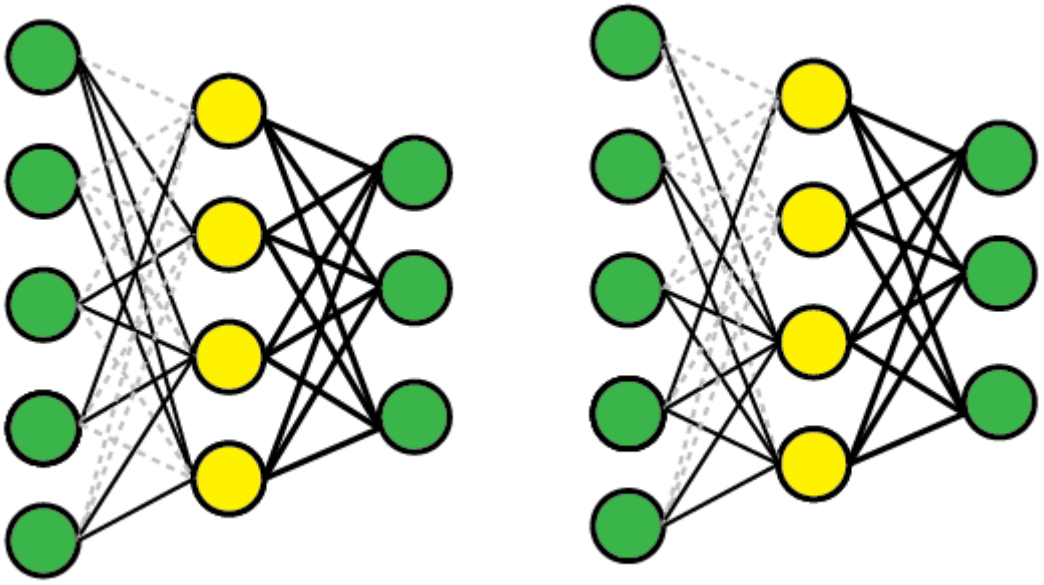
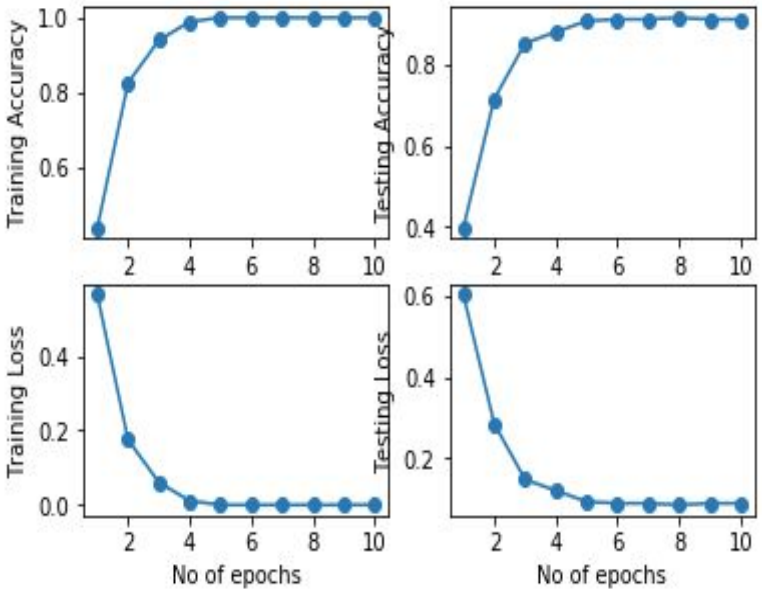
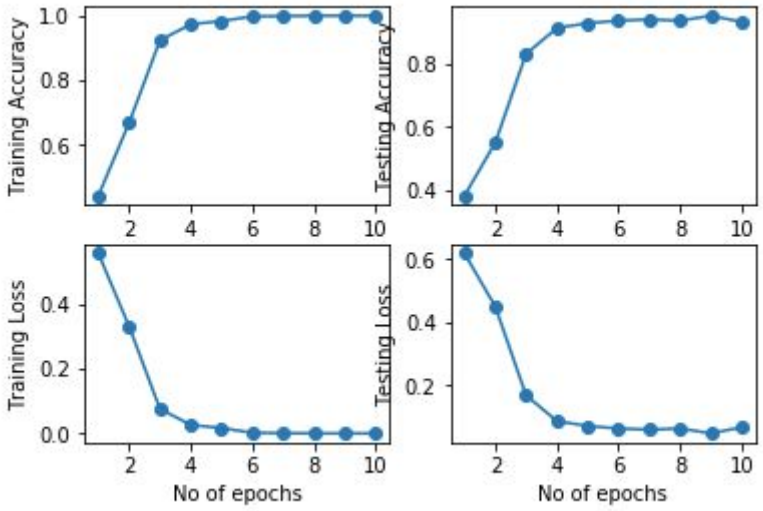


Fig 2. Visualization of DropConnect

This is the graph of testing and training accuracy without using dropout.



After adding dropout in 3 layers with prob as 0.35 , 0.35 and 0.5 the results obtained are -



We have plotted the graph of testing and training accuracy with drop connect for 10 epochs and we observed that with drop connect overfitting was reduced and testing accuracy **improved by 3.5%**. Training accuracy was initially less in drop connect but after a few epochs it increased gradually.

The testing accuracy was 91% without drop connect and after using drop connect it increased to 94.5%.