Indian Institute of Technology Ropar

CSL-603

Self-Driving Cars and Neural Network

Submitted To:
Narayanan C Krishnan
Computer Science
Department

Submitted By :
Siddharth Nahar
2016CSB1043
Group-G1
5th Semester

Contents

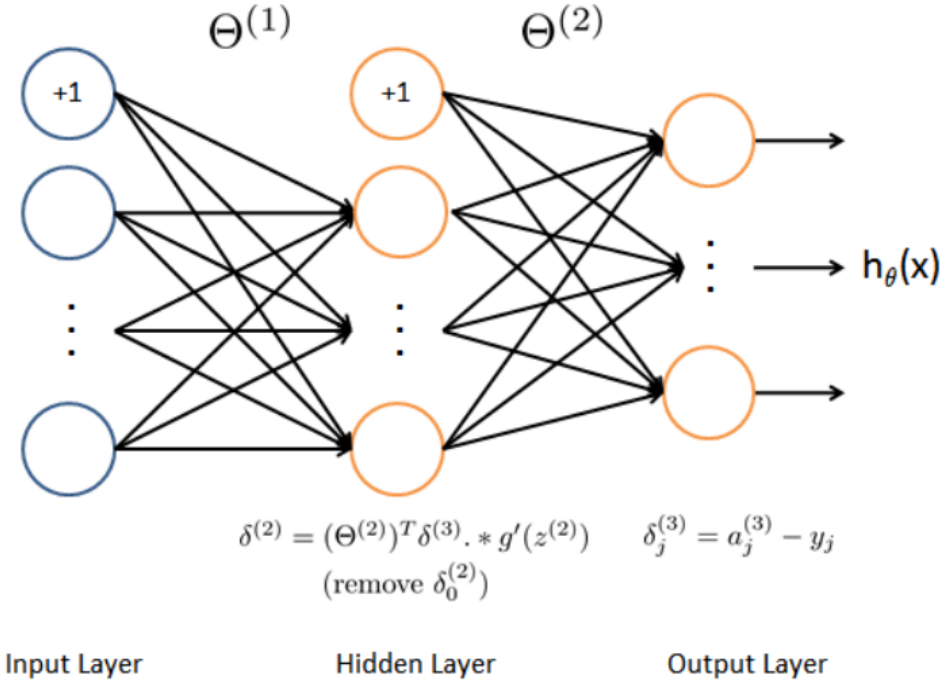
1 Neural Network Model and Basic Analysis

This is Report for Neural Networks Model for predicting Steering angle looking at Road Images.Sample Images are taken from https://github.com/udacity/CarND-Behavioral-Cloning-P3

From the Training Examples I have samples 20% as Test Set and Remaining is partitoned as necessary in Training and Validation Set.Training set is saved in steering folder.All Results and Analysis is mentioned below.

1.1 Basic Analysis of Multilayer Neural Networks

Model Representation :
X = $a^{(1)}$ = (N,1025) , $a^{(2)}$ = (N,513) , $a^{(3)}$ = (N,65) , o = (N,1)
$\Theta^{(1)}$ = (512,1025) , $\Theta^{(2)}$ = ((64,513) , $\Theta^{(3)}$ = (1,65)



1.1.1 Forward Pass :

$a^{(l)}$ represents Activation layer Output,Θ represents Weights between layer.

Stable Sigmoid :-

$$\sigma(z) = \begin{cases} 1/(1+\exp(-z)), & \text{if } z \geq 0. \\ \exp(z)/(1+\exp(z)), & \text{if } z < 0. \end{cases} \tag{1}$$

$$a^{(1)} = X + bias \tag{2}$$

$$z^{(l)} = a^{(l-1)}\Theta^{(l)T} \quad .....\forall l > 1 \tag{3}$$

$$a^{(l)} = \sigma(z^{(l)}) + bias \quad .....\forall l > 1 \tag{4}$$

$$o = a^{(last)}\Theta^{(last)T} \tag{5}$$

1.1.2 Backward Pass :

Cost Function :-

$$J = \sum_{i=1}^{i=n} (o-y)^2/2 * n \tag{6}$$

Sigmoid Gradient :-

$$\sigma^{'}(z) = \sigma(z) * (1 - \sigma(z)) \tag{7}$$

$$\delta^{(4)} = o - y \tag{8}$$

$$\delta^{(l)} = \delta^{(l+1)}\Theta^{(l)} .* \sigma^{`}(z^{(l)}) \tag{9}$$

$$\bigtriangledown\Theta^{(l)} = \alpha * \delta^{(l+1)}a^{(l)} \tag{10}$$

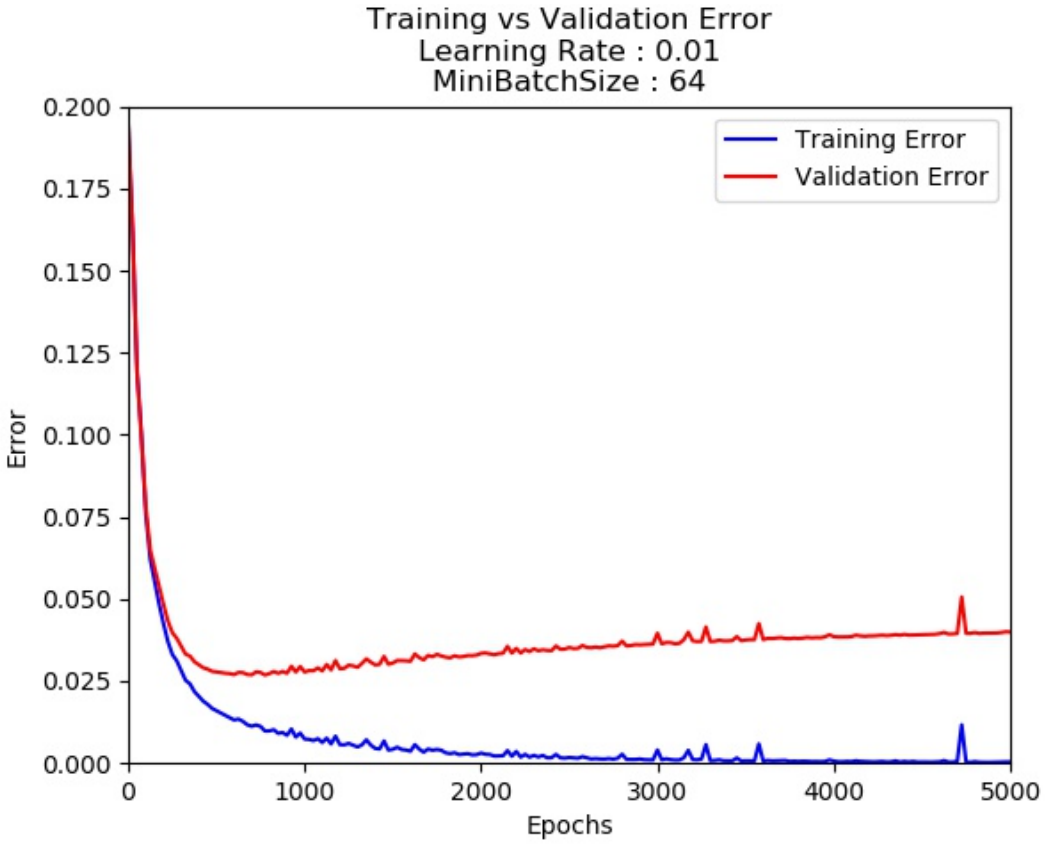
1.2 Observations and Graphs :

All Observations are done using MiniBatch Gradient Descent.MiniBatch Gradient Descent is midieval way for Gradient descent and Stochastic Gradient Descent.

- It is more faster and efficient than gradient Descent on total Training Set.
- It provides faster convergence than Gradient Descent.

1.2.1 Graph of Training and Validation Error vs Iterations :

Graph of Q1 : -



Parameters :

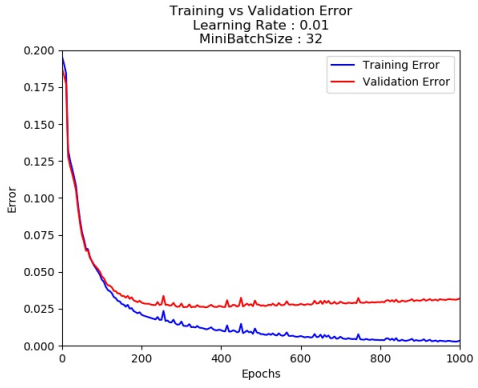
- Learning Rate = 0.01
- Epochs = 5000
- Minibatch Size = 64
- Dropout = 0

Observations :-

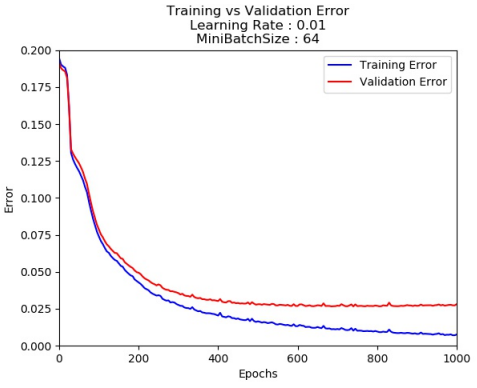
- Training Error :- 0.000202 , Validation Error :- 0.0456
- Training and Validation both eventually decreases on number of iterations.
- Gradient descent is slow convergence.Effective decrease is observed after 100 iterations.
- Alpha is quite high So there are many oscillations for converging to minimum.
- Validation Error increases for higher iterations as Model tries to Overfitt the model.

1.2.2 Graphs and Observations with different Batch Sizes :-

Graphs for Q2 :-



(a) Batch Size = 32



(b) Batch Size = 64

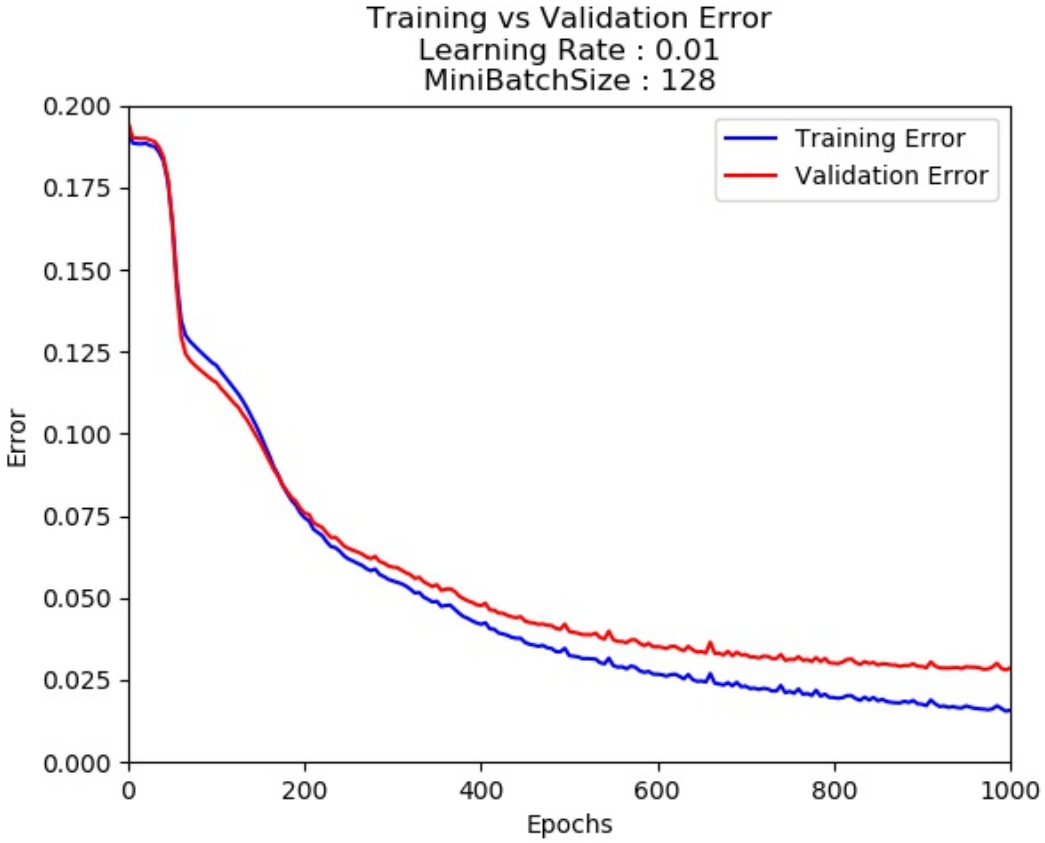


Figure 2: Batch Size = 128

Parameters :

- Learning Rate = 0.01
- Epochs = 1000
- Batch Size = 32,64,128
- DropOut = 0

Observations :

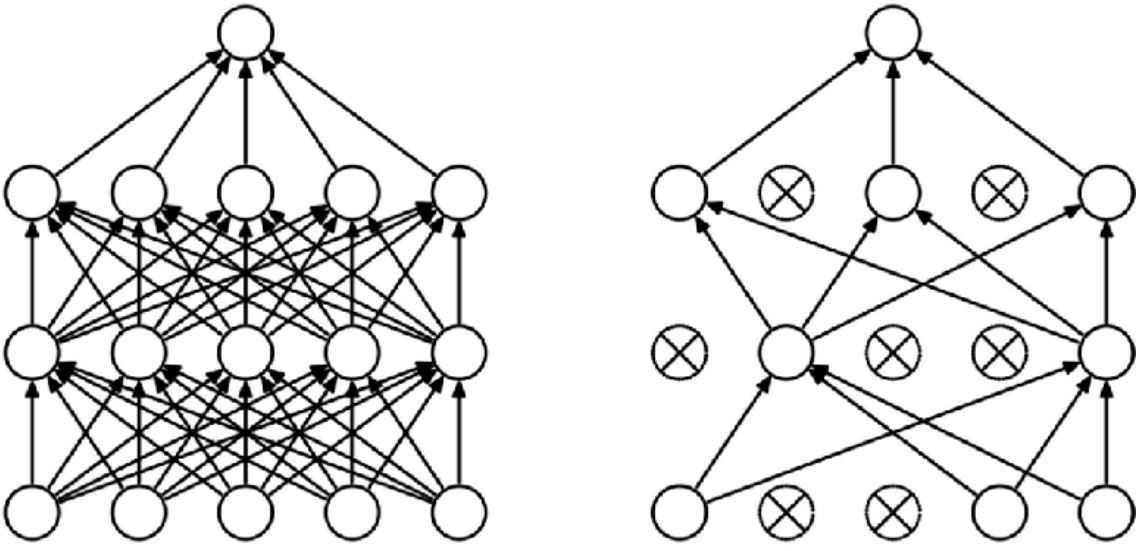
- – Batch Size = 32, Train Error = 0.00124, Validation error = 0.035
  – Batch Size = 64, Train Error = 0.00452, Validation Error =

0.0301

- Batch Size = 128, Train Error = 0.00921, Validation Error = 0.032

- We can See Less Batch Size tends towards Fast Convergence i.e Stochastic Gradient Descent. But doesn't fits Test model.So there is Trade Off of Batch Size vs Convergence Rate vs Accuracy.
- We can See Difference between Training Accuracy and Validation Accuracy decreases with increasing Batch Size.This shows the nature of Gradient Descent and Stochastic Gradient Descent.
- Oscillations are less for high Batch Size gradient.As effective learning rate decreases for high batch size.

1.2.3 DropOuts :-

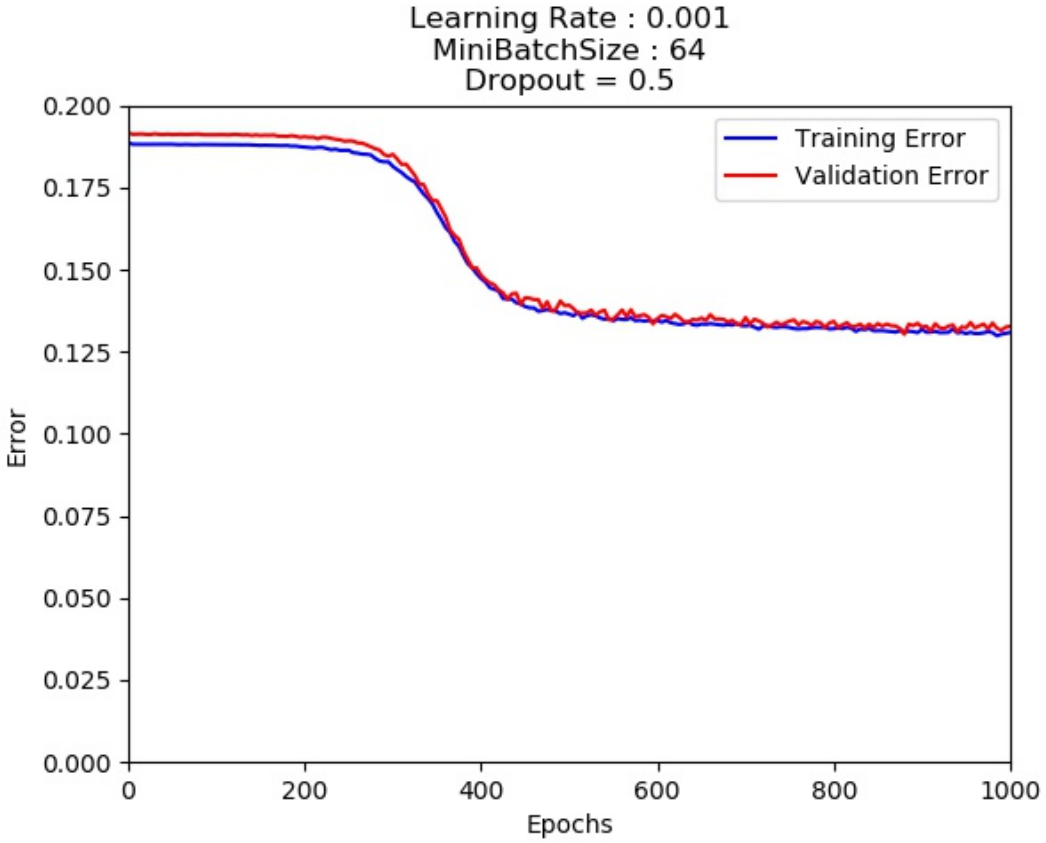


(a) Standard Neural Net (b) After applying dropout

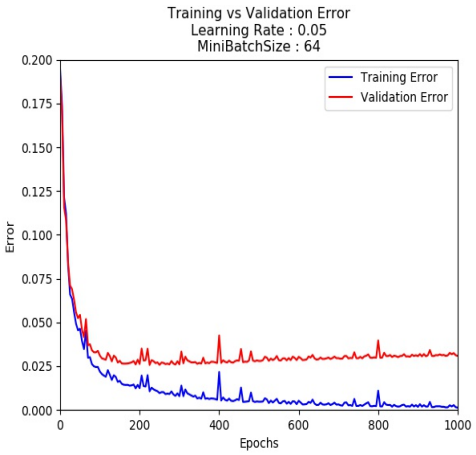
DropOut act as Regularization for Neural Networks.It avoids Overfitting of our Model.It randomly selects nodes in each layer to drop based on percentage and make them 0 as they will not contribute in training. As shown above :

Observations :-

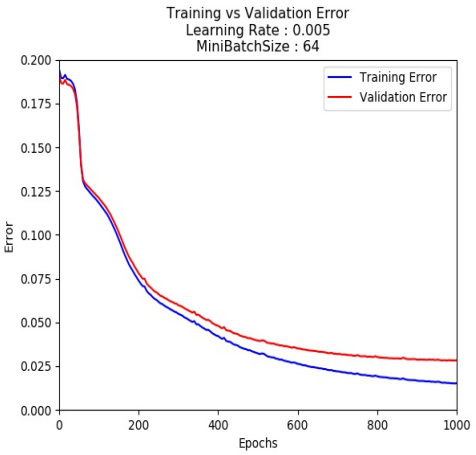
- For DropOut 0.5 , Training Error : 0.13, Validation Error : 0.12998
- It Regularizes the Training , So Training Error increases as compared to Error without DropOuts.
- Difference between Training and Test Error is minimum as it prevents overfitting of data.



1.2.4 Graphs with different Learning Rates :-



(a) Learning Rate : 0.05



(b) Learning Rate : 0.005

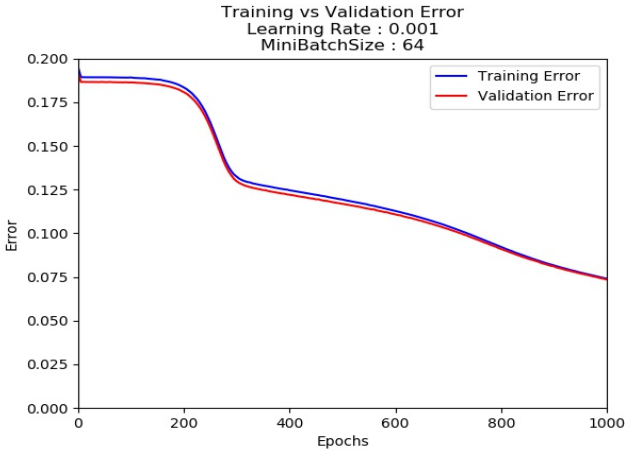


Figure 4: Learning Rate : 0.001

Observations :-

- High Learning Rate may fast converge but as iterations increases it starts Oscillating to attain minimum.
- Difference between Validation Error and Error decreases with decreasing learning Rate.As Oscillations about minimum decreases.

1.3 Different Activation Functions :-

1.3.1 Tanh :-

Function :-

$$g(x) = \tanh(x) \tag{11}$$

$$g^{'}(x) = 1 - \tanh^2(x) \tag{12}$$

Observation :-

- Tanh is more fast and efficient than sigmoid.
- It is more preffered due to it's symmetricity about 0.
- Graph has been Plotted using tanh as activation function

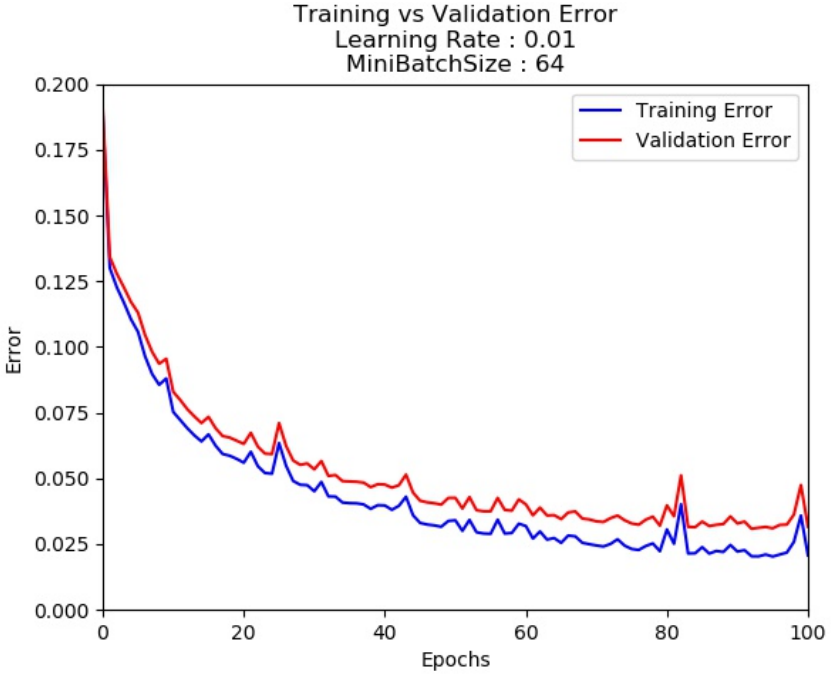


Figure 5: Tanh function

1.3.2 ReLU :-

Function :-

$$g(x) = \begin{cases} x & \text{if x} \geq 0. \\ 0 & \text{if x} < 0. \end{cases} \tag{13}$$

$$g^{'}(x) = \begin{cases} 1 & \text{if x} \geq 0. \\ 0 & \text{x} < 0 \end{cases} \tag{14}$$

Observation :-

- Reduce liklihood of gradient to vanish.
- Constant gradient increases speed of convergence.
- Sparsity of Gradient is again of importance

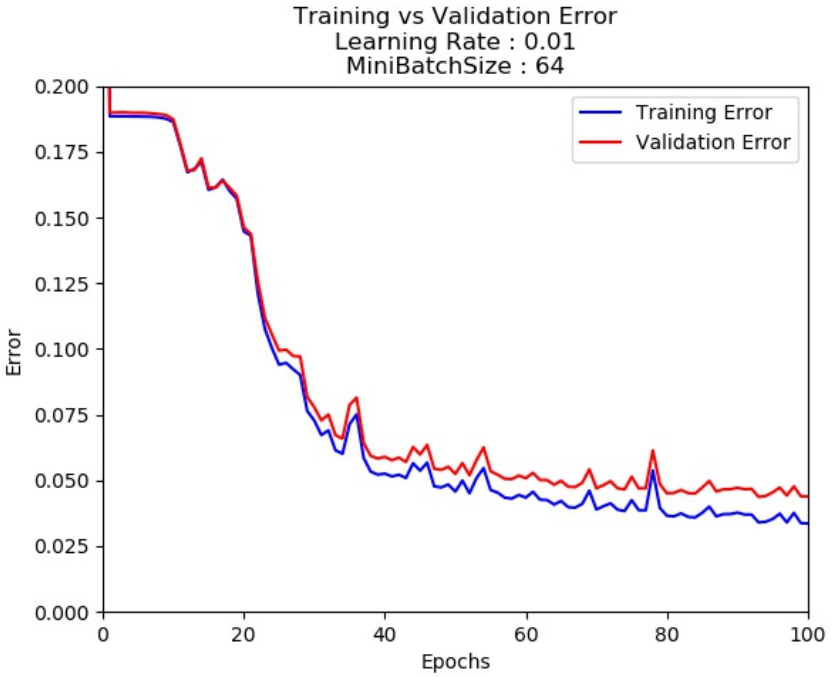


Figure 6: ReLU Activation Function

1.4 Different Optimizers :

1.4.1 Adam's Optimizer :-

Learning Rates are most concern for Model to Converge . If Learning rate is not proper it may lead to divergence or oscillations about minimum. We also have no control to change learning rates in between training of model.
Adam's Optimizer vanishes this property and adjust learning rate while training. So it provides fast convergence than Gradient Descent.

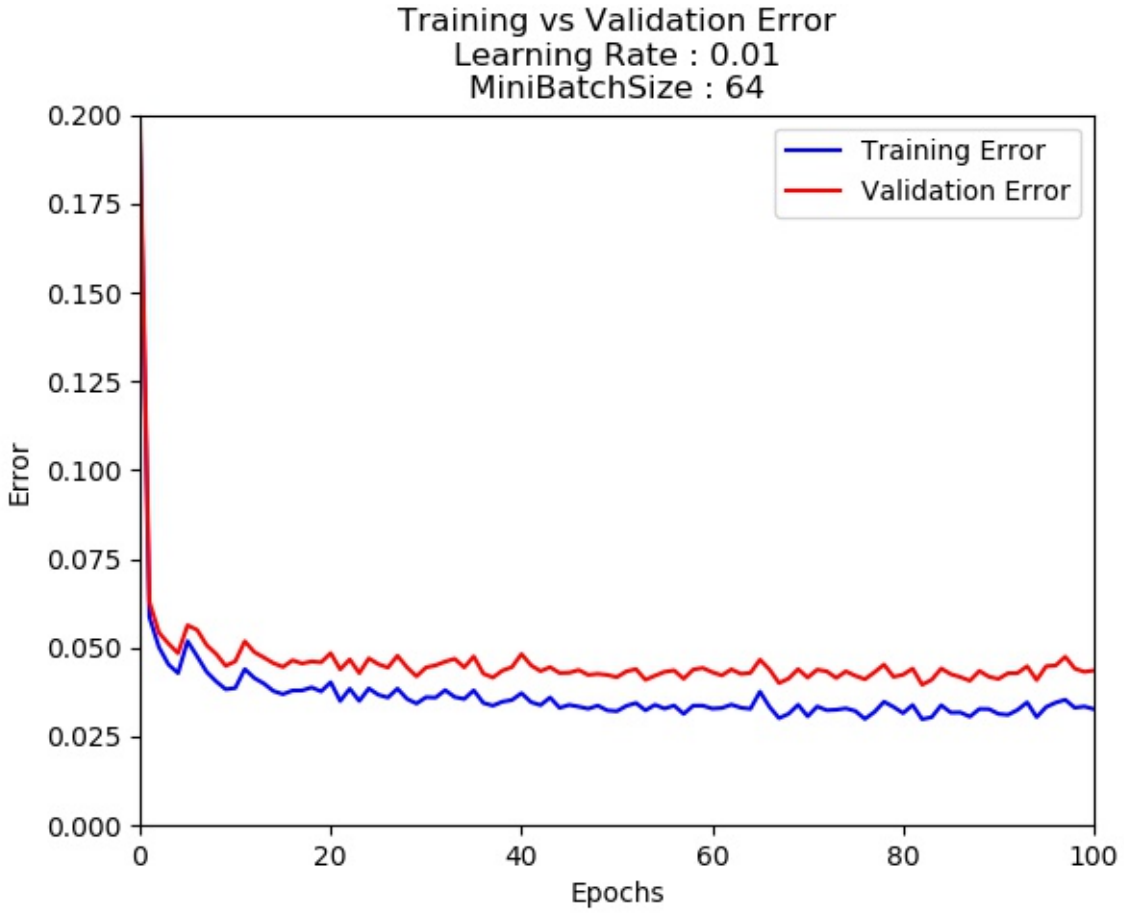


Figure 7: Adam's Optimizer