Assignment - 3 Report
Section B

Neural Nets

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously.

Simple neural network architecture

A basic neural network has interconnected artificial neurons in three layers:

Input Layer

Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.

Hidden Layer

Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.

Output Layer

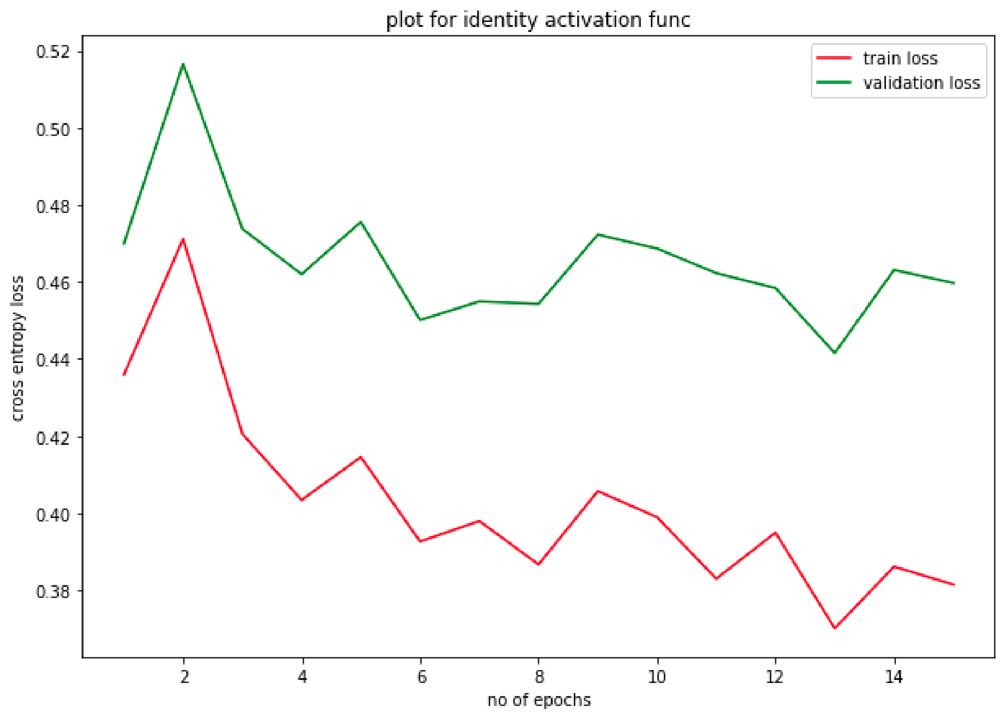
The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

The input layer in our case has 28*28 = 784 neurons and the output layer has 10 neurons.

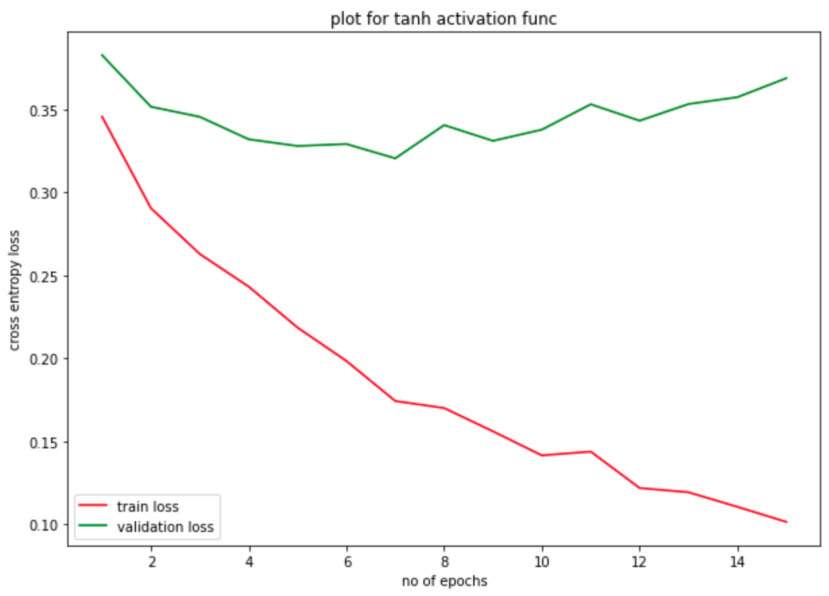
The hidden layer used in our model is [256, 128, 64, 32] and there is a variety of activation functions that can be used such as sigmoid, tanh, ReLU, Leaky ReLU, linear and softmax (it is used at the output layer because)

Training loss & validation loss vs epochs curve

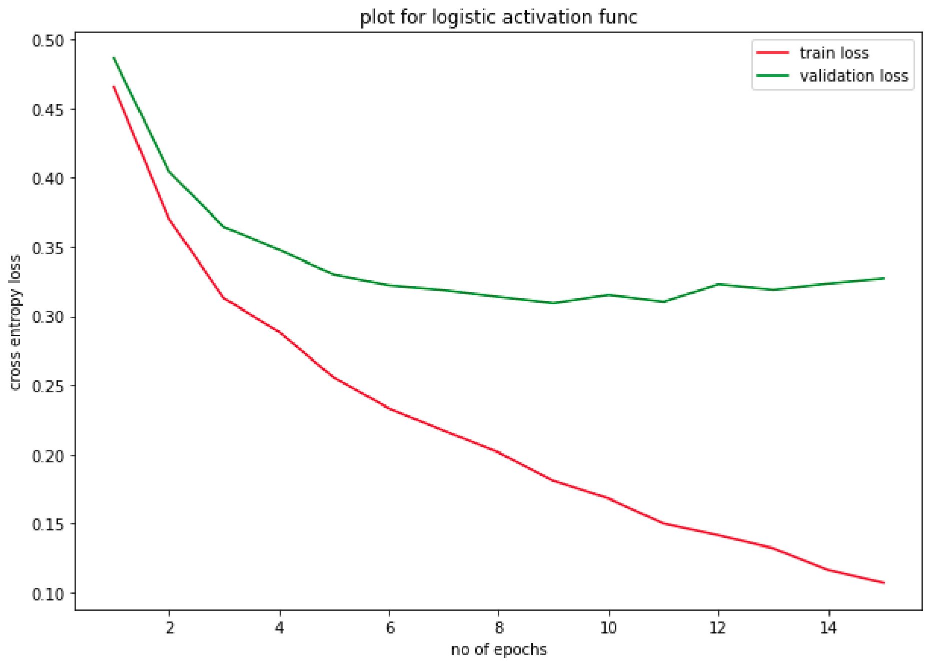
For linear



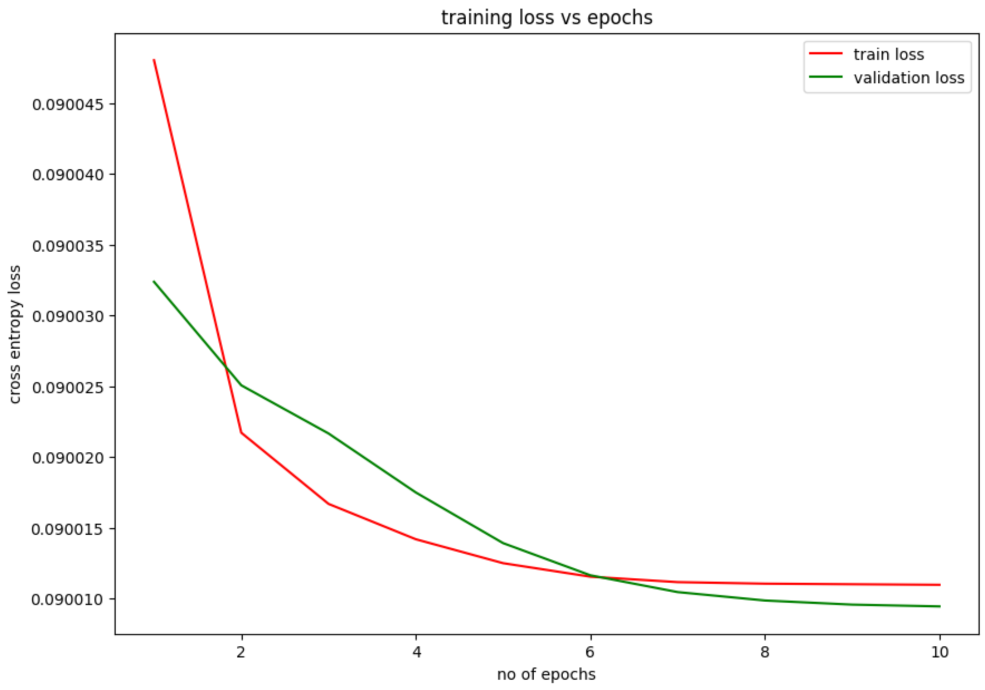
For tanh



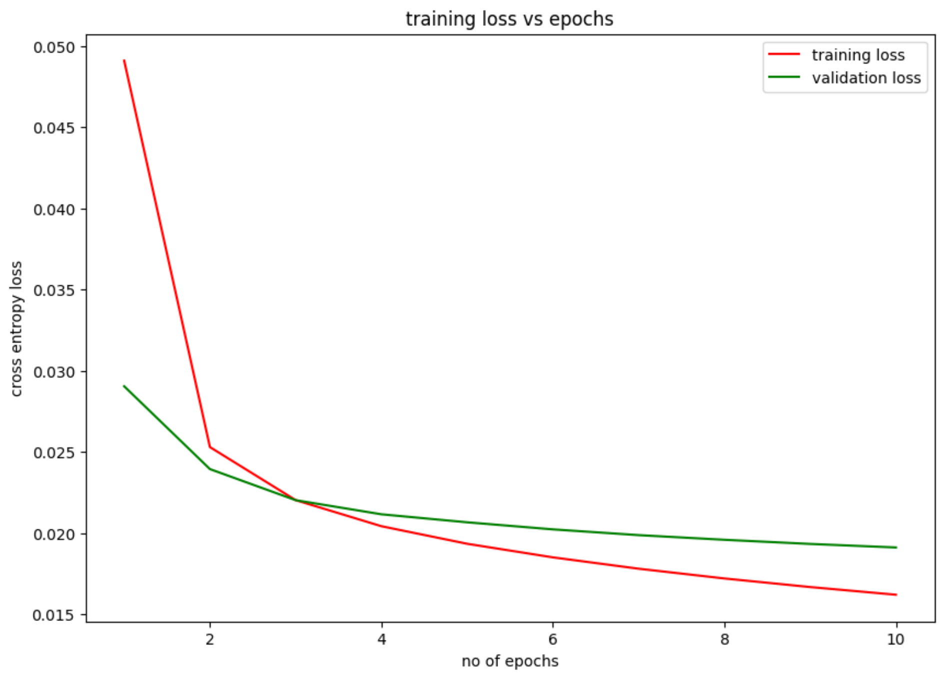
For sigmoid



for relu



for leaky relu



The weight initialization function initialises the weights at each hidden layer. I have used 3 main initialization techniques, zero initialization (assign value 0 to all the weights), random initialization (randomly assign values to all the weights), and Normal initialization N(0, 1) (assign values from)

I have used softmax at the end of the output layer to convert the output in the form of probabilities. the class that has the max probability is the

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

predicted value.

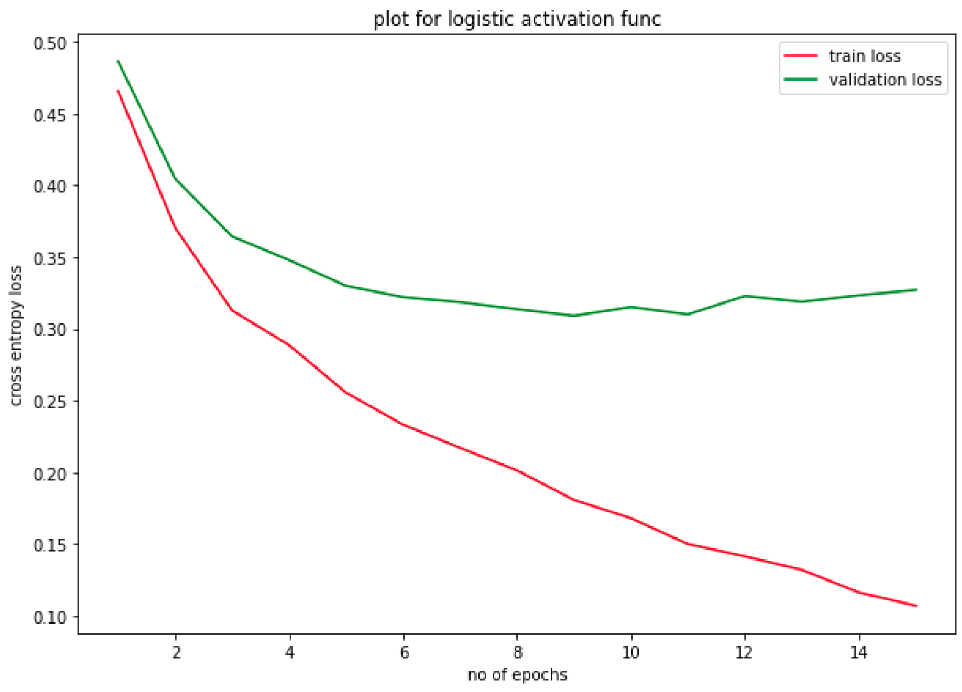
Section - C

a) Relu comes out to be the best activation function in our case, with better accuracy on the data.
The biggest advantage of ReLu is indeed the non-saturation of its gradient, which greatly accelerates the convergence of stochastic gradient descent compared to the sigmoid / tanh functions. Basically, it solves the vanishing gradient problem. Thus, learning is faster.
For hidden layers, tanh is better because it keeps the values in the range [-1, 1] and is centred around 0 and the derivative is between 0 and 1.

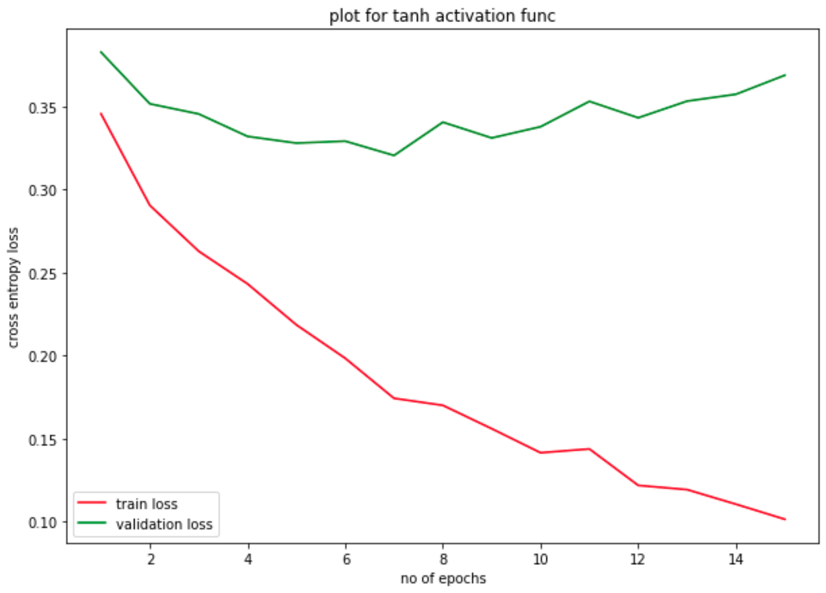
b) 0.001 comes out to be the better learning rate because the learning is slow, and hence it doesn't overshoot the global minima.
Hence, a smaller α (learning rate) results in a smaller step size and a better approximation of the true derivative, which in turn improves the ability to locate the optimal point.

c) When we decrease the number of neurons in the hidden layer this will result in something called underfitting. This happens when our model performs poorly on training data itself. If we decrease the number of neurons then our model will fail to capture the relationship between the different features and the target values.

d) After applying the grid search for the activation function `['logistic', 'relu', 'tanh', 'identity']` and learning `[0.1, 0.01, 0.001]` The best hyperparameters that performed well was 'relu' with a learning rate of '0.001'. The same result was obtained in parts a and b. and batch size was = 100



```
accuracy on test data ->  0.8848
avg loss on training set :  0.22270076413813877
avg loss on validation set :  0.34093655639061426
```



```
accuracy on test data ->  0.8814
avg loss on training set :  0.1865843944059124
avg loss on validation set :  0.34491555068420665
```