Exp-6 - Implement Gradient Descent & Backpropagation in Deep Neural Network

13/9/25

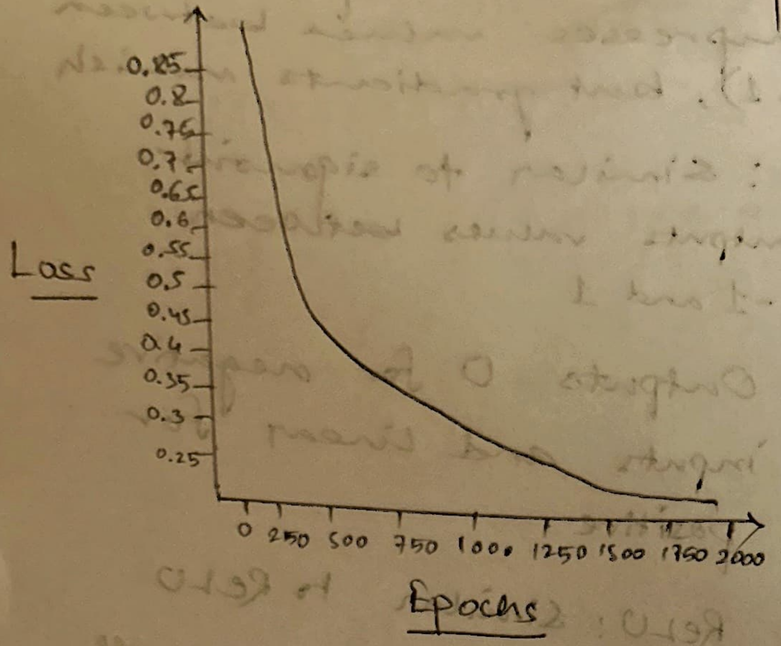
Aim: To implement a Deep Neural Network (DNN) from scratch using Numpy and train it on a toy dataset (make_moons) by applying gradient descent ~~ab~~ and backpropagation algorithms.

Objective:

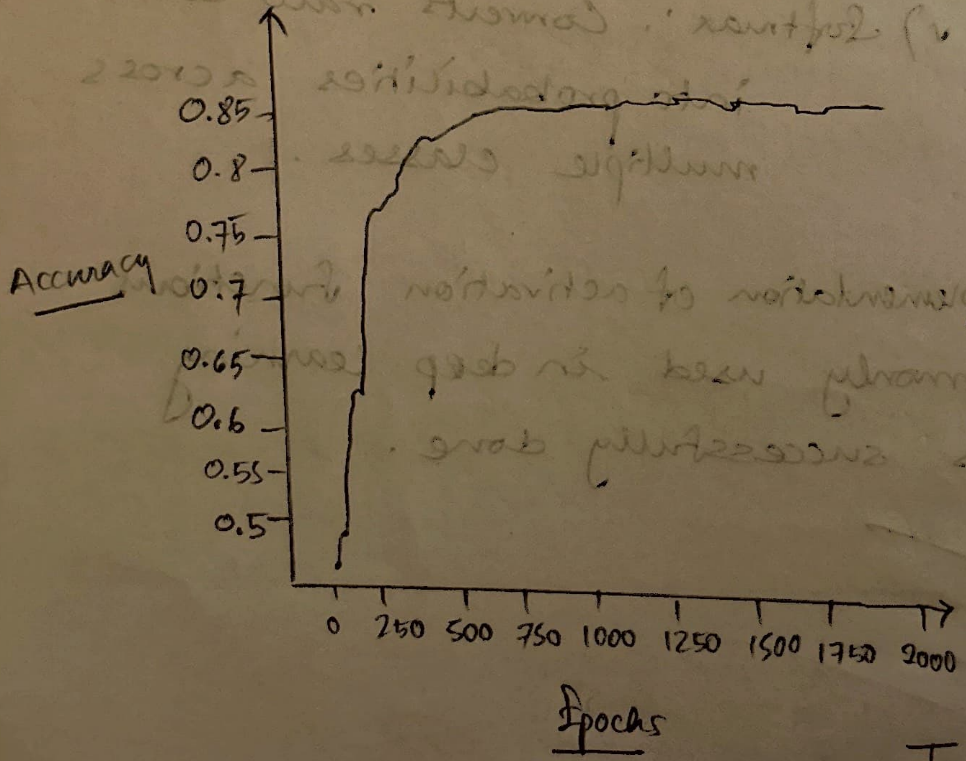
1) To understand the working of forward propagation and backpropagation in deep neural networks.
2) To implement gradient descent optimization for updating networking weights.
3) To train the DNN on a non-linear dataset (two moons) and evaluate classification accuracy.
4) To demonstrate that neural networks can learn non-linear decision boundaries without external libraries like Tensorflow or Pytorch.

Pseudocode:

```
1) Import required libraries (Numpy, sklearn).
2) Generate dataset using make_moons
   and split into training/testing sets.
3) Initialize network parameters:
   - Random weights with He/Xavier
     initialisation.
   - Zero biases
```



Training Loss Curve



Training Accuracy Curve

4) Define activation functions (ReLU and Sigmoid)

5) For each epoch:

   a. Forward Pass:
      - Computations done on activation layer by layer.

   b. Compute Loss:
      - Binary cross-entropy loss.

   c. Backward Pass:
      - Calculate gradients for each layer using chain rule.

   d. Update Weights:
      - Apply gradient descent with learn rate.

   e. Print loss and accuracy every few iterations.

6) After training, evaluate accuracy on test set.

7) Display final results.

Observations: Initially, the model performed close to random guessing (~50% accuracy). With training, loss decreased steadily and accuracy improved. The network successfully learned the non-linear decision boundary, reaching ~90% accuracy on the test set.

[illegible] 16/9/25

Result: The aforementioned experiment was successfully carried out.

Results.

	Epoch	Loss	Accuracy
1)	0	0.6623	0.4775
2)	200	0.4576	0.7950
3)	400	0.3620	0.8200
4)	600	0.3205	0.8375
5)	800	0.3004	0.8450
6)	1000	0.2878	0.8525
7)	1200	0.2789	0.8600
8)	1400	0.2721	0.8650
9)	1600	0.2665	0.8675
10)	1800	0.2616	0.8650

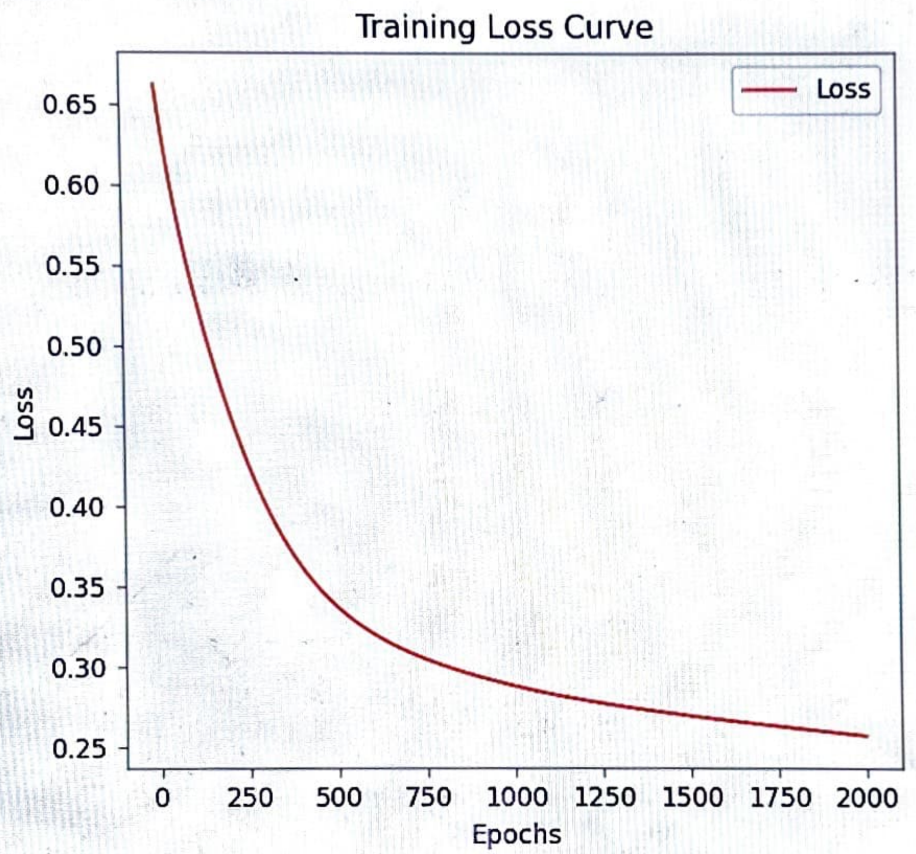
Final Test Accuracy → 0.8600

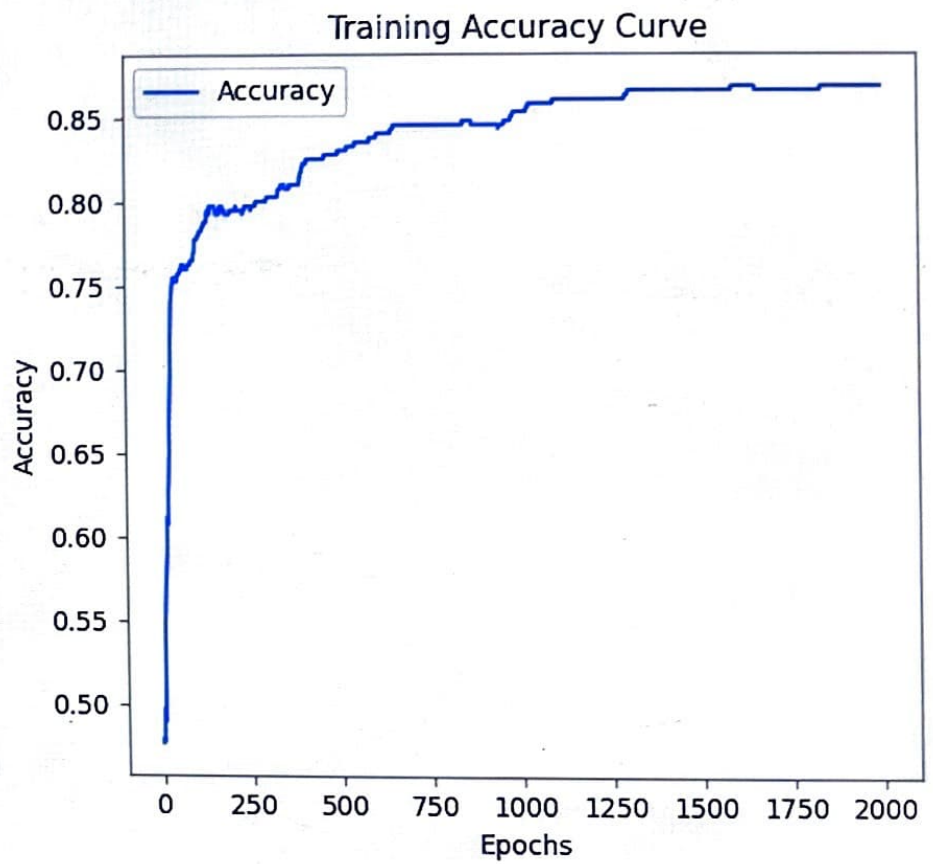
NAME: Aarush Talukdar Reg: RA2311047010014.

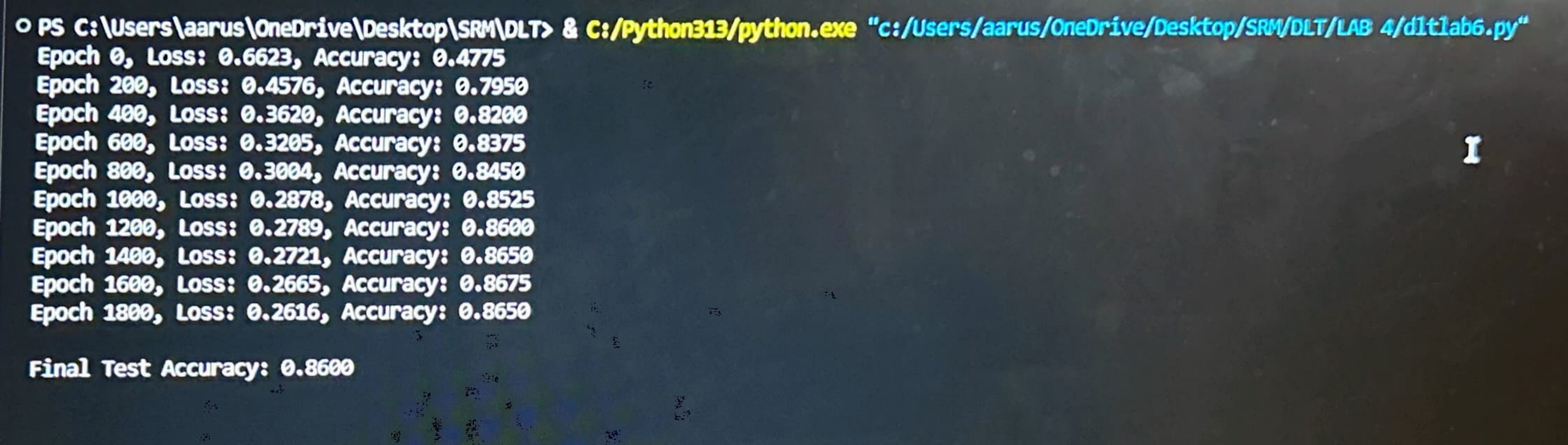
Serial. No.	Topic.	Date.	Signature.
1)	Exploring the Deep Learning Platforms & frameworks	31/07/2025.	[illegible]
2)	Implement a Classifier using an open-source dataset	7/8/2025	[illegible] 7/8/25
3)	Study of Classifiers with respect to Statistical Parameter	7/8/2025	
4)	Build a simple feed forward network to recognize handwritten character	14/8/2025	[illegible] 14/8/25
5)	Study of Activation Functions and its role	9/9/2025	[illegible] 9/9
6)	Implement gradient descent and backpropagation in deep neural network.	13/9/2025	[illegible]
7)	Build a CNN model to classify Cat & dog image	13/9/2025	[illegible]
8)	Experiment using LSTM	13/9/2025	[illegible]
9)	Build a Recurrent Neural Network	13/9/2025.	[illegible] 13/9/25
10)	Perform compression on MNIST	02/11/25	[illegible]
11)	Expeiment using VAE		
12)	Implement a DCGAN		
13)	Understand pre-trained model		
14)	Transfer Learning		
15	YOLO Model		

Completed [illegible]

Figure 1


Training Loss Curve
Loss
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0
250
500
750
1000
1250
1500
1750
2000
Epochs


Training Accuracy Curve
Accuracy
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
250
500
750
1000
1250
1500
1750
2000
Epochs



```
○ PS C:\Users\aarus\OneDrive\Desktop\SRM\DLT> & C:/Python313/python.exe "c:/Users/aarus/OneDrive/Desktop/SRM/DLT/LAB 4/dltlab6.py"
  Epoch 0, Loss: 0.6623, Accuracy: 0.4775
  Epoch 200, Loss: 0.4576, Accuracy: 0.7950
  Epoch 400, Loss: 0.3620, Accuracy: 0.8200
  Epoch 600, Loss: 0.3205, Accuracy: 0.8375
  Epoch 800, Loss: 0.3004, Accuracy: 0.8450
  Epoch 1000, Loss: 0.2878, Accuracy: 0.8525
  Epoch 1200, Loss: 0.2789, Accuracy: 0.8600
  Epoch 1400, Loss: 0.2721, Accuracy: 0.8650
  Epoch 1600, Loss: 0.2665, Accuracy: 0.8675
  Epoch 1800, Loss: 0.2616, Accuracy: 0.8650

  Final Test Accuracy: 0.8600
```