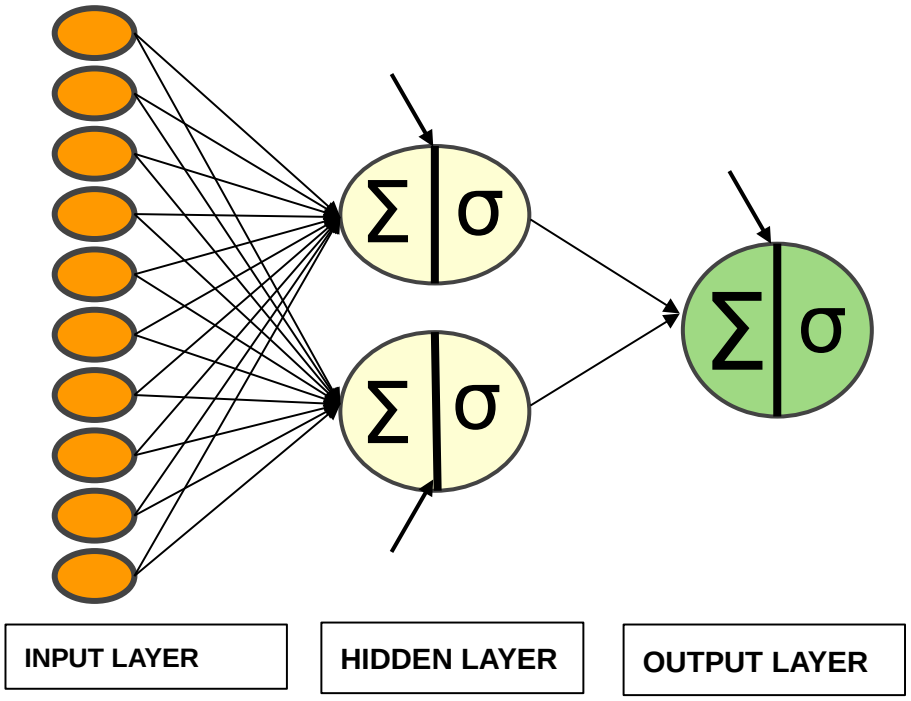
Implementation of Backpropagation and Training a Palindrome Network

Aditya Pande	22M2108

Problem Statement

- **Input**: 10-bit String (of numbers)
- **Output**: 1 if Palindrome, 0 otherwise

Architecture and Hyperparameter details



INPUT LAYER

No of neurons : 10
Input layer is **FULLY CONNECTED** to next layer

HIDDEN LAYER

No of neurons : 2
Activation : Sigmoid σ

HYPERPARAMETER

- Learning Rate: 0.1
- Batch Size: 32
- Momentum **β:** 0.9
- OverSampling: duplicate 20 times Minor class

OUTPUT LAYER

No of neurons : 1
Threshold : 0.5 for predicting 1
Activation : Sigmoid σ

BIAS IS USED IN HIDDEN AS WELL AS OUTPUT LAYER

Overall performance

Accuracy (Total and class-wise)

- Total = 100%
- Class0 Non- Palindrome (NP)= 100%
- Class1 Palindrome(P) = 100%

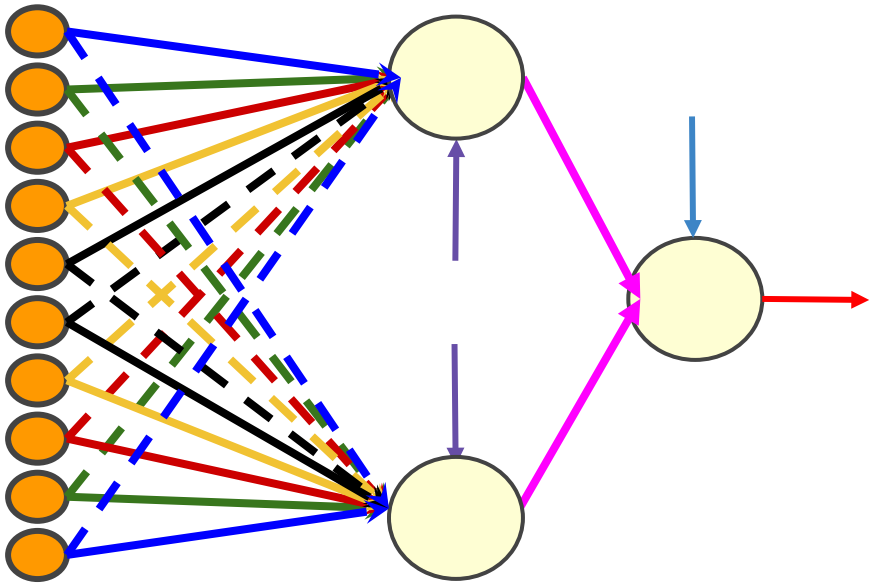
Classification Report

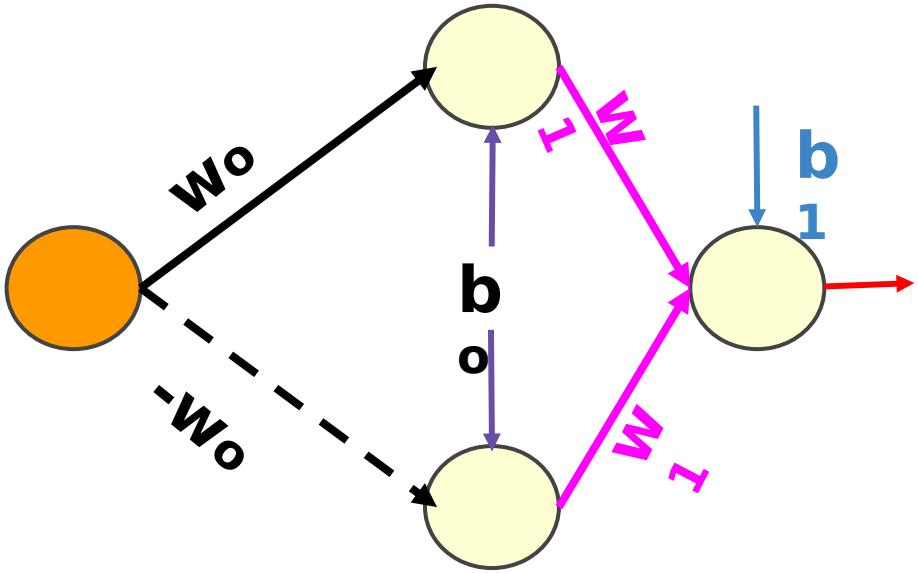
		Precision	Recall	f1-score		support
Class0: Non - Palindrome		1.00	1.00	1.00		992
Class1: Palindrome		1.00	1.00	1.00		32
Accuracy	Total				1.00	1024
	Class 0: NP				1.00	992
	Class 1: P				1.00	32

Confusion Matrix

Predicted Value	True Value: 1	True Value: 0
1	True Positive: 32	False Positive: 0
0	False Negative: 0	True Negative: 992

Interpretability of middle layer





OBSERVATIONS

W_o Anti Symmetry around axis

+w11 +w12 +w13 +w14 +w15 -w15 -w14 -w13 -w12 -w11

bo Same value for both neurons

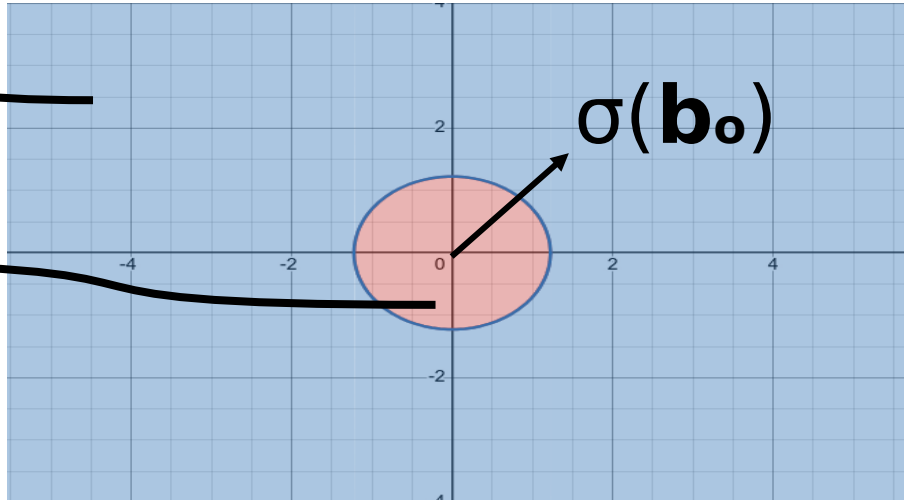
W_1 Both weights are the same

CONSEQUENCES

PALINDROME			
Zup	**bo**	**A**up	**similar**
Zlow	**bo**	**A**low	**similar**
NON PALINDROME			
Aup	**Close to O or 1**		
Alow	**Close to 1 or O**		

Learnings

NON PALINDROME REGION	
Aout	**w1 + b1**
PALINDROME REGION	
Aout	**2 x w1 x σ(b₀) + b1**
NOTE :	**σ(b₀) > 0.5**



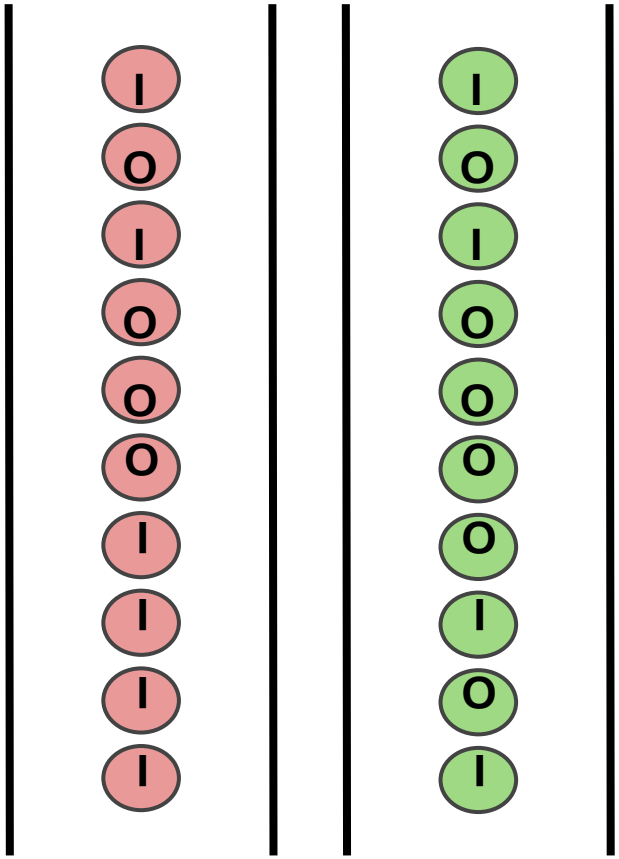
OUTER NETWORK JUST learns this decision boundary as above

- Two neuron in hidden layer is required if activation is sigmoid and no skip connection allowed.
- One neuron is sufficient in case of mod or square activation in hidden layer OR with skip connections
- As data is imbalance, oversampling minority class improves performance.
- Weighted BCE loss did not improved performance.

Evaluation Scheme

- Correct implementation of BP from scratch: 10 marks (show the code parts that implement weight change rules)
- Theory of BP clarity: 10
- Overall Performance: accuracy >=90: 10 marks; 80-89: 9; 70-79: 8; 60-69: 7; 50-59: 40-49: 5; 30-39: 4. And so on
- Interpretability of middle layer: 10
- Demo -10

APPENDIX: Input representation



- Each input x is a 10 x 1 numpy array
- Each entry of the input array form the input of the first (input) layer of the neural network
- The function **generate_combinations** generates all the possible 10 bit lists

```
def generate_combinations(length):
    if length <= 0:
        return [[]]
    result = [[0], [1]]
    for _ in range(length - 1):
        result = [comb + [0] for comb in result] + [comb + [1] for comb in result]
    return result
```

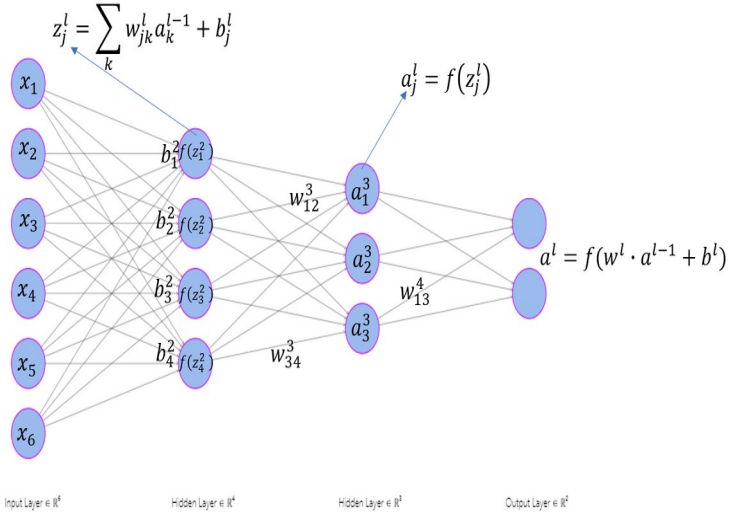
Example Non -Palindrome

Example Palindrome

Function to generate all 10 bit strings

PLEASE NOTE : This slide is not part of the presentation. This is a REFERENCE slide

APPENDIX: Theory of BP



$a^1 = [x_1, x_2 \dots \dots \dots x_n]^T$ $a^l = [a_1^l, a_2^l, \dots \dots \dots a_n^l]^T$ $b^l = [b_1^l, b_2^l, \dots \dots \dots b_n^l]^T$

$w^l = [w_{ij}^l]$

Let's define cost function as $C(a^L, y)$

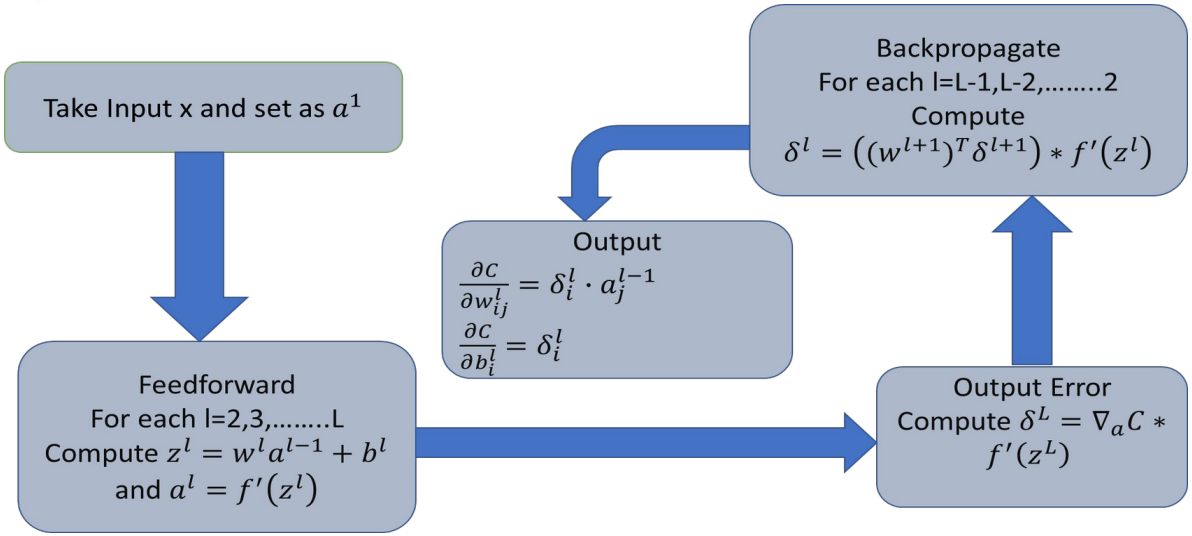
Now we have to find gradient of C with respect to $w_{ij}^l and\ b_j^l$

$$\frac{\partial C}{\partial w_{ij}^L} = \left(\frac{\partial C}{\partial a_i^L} \cdot \frac{\partial a_i^L}{\partial z_i^L}\right) a_j^{L-1} = \delta_i^L \cdot a_j^{L-1}$$

$$\frac{\partial C}{\partial b_i^L} = \left(\frac{\partial C}{\partial a_k^L} \cdot \frac{\partial a_k^L}{\partial z_i^L}\right) = \delta_i^L$$

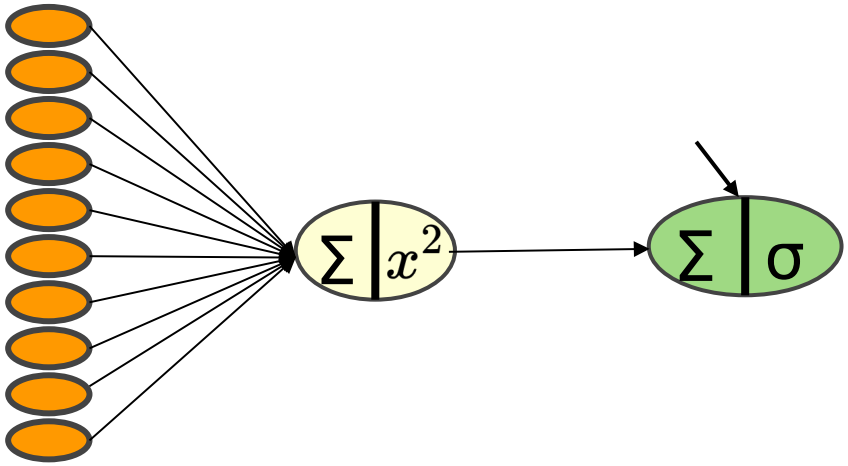
$$\frac{\partial C}{\partial w_{ij}^l} = \left(\sum_k \frac{\partial C}{\partial z_k^{l+1}} \cdot \frac{\partial z_k^{l+1}}{\partial a_i^l} \cdot \frac{\partial a_i^l}{\partial z_i^l}\right) a_j^{l-1} = \left(\sum_k \delta_k^{l+1} w_{ki}^{l+1} f'(z_i^l)\right) a_j^{l-1} = \delta_i^l \cdot a_j^{L-1}$$

$$\frac{\partial C}{\partial b_i^L} = \left(\sum_k \frac{\partial C}{\partial z_k^{l+1}} \cdot \frac{\partial z_k^{l+1}}{\partial a_i^l} \cdot \frac{\partial a_i^l}{\partial z_i^l}\right) = \left(\sum_k \delta_k^{l+1} w_{ki}^{l+1} f'(z_i^l)\right) = \delta_i^L$$



PLEASE NOTE : This slide is not part of the presentation. This is a REFERENCE slide

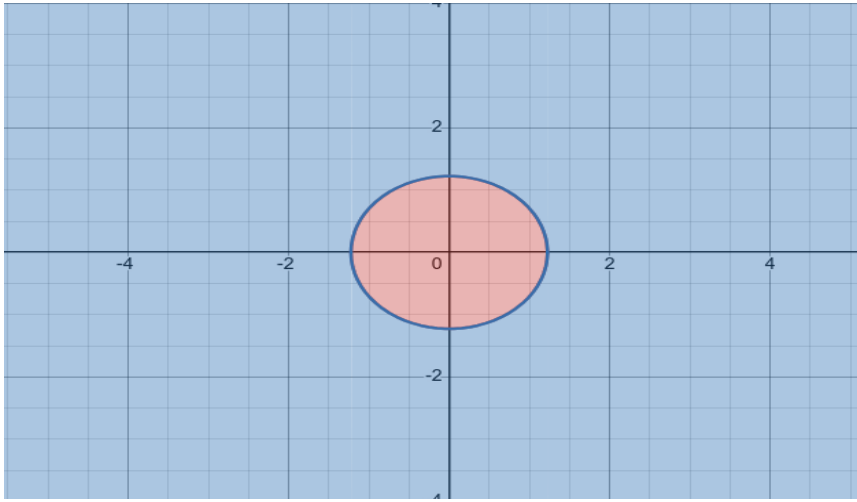
APPENDIX: One Neuron hidden layer (square activation)

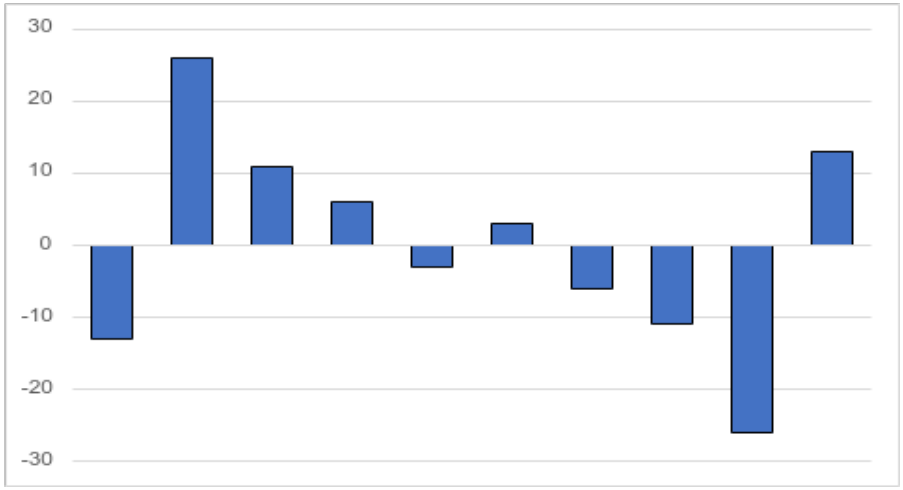


INPUT LAYER

HIDDEN LAYER

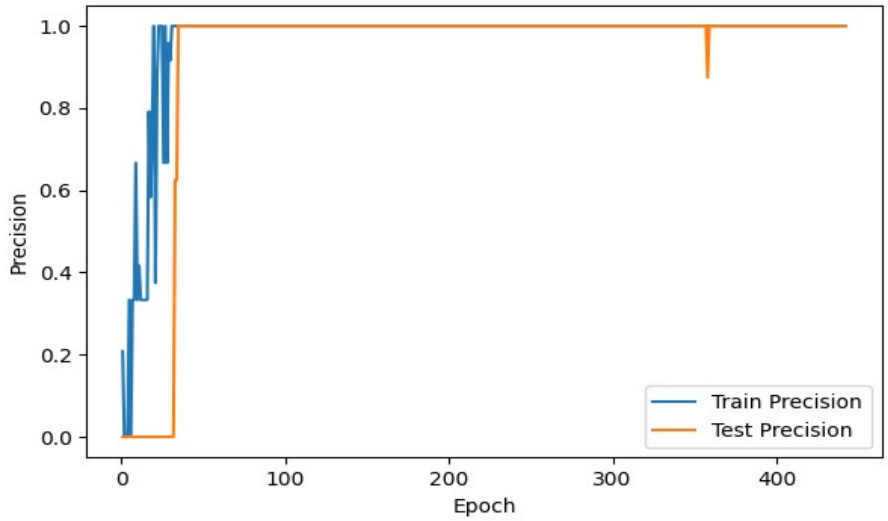
OUTPUT LAYER

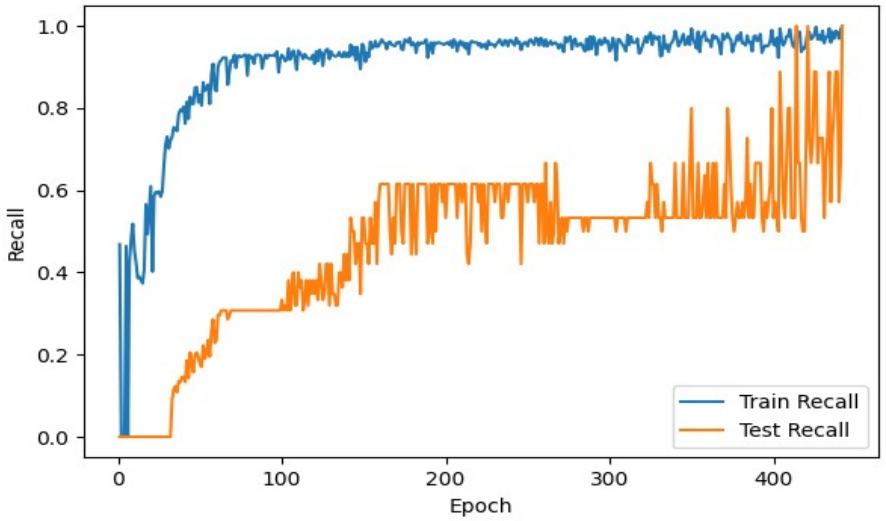


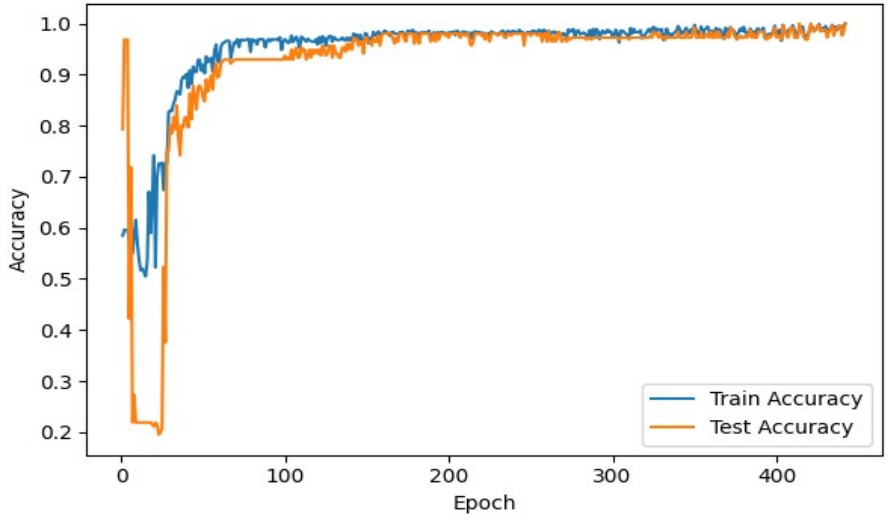


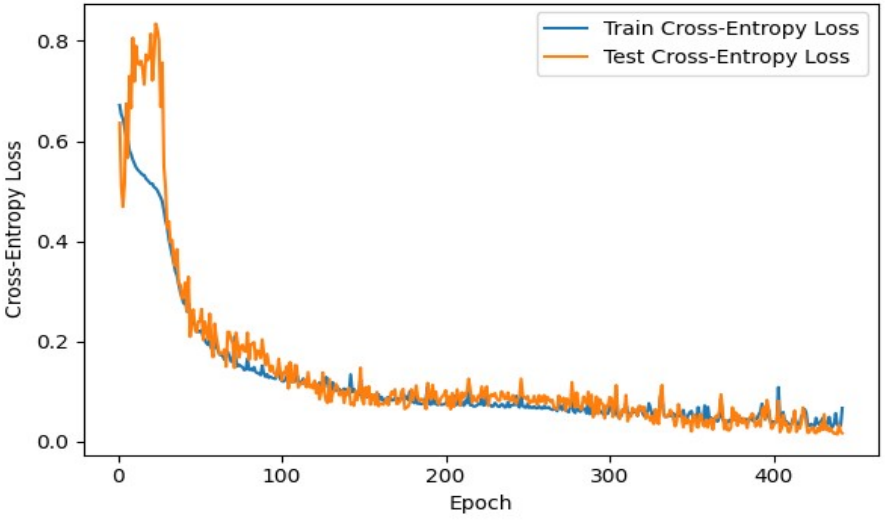
PLEASE NOTE : This slide is not part of the presentation. This is a REFERENCE slide

APPENDIX: Plots









PLEASE NOTE : This slide is not part of the presentation. This is a REFERENCE slide