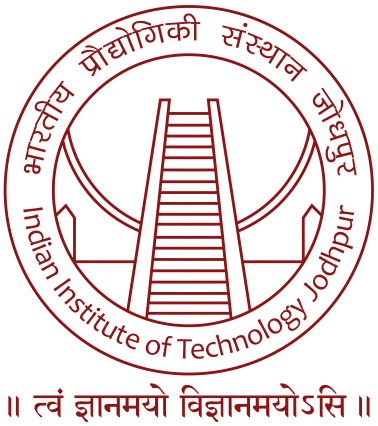
Indian Institute of Technology, Jodhpur



Building Transformer from Scratch and using it for Multi-class Classification

CSL7590: Deep Learning (Spring, 2024)

Assignment: 2

Due Date: February 22, 2024

Name:- Sahil
Roll No.:- M21MA210

M.Sc - M.Tech (Data and Computational Sciences)
Department of Mathematics

Contents

1 Exploratory Data Analysis **3**

2 Architecture 1: Use 1D - Convolution for feature extraction and perform multi-class classification **5**
2.1 Network Architecture 1 (Layer-wise description) 5
2.2 Implementation of CNN Architecture 1 (Methodology) and Training setup . . 6

3 Architecture 2: Build a transformer encoder network (from scratch) with a multi-head self-attention mechanism **6**
3.1 Network Architecture 6
3.2 Implementation of Architecture 2 (Methodology) 7

4 Results **8**
4.1 Experiment 1 8
4.2 Experiment 2 (A) Heads = 1 12
4.3 Experiment 2 (B) Heads = 2 17
4.4 Experiment 2 (C) Heads = 4 22

5 Comparative Analysis and Results **27**
5.1 Conclusions & Observations 30

6 Link to Notebook **31**

7 Resources Used **31**

List of Figures

1 Class Distribution . . . 3
2 Waveform of some sample data points . . . 4
3 Loss Curve during the training . . . 8
4 Accuracy Curve during the training . . . 8
5 Loss Curve during the training . . . 9
6 Accuracy Curve during the training . . . 9
7 Confusion Matrix . . . 10
8 F1 Scores . . . 11
9 ROC Curve . . . 12
10 Loss Curve during the training 2A . . . 13
11 Accuracy Curve during the training 2A . . . 13
12 Loss Curve during the training 2A . . . 14
13 Accuracy Curve during the training 2A . . . 14
14 Confusion Matrix 2A . . . 15
15 F1 Scores 2A . . . 16
16 ROC Curve 2A . . . 17
17 Loss Curve during the training 2B . . . 18
18 Accuracy Curve during the training 2B . . . 18
19 Loss Curve during the training 2B . . . 19
20 Accuracy Curve during the training 2B . . . 19
21 Confusion Matrix 2B . . . 20
22 F1 Scores 2B . . . 21
23 ROC Curve 2B . . . 22
24 Loss Curve during the training 2C . . . 23
25 Accuracy Curve during the training 2C . . . 23
26 Loss Curve during the training 2C . . . 24
27 Accuracy Curve during the training 2C . . . 24
28 Confusion Matrix 2C . . . 25
29 F1 Scores 2C . . . 26
30 ROC Curve 2C . . . 27
31 Training Accuracies . . . 28
32 Training Losses . . . 28
33 Validation Accuracies . . . 29
34 Validation Losses . . . 29

List of Tables

1 Classification Report . . . 11
2 Classification Report 2A . . . 16
3 Classification Report 2B . . . 21
4 Classification Report 2C . . . 26
5 Comparison among the models . . . 30

1 Exploratory Data Analysis

- The given data set is Audio dataset with 400 total data points. It has total 10 classes with 40 samples each.
- Shape of the waveform is: $[1, 220500]$.
- The data sample shape is: $[9, 1, 16000]$.
- Data distribution in different classes

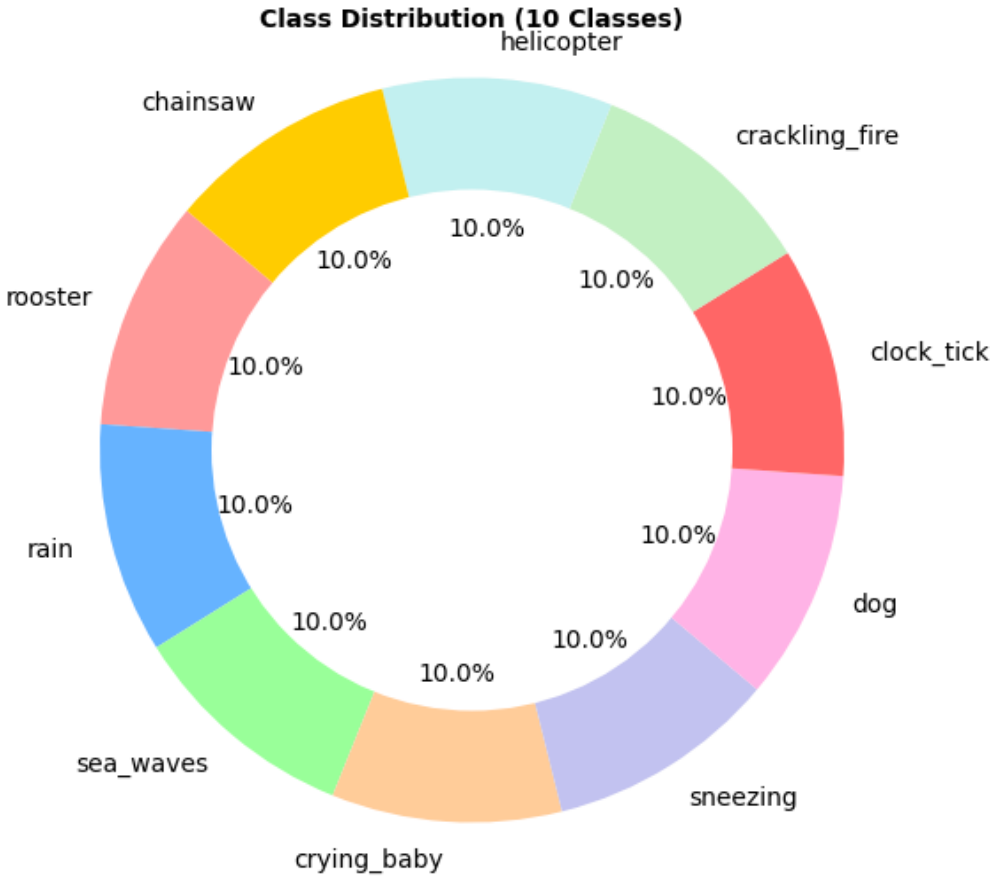


Figure 1: Class Distribution

- Visualisations of Waveform for some sample data points.

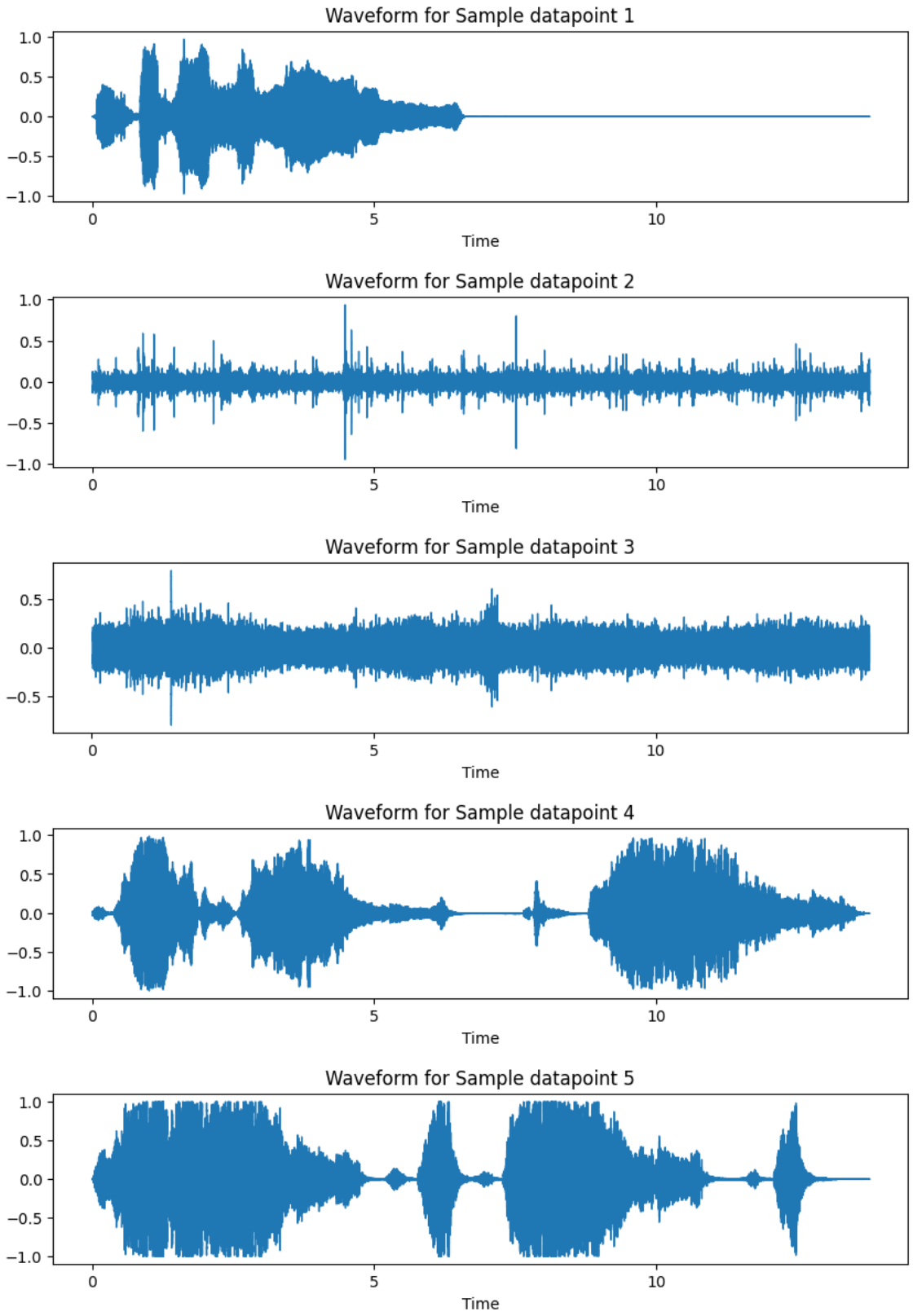


Figure 2: Waveform of some sample data points

2 Architecture 1: Use 1D - Convolution for feature extraction and perform multi-class classification

2.1 Network Architecture 1 (Layer-wise description)

The CNN Architecture used in the assignment is as follows:- It has 3 fully convolutional layers and 1 fully connected layer.

1. **Convolutional Layer (I)**
   (a) Input channels: 1
   (b) Output channels: 32
   (c) Kernel size: 3×3
   (d) Stride: 1
   (e) Zero-Padding: 1
   (f) Activation: ReLU
   (g) Pooling: Max pooling with a kernel size of 2×2 and stride of 2
2. **Convolutional Layer (II)**
   (a) Input channels: 32
   (b) Output channels: 64
   (c) Kernel size: 3×3
   (d) Stride: 1
   (e) Zero-Padding: 1
   (f) Activation: ReLU
   (g) Pooling: Max pooling with a kernel size of 2×2 and stride of 2
3. **Convolutional Layer (III)**
   (a) Input channels: 64 (as these are black and white (gray-scale) images).
   (b) Output channels: 128
   (c) Kernel size: 3×3
   (d) Stride: 1
   (e) Zero-Padding: 1
   (f) Activation: ReLU
   (g) Pooling: Average pooling with a kernel size of 2×2 and stride of 2
4. **Fully Connected Layer (Output Layer)**
   (a) Flattened the output received from the previous layer
   (b) Fully connected (Linear) layer with 64×4000 input features

(c) Activation: Softmax

(d) Since number of classes = 10, so we set the output size = 10

5. Zero Padding is used in it so that the dimension of the input images can be preserved.
6. The standard train-test was done on the dataset (as per the given dataset).
7. Used Adam optimiser with learning rate as 0.001 and loss function as **Cross Entropy Loss**.

2.2 Implementation of CNN Architecture 1 (Methodology) and Training setup

The CNN model is trained using the training dataset in following methodology and the observations were recorded as below:

1. The data was fetched from drive and some data analysis was performed on it.
2. Performed k-fold validation, for k=4.
3. Using the CNN architecture, CNN Model was created.
4. The dataset loaded as training dataset.
5. The optimiser and loss function employed, were called for training setup and then used for our model.
6. Then for 100 epochs, the model is trained.
7. The model accuracy and loss occurred during training were recorded and were plotted on wandb platform.
8. Then the model is tested on test dataset and accuracy on test data is noted.
9. Confusion matrix and ROC Curve with AUC Scores is plotted, F1 scores are calculated and total parameters were noted.
10. Classification report for the model was obtained

3 Architecture 2: Build a transformer encoder network (from scratch) with a multi-head self-attention mechanism

3.1 Network Architecture

1. Used the Same 1D CNN network for feature extraction
2. Then made a transformer encoder architecture with multi head self attention mechanism.

3. Created a combined classifier transformer which combines all the components and operates in a single transformer encoder block

4. The different components constructed are: Multi head attention layer (contains 3 self attention layers), single transformation layer, positional encoding, Encoder Block and a final transformer classifier

3.2 Implementation of Architecture 2 (Methodology)

1. Performed k-fold validation, for k=4.

2. The data is first fed into the CNN base network which was designed in Architecture 1.

3. The output of CNN serves as embedding or features for the transformer input.

4. Employ a cls token which has same dimensions as that of input embeddings.

5. This cls token is then concatenated with these inputs and then positional encodings were given to them so that transformer can learn about their relative positions.

6. The embeddings are then divided into the mutliple heads where each head is a self attention layer. where using a MLP layer these are divided to generate keys, queries and values

7. In each self attention layer, the alignment scores then attention scores are calculated and these values of attentions are multiplied with valued to get the context vectors from all the heads and using MLP, these are concatenated to get a single output, which serves as hidden input to other layers of the multi head self attention layers.

8. In this way based on the number of self attention layers (here used 3 layers), the input data is flow into the architecture and at last we use a fully connected layer and representation of cls token serves as the classification task.

9. Then for 100 epochs, the model is trained.

10. The model accuracy and loss occurred during training were recorded and were plotted on wandb platform.

11. Then the model is tested on test dataset and accuracy on test data is noted.

12. Confusion matrix and ROC Curve with AUC Scores is plotted, F1 scores are calculated and total parameters were noted.

13. Classification report for the model was obtained

4 Results

4.1 Experiment 1

Following are the results obtained when we classified the data for 10 classes:

- On training dataset, we get accuracy of the model as: 99.58%
- **Curve representing the loss** during the training for Experiment 1

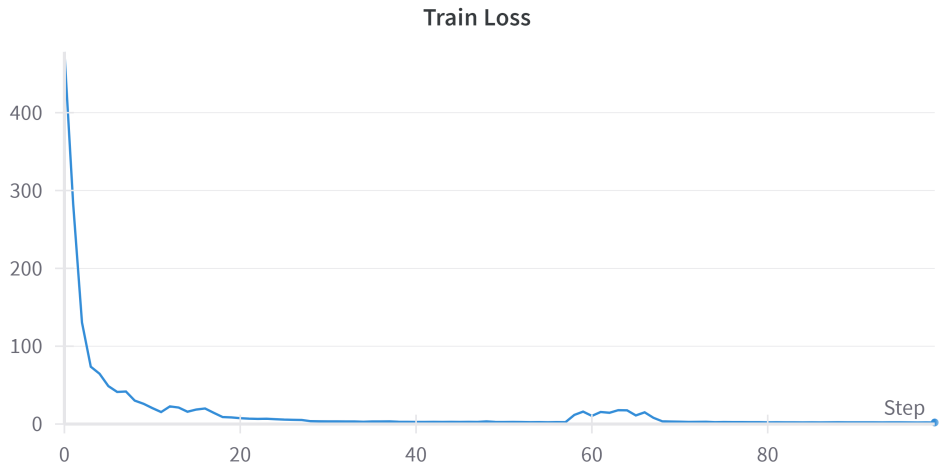


Figure 3: Loss Curve during the training

- **Curve representing the Accuracy** during the training for Experiment 1

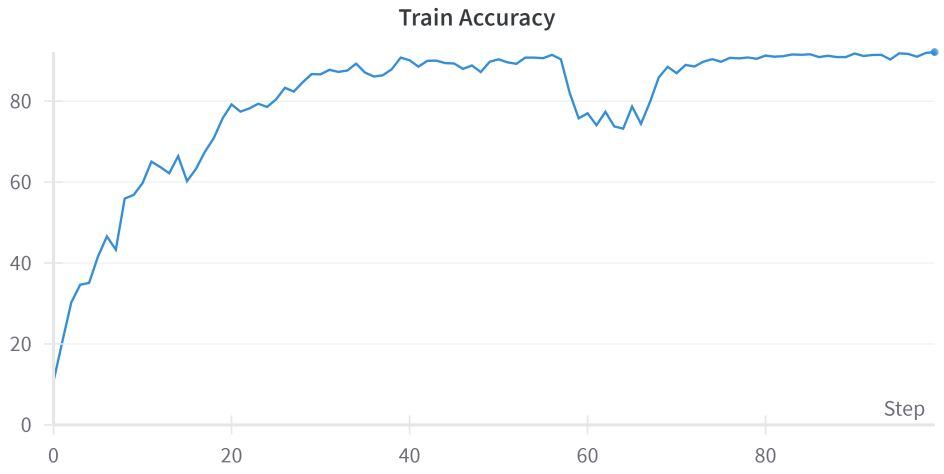


Figure 4: Accuracy Curve during the training

- **Curve representing the loss** during the Validation phase for Experiment 1

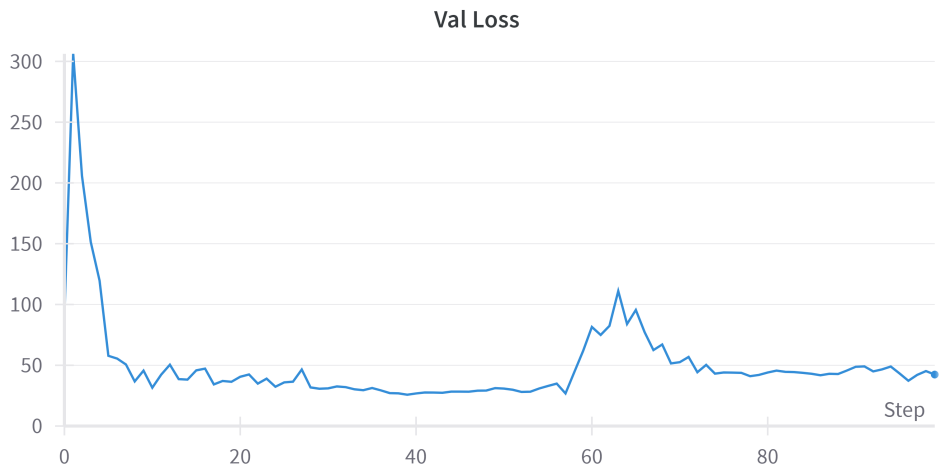


Figure 5: Loss Curve during the training

- **Curve representing the Accuracy** during the Validation phase for Experiment 1

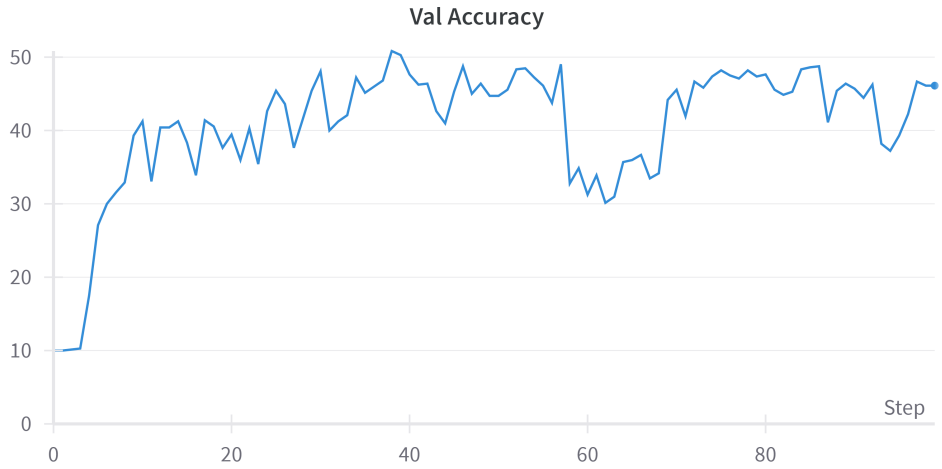


Figure 6: Accuracy Curve during the training

- **Accuracy** obtained by the model on the test data set is as: 52.500%
- **Confusion Matrix** obtained as follows:-

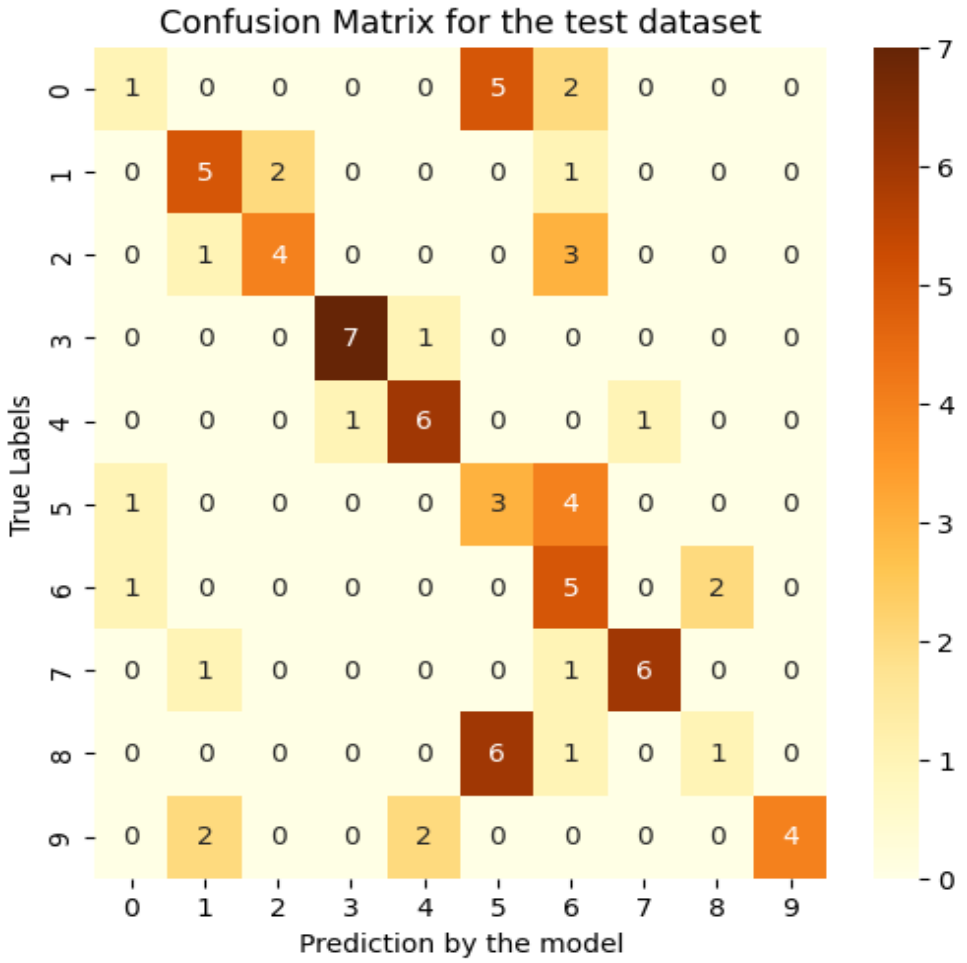


Figure 7: Confusion Matrix

- **Classification Report for Experiment 1**
- The F1 Scores are plotted as follows:

Class	Precision	Recall	F1-Score	Support
0	0.33	0.12	0.18	8
1	0.56	0.62	0.59	8
2	0.67	0.50	0.57	8
3	0.88	0.88	0.88	8
4	0.67	0.75	0.71	8
5	0.21	0.38	0.27	8
6	0.29	0.62	0.40	8
7	0.86	0.75	0.80	8
8	0.33	0.12	0.18	8
9	1.00	0.50	0.67	8
Accuracy			0.53	80
Macro Avg	0.58	0.53	0.52	80
Weighted Avg	0.58	0.53	0.52	80

Table 1: Classification Report

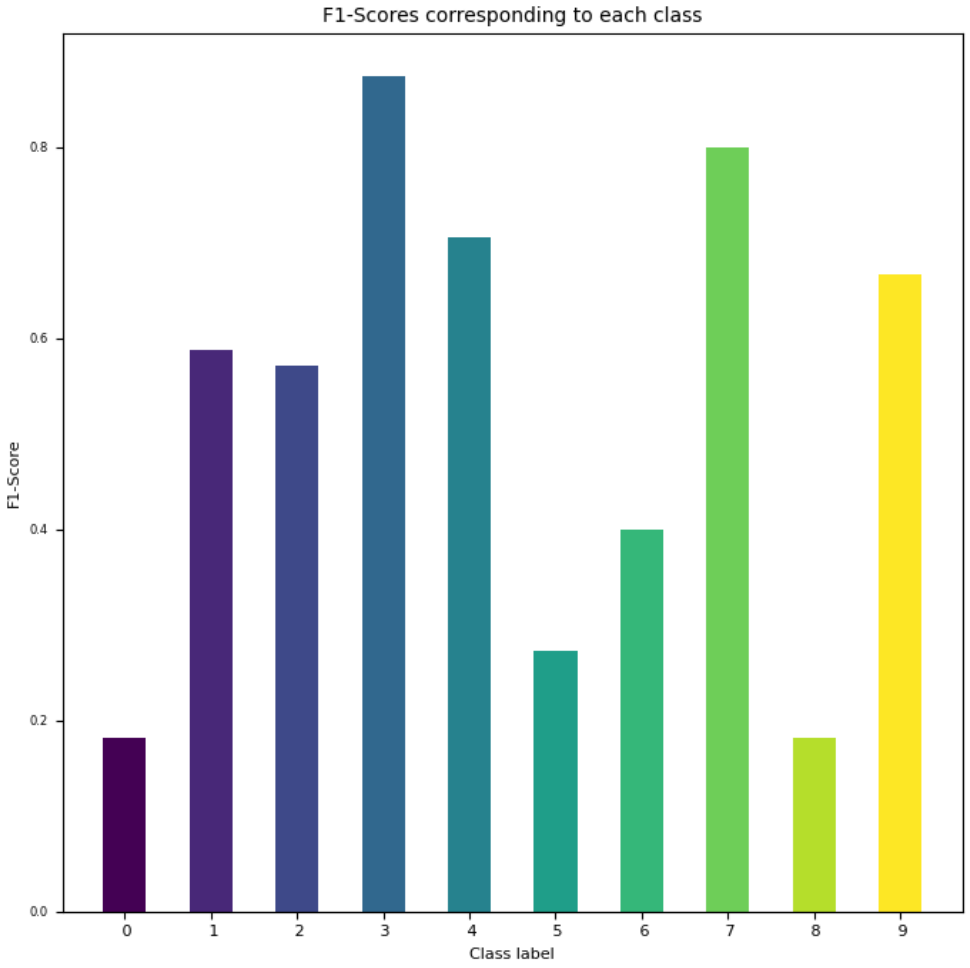


Figure 8: F1 Scores

- The ROC Curve and AUC Scores are plotted as follows:

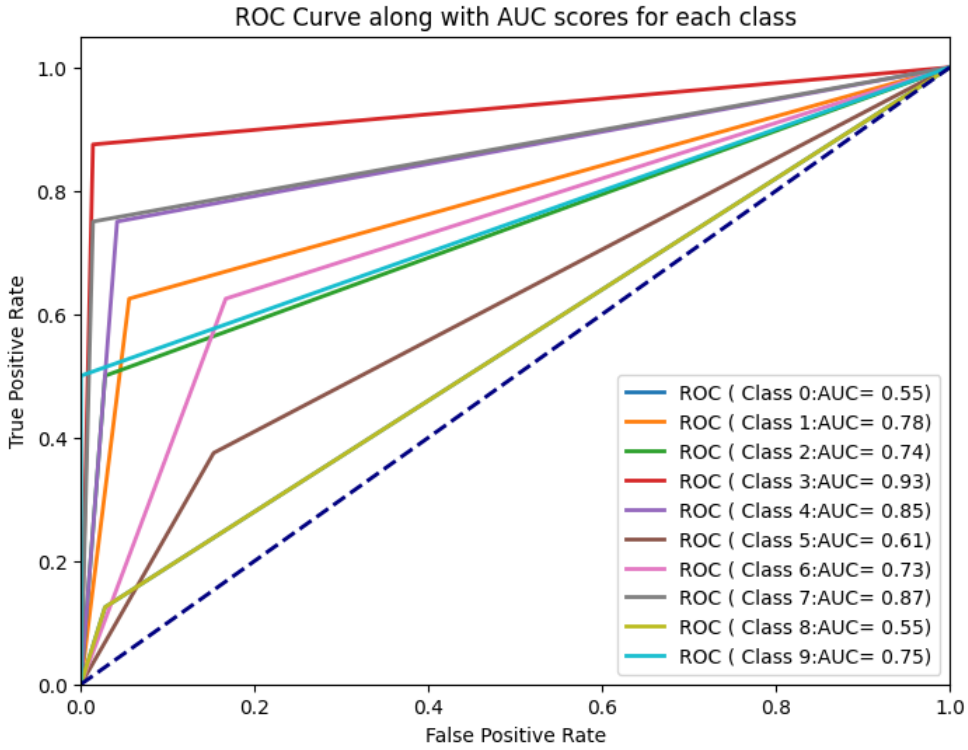


Figure 9: ROC Curve

4.2 Experiment 2 (A) Heads = 1

Following are the results obtained when we classified the data for 10 classes:

- On training dataset, we get accuracy of the model as: 12.92%
- **Curve representing the loss** during the training for Experiment 2A

Figure 10: Loss Curve during the training 2A

- **Curve representing the Accuracy** during the training for Experiment 2A

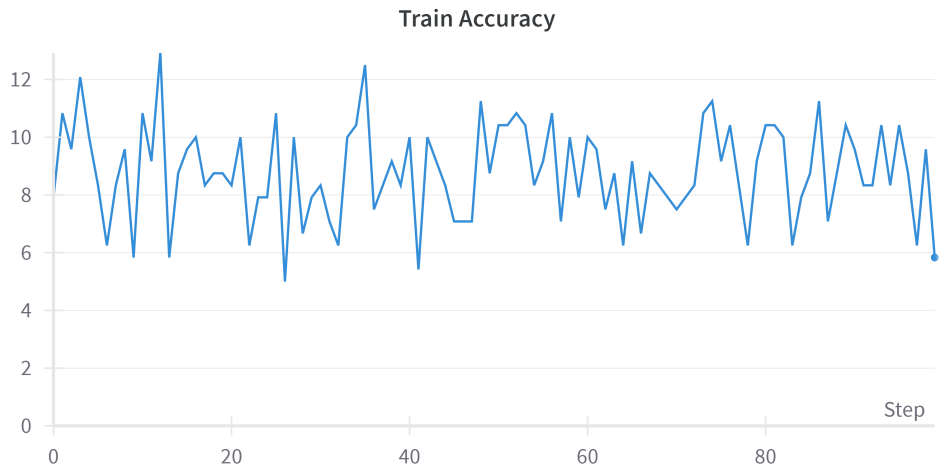


Figure 11: Accuracy Curve during the training 2A

- **Curve representing the loss** during the Validation phase for Experiment 2A

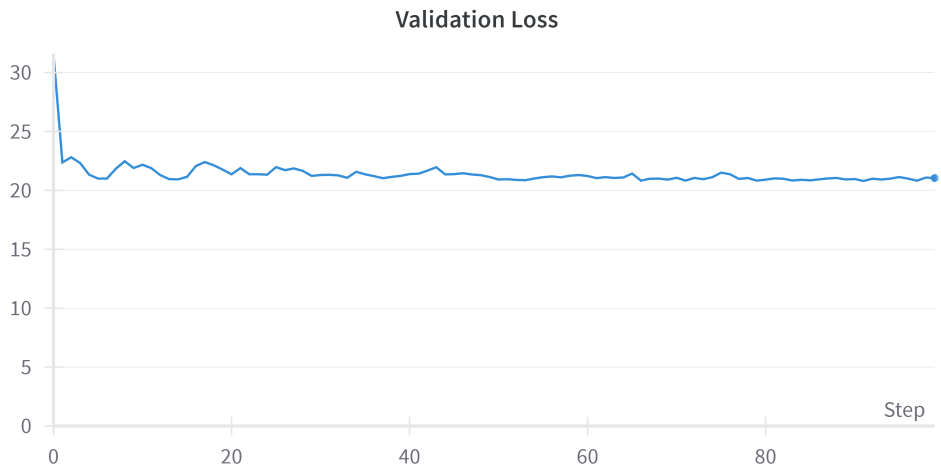


Figure 12: Loss Curve during the training 2A

- **Curve representing the Accuracy** during the Validation phase for Experiment 2A

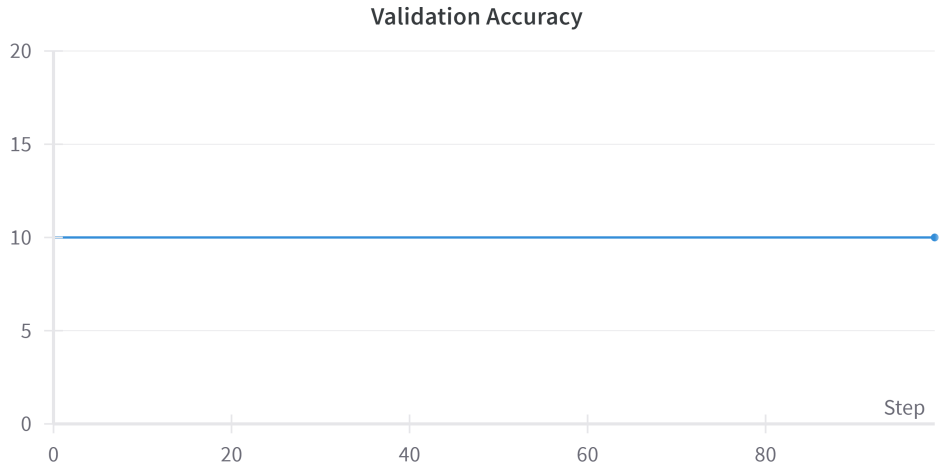


Figure 13: Accuracy Curve during the training 2A

- **Accuracy** obtained by the model on the test data set is as: 10.00%
- **Confusion Matrix** obtained as follows:-

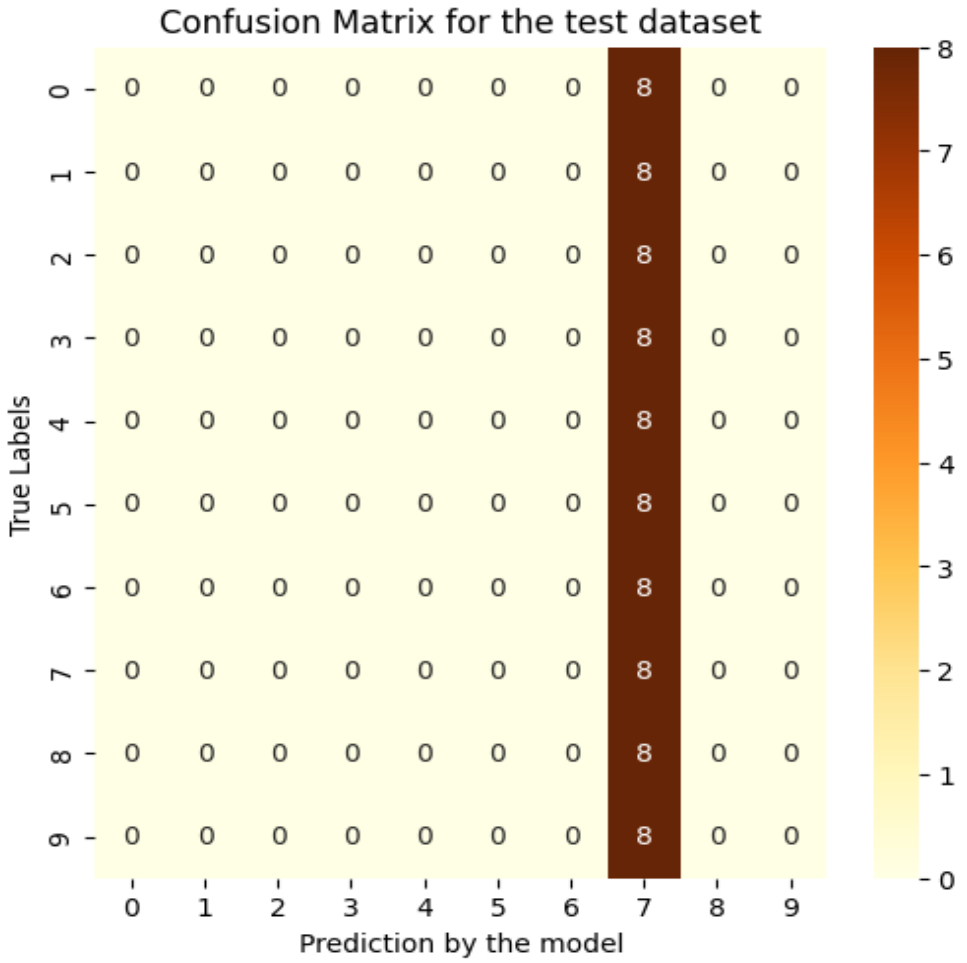


Figure 14: Confusion Matrix 2A

- **Classification Report for Experiment 1**
- The F1 Scores are plotted as follows:

Class	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	8
2	0.00	0.00	0.00	8
3	0.00	0.00	0.00	8
4	0.00	0.00	0.00	8
5	0.00	0.00	0.00	8
6	0.00	0.00	0.00	8
7	0.10	1.00	0.18	8
8	0.00	0.00	0.00	8
9	0.00	0.00	0.00	8
Accuracy			0.10	80
Macro Avg	0.01	0.10	0.02	80
Weighted Avg	0.01	0.10	0.02	80

Table 2: Classification Report 2A

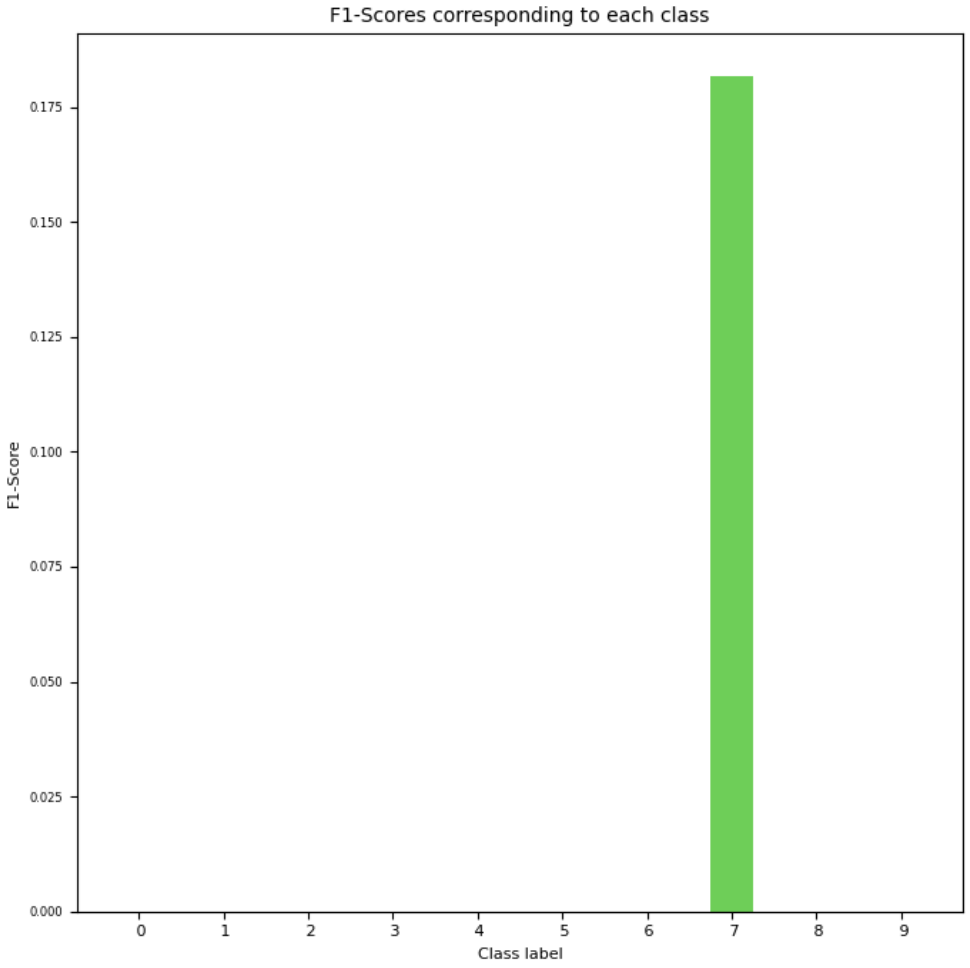


Figure 15: F1 Scores 2A

- The ROC Curve and AUC Scores are plotted as follows:

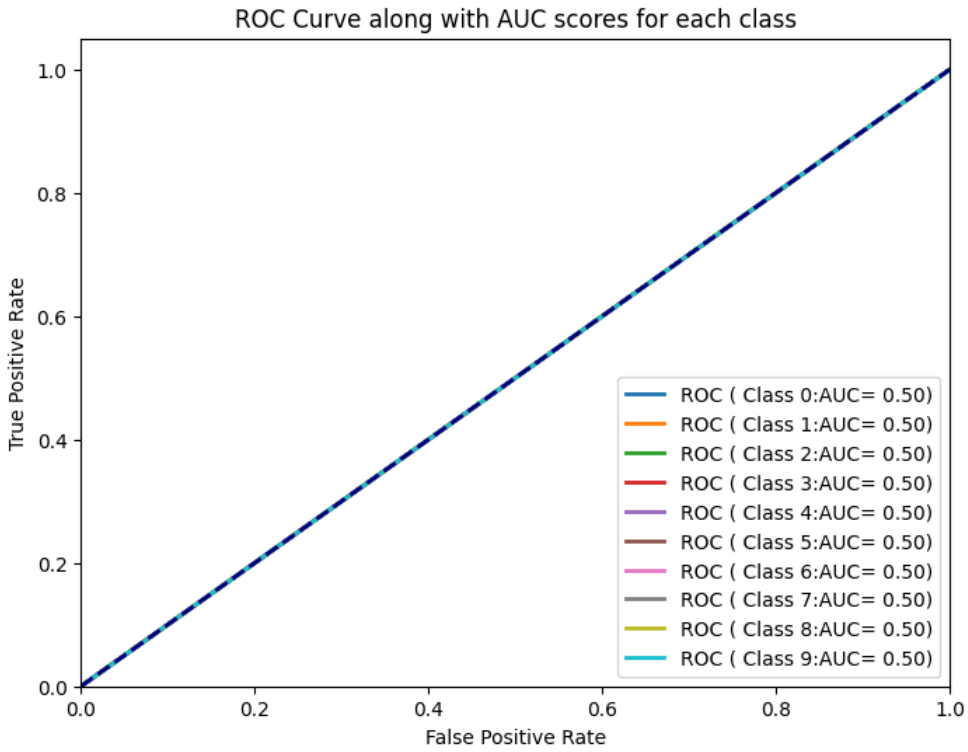


Figure 16: ROC Curve 2A

4.3 Experiment 2 (B) Heads = 2

Following are the results obtained when we classified the data for 10 classes:

- On training dataset, we get accuracy of the model as: 12.50%
- **Curve representing the loss** during the training for Experiment 2B

Figure 17: Loss Curve during the training 2B

- **Curve representing the Accuracy** during the training for Experiment 2B

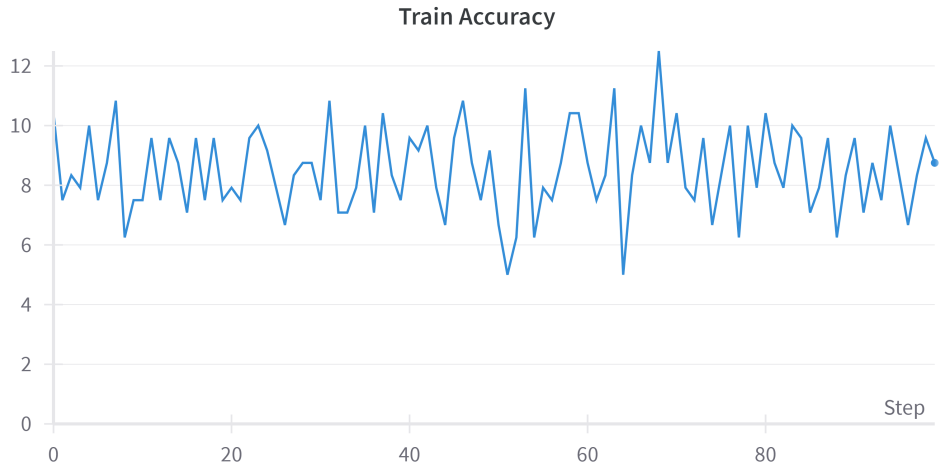


Figure 18: Accuracy Curve during the training 2B

- **Curve representing the loss** during the Validation phase for Experiment 2B

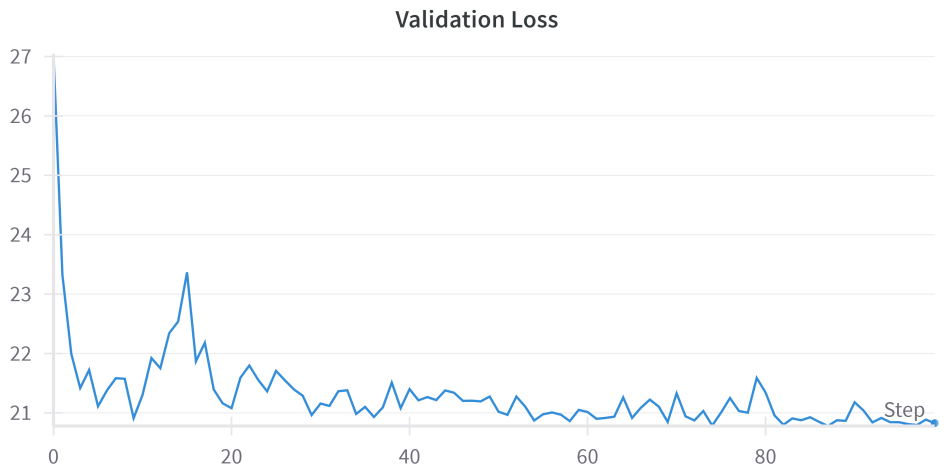


Figure 19: Loss Curve during the training 2B

- **Curve representing the Accuracy** during the Validation phase for Experiment 2B

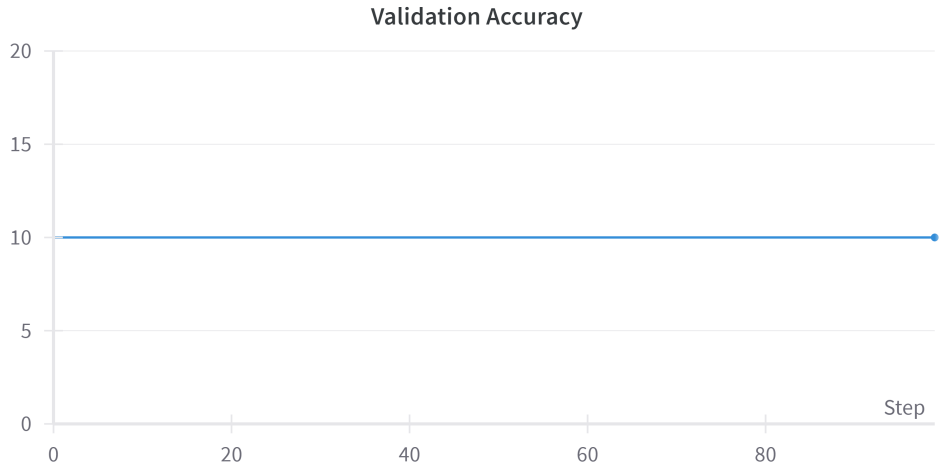


Figure 20: Accuracy Curve during the training 2B

- **Accuracy** obtained by the model on the test data set is as: 10.00%
- **Confusion Matrix** obtained as follows:-

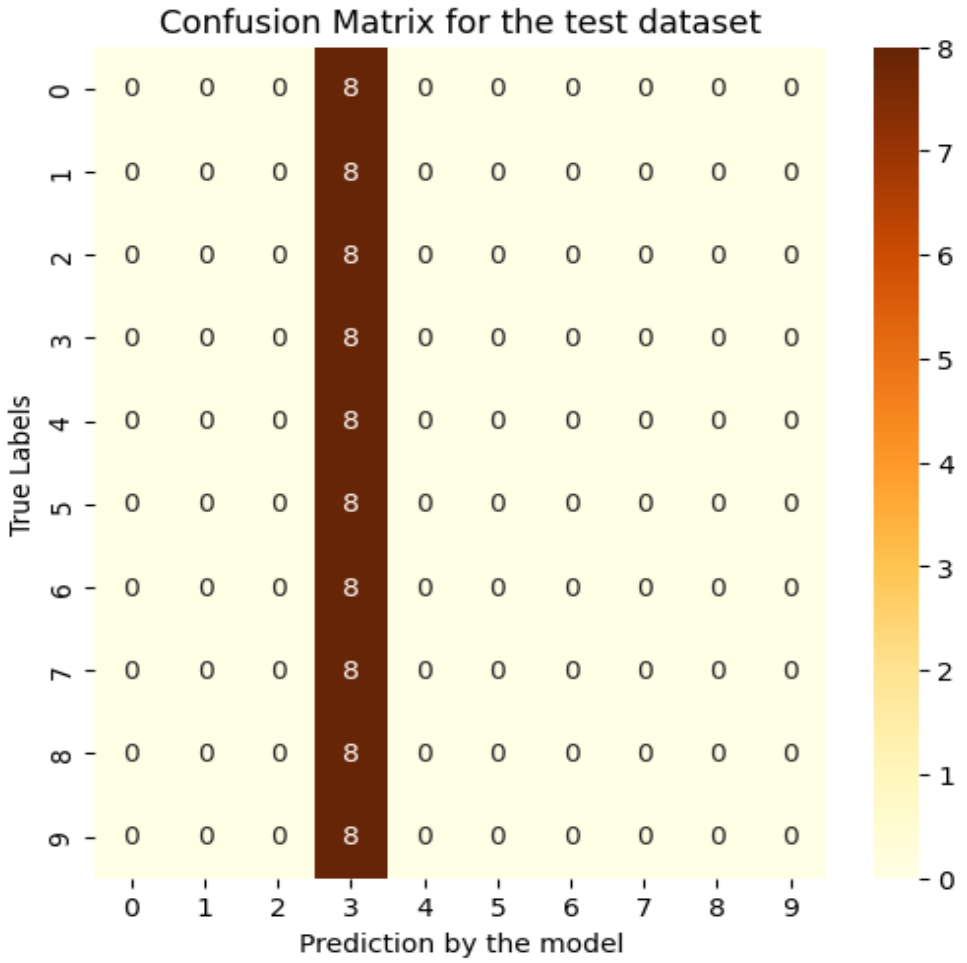


Figure 21: Confusion Matrix 2B

- **Classification Report for Experiment 1**
- The F1 Scores are plotted as follows:

Class	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	8
2	0.00	0.00	0.00	8
3	0.10	1.00	0.18	8
4	0.00	0.00	0.00	8
5	0.00	0.00	0.00	8
6	0.00	0.00	0.00	8
7	0.00	0.00	0.00	8
8	0.00	0.00	0.00	8
9	0.00	0.00	0.00	8
Accuracy			0.10	80
Macro Avg	0.01	0.10	0.02	80
Weighted Avg	0.01	0.10	0.02	80

Table 3: Classification Report 2B

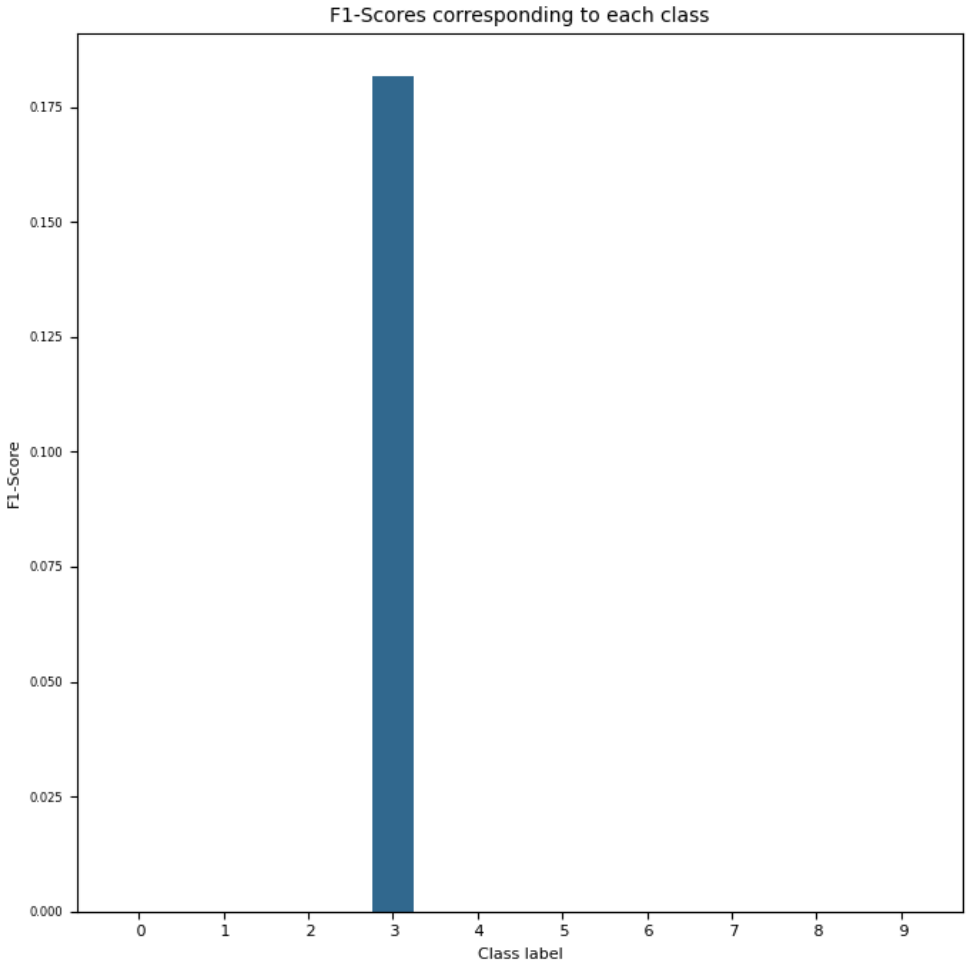


Figure 22: F1 Scores 2B

- The ROC Curve and AUC Scores are plotted as follows:

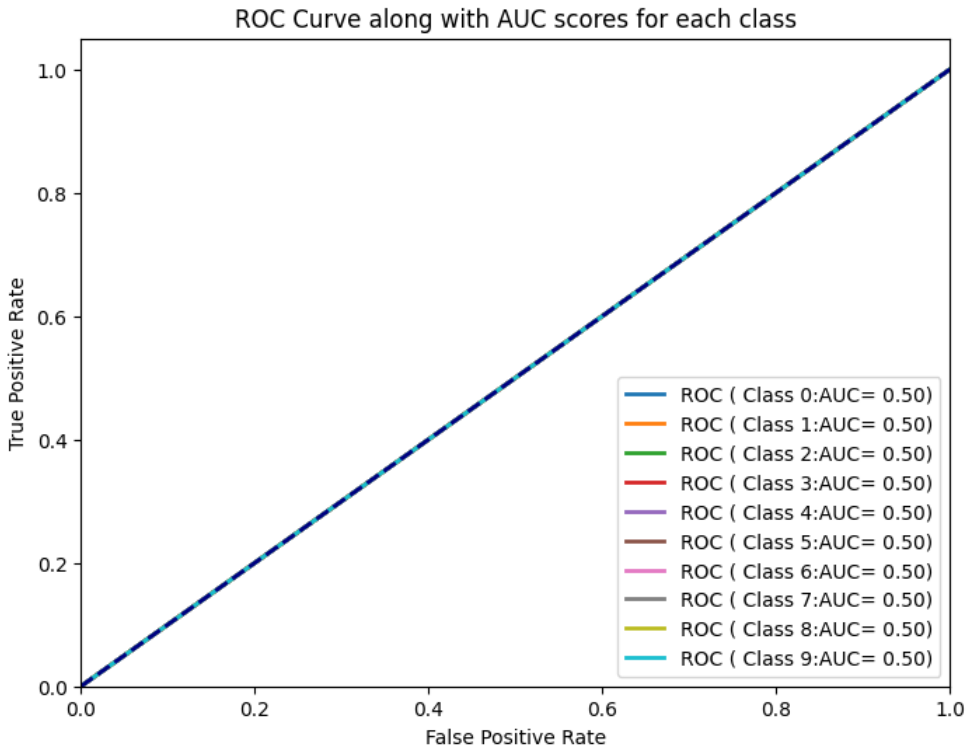


Figure 23: ROC Curve 2B

4.4 Experiment 2 (C) Heads = 4

Following are the results obtained when we classified the data for 10 classes:

- On training dataset, we get accuracy of the model as: 12.50%
- **Curve representing the loss** during the training for Experiment 2C

Figure 24: Loss Curve during the training 2C

- **Curve representing the Accuracy** during the training for Experiment 2C

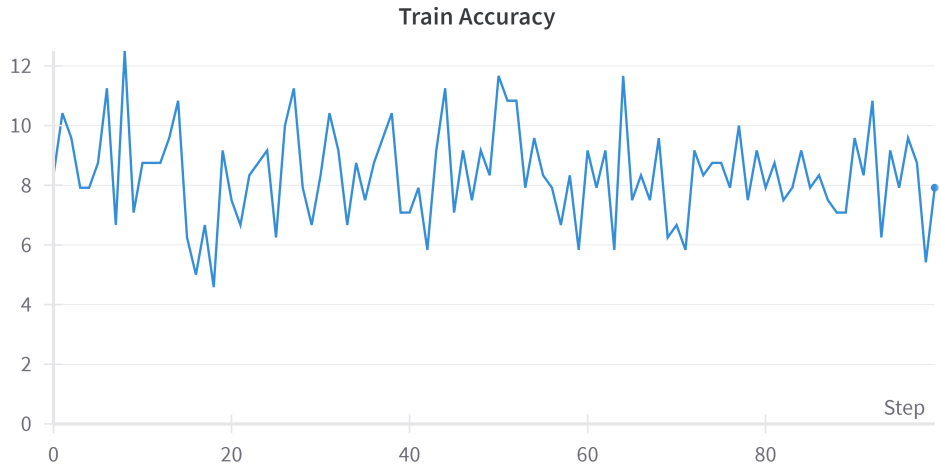


Figure 25: Accuracy Curve during the training 2C

- **Curve representing the loss** during the Validation phase for Experiment 2C

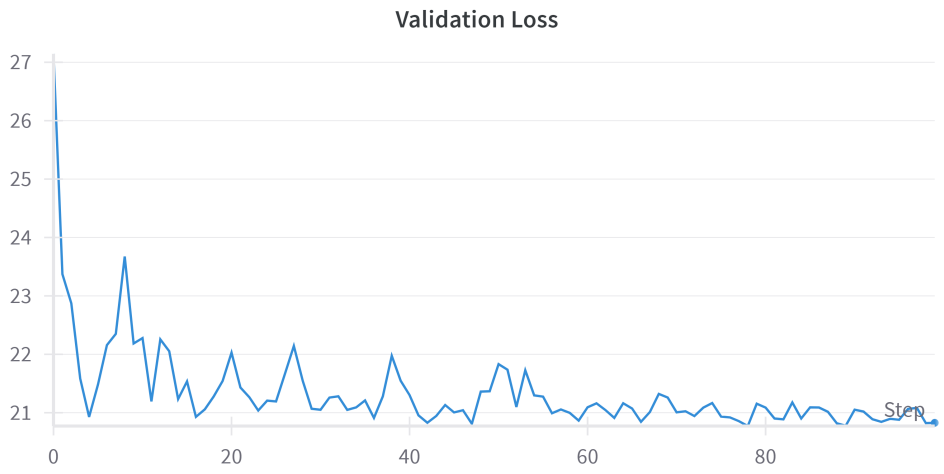


Figure 26: Loss Curve during the training 2C

- **Curve representing the Accuracy** during the Validation phase for Experiment 2C

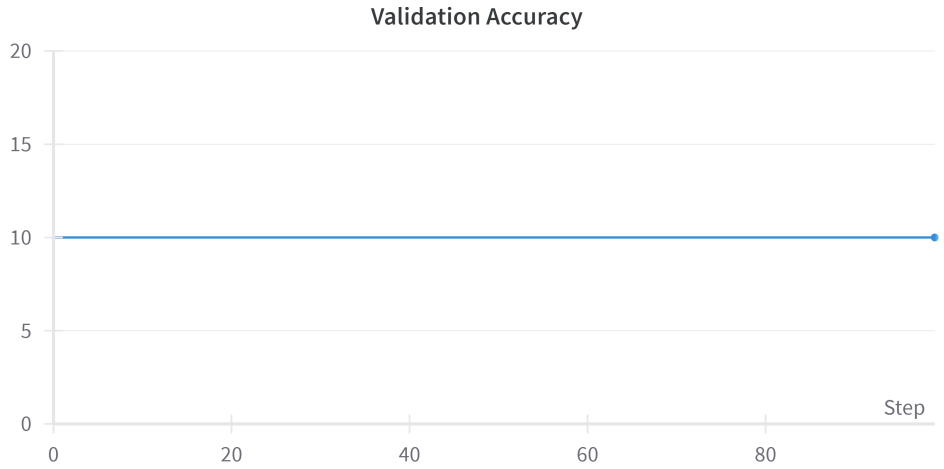


Figure 27: Accuracy Curve during the training 2C

- **Accuracy** obtained by the model on the test data set is as: 10.00%
- **Confusion Matrix** obtained as follows:-

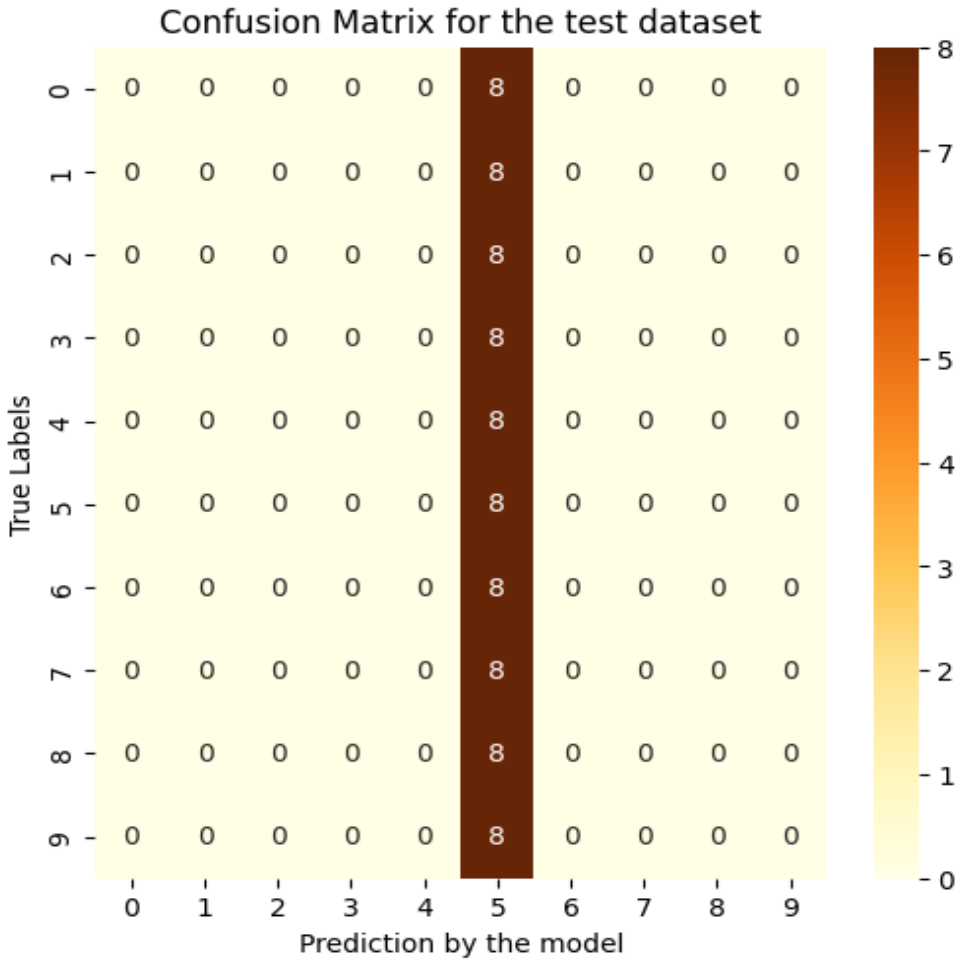


Figure 28: Confusion Matrix 2C

- **Classification Report for Experiment 1**
- The F1 Scores are plotted as follows:

Class	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	8
1	0.00	0.00	0.00	8
2	0.00	0.00	0.00	8
3	0.00	0.00	0.00	8
4	0.00	0.00	0.00	8
5	0.10	1.00	0.18	8
6	0.00	0.00	0.00	8
7	0.00	0.00	0.00	8
8	0.00	0.00	0.00	8
9	0.00	0.00	0.00	8
Accuracy			0.10	80
Macro Avg	0.01	0.10	0.02	80
Weighted Avg	0.01	0.10	0.02	80

Table 4: Classification Report 2C

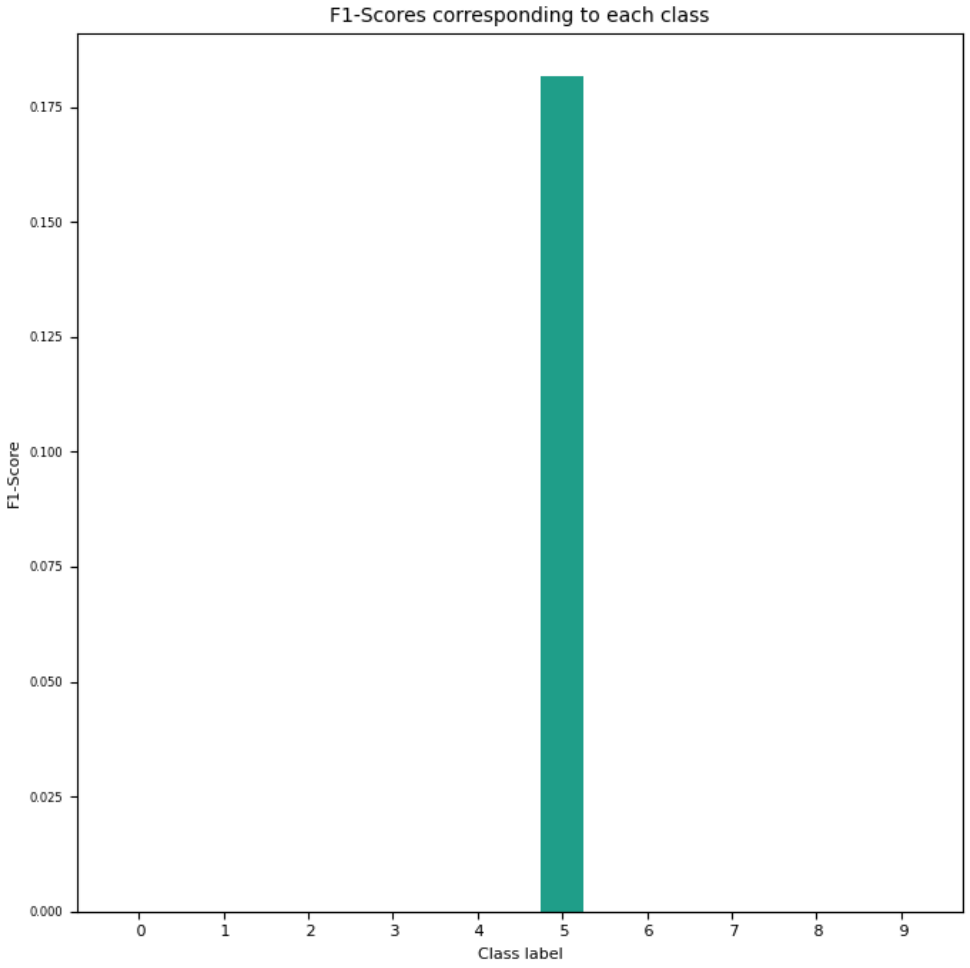


Figure 29: F1 Scores 2C

- The ROC Curve and AUC Scores are plotted as follows:

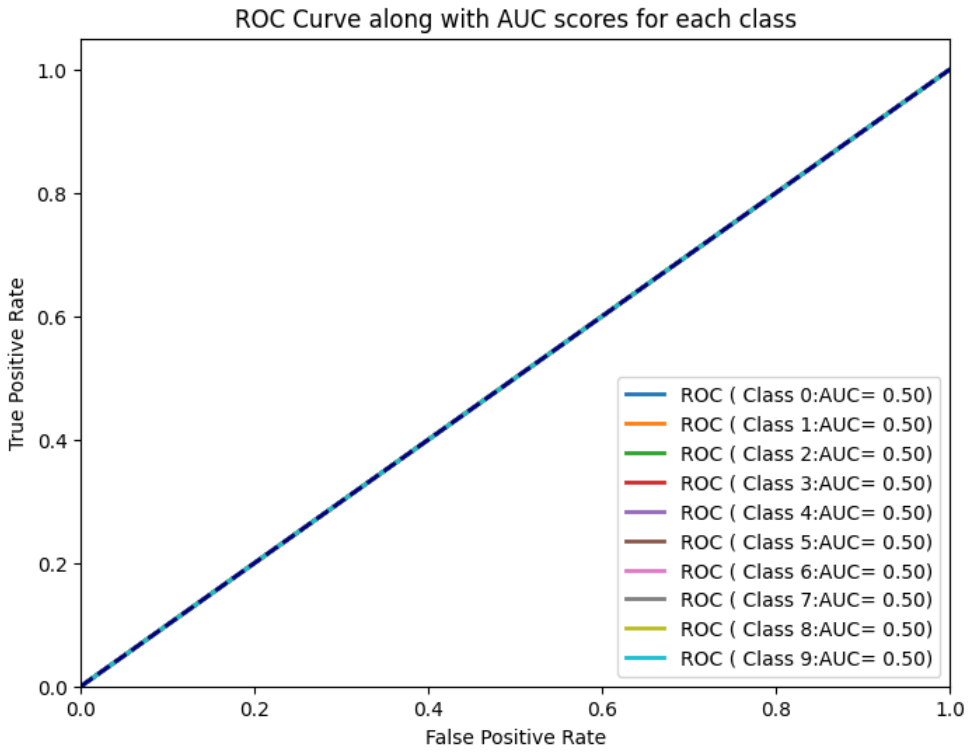


Figure 30: ROC Curve 2C

5 Comparative Analysis and Results

Overall, when the models are trained the following graph is the collective record of both the train and validation accuracy and losses.

- Training Accuracy

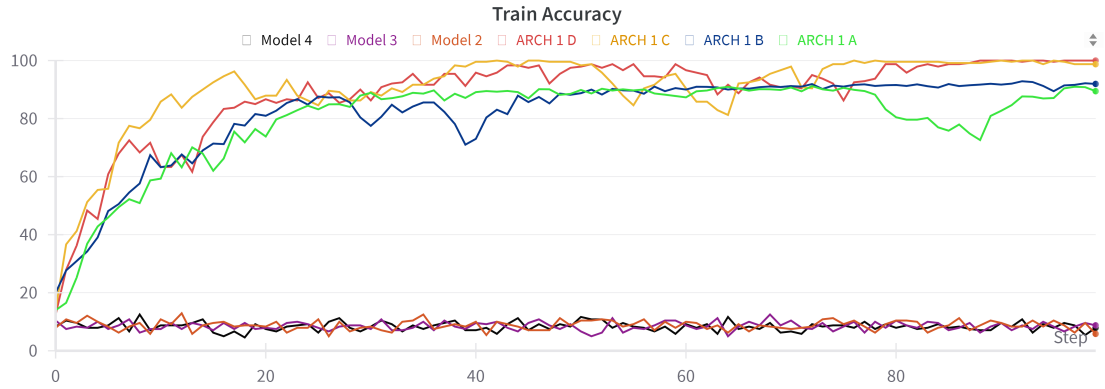


Figure 31: Training Accuracies

- Training Losses

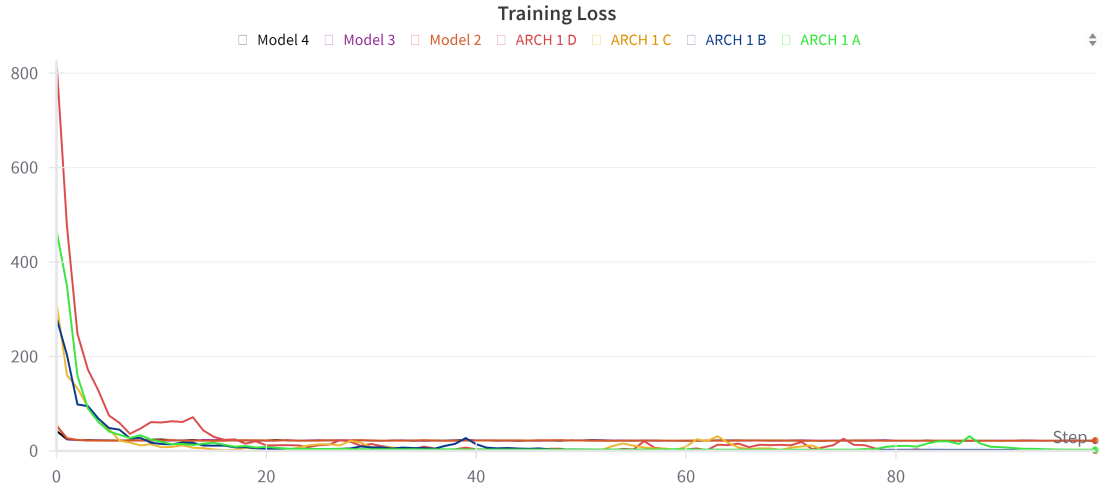


Figure 32: Training Losses

- Validation Accuracy

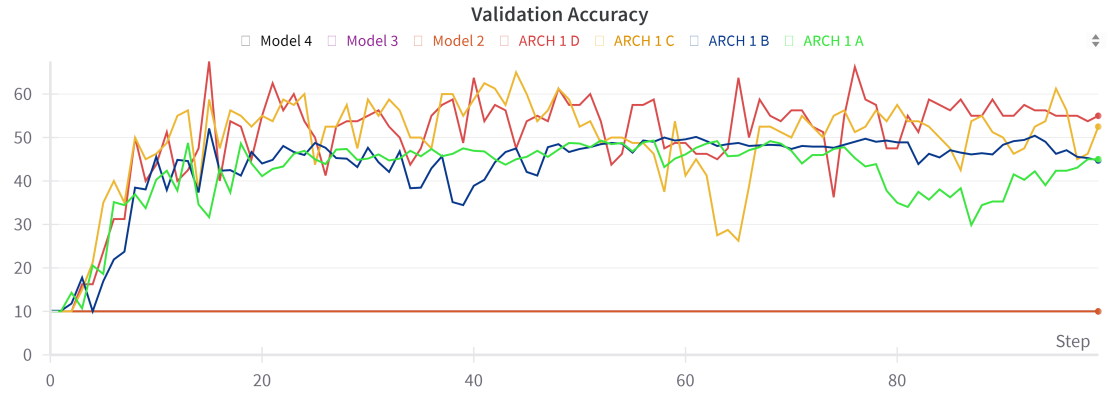


Figure 33: Validation Accuracies

- Validation Losses

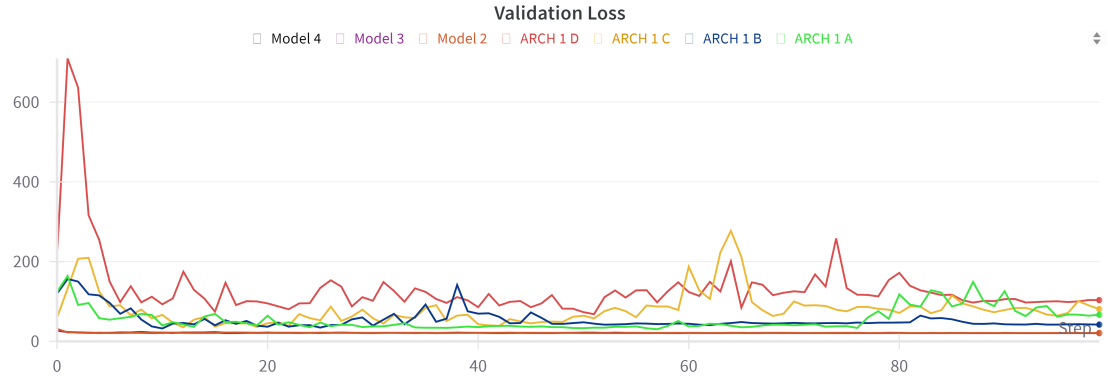


Figure 34: Validation Losses

Model	Train Accuracy	Val Accuracy	Trainable Pars	Non trainable Pars
Model 1 (Arch 1)	99.58	52.50	2591498	0
Model 2 (Arch 2A)	12.92	10.00	48467946	0
Model 3 (Arch 2B)	12.50	10.00	48467946	0
Model 4 (Arch 2C)	12.50	10.00	48467946	0

Table 5: Comparison among the models

5.1 Conclusions & Observations

1. The learning rate used is 0.001.
2. Both the models are trained on 100 epochs and then the results are noted and presented in this report.
3. For the Architecture 1, we can observe that the model is trained very well and is able to capture the relationship among the features but it achieves 52% accuracy on test dataset, which means the model gets over-fitted. The possible reason for this is the less size of data set and it leads to model learn all the information and noisy patterns in the data. As a result it doesn't fits perfectly for the test set. Some techniques were also used to reduce some amount of overfitting like data augmentation and dropout. Initially the accuracy obtained was 34% but using these regularisation techniques improved the accuracy.
4. We have seen in the lectures and referring the paper "Attention is all you need", transformers are the most powerful architectures among all the existing architectures. Once this is trained, we can fine tune it according to our tasks required and set the best hyper parameters.
5. But here we can see that the architecture of the transformer is working well but is not fined tuned. I adjusted the hyper parameters for different values, but still it is not able to fine tuned from my end and hence this is the reason that it achieved very low accuracy even on the training data.
6. The hyper parameters used are: dimension of the model: 2000, maximum length of embedding sequence = 2000, dimension of hidden feedforward network = 32, some other combinations of the hyperparameters are also used.
7. Other reason I can observe that the cls token is not able to capture the feature representation, as a result of it, during classification phase it is not able to predict the correct results but the classification is biased towards a single class.
8. In model 2A, the classification is biased towards class 7, for model 2B, it is biased towards class 3 and model 3 towards class 5.
9. Thus cls token is biased towards a particular class.
10. Once the hyper-parameters are fine tuned, then the transformer will be able to solve the task with more and more accuracy.

11. Thus at all, it can be concluded that this transformer architecture was not able to get fined tuned which is the highlights of obtaining the low accuracy.

12. Conclusion can be made that model got stuck in local minima or gets underfit.

6 Link to Notebook

Google Collab Notebook link:- Click here to navigate to Google Colab Notebook

7 Resources Used

- https://pytorch.org/tutorials/beginner/nlp/pytorch_tutorial.html
- https://pytorch.org/docs/stable/nn.html
- https://www.kaggle.com/code/buchan/transformer-network-with-1d-cnn-feature-extraction
- Lecture Slides provided by the instructor
- https://arxiv.org/abs/1706.03762

********** THE END **********