


Deep Learning Assignment 2

 m23csa011-DL2.ipynb

Name : Kushal Agrawal

Roll No: M23CSA011

Work Prior to defining Architecture 1 :-

1. Imports and Setup:

- The code starts by importing necessary libraries and setting up the environment for running the code in Google Colab.
- torch, pandas, torchaudio, numpy, and other libraries are imported for various functionalities such as deep learning, data manipulation, audio processing, and numerical operations.
- Additional packages like torchinfo, scikit-plot, wandb, and lightning are installed via pip to provide additional functionalities like model inspection, visualization, and logging.

2. Mounting Google Drive:

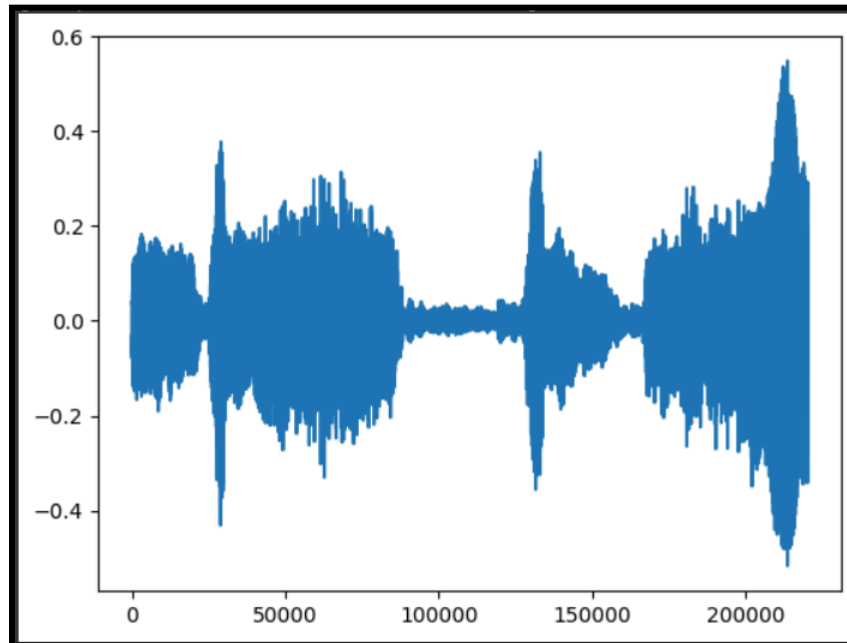
- The Google Drive is mounted to access files stored there. This is useful for loading datasets or saving model checkpoints.

3. Data Loading and Preparation:

- The ESC-50 dataset metadata (CSV file) is loaded using pandas. This CSV file contains information about the audio files in the dataset, such as filenames, labels, and folds.
- Audio files are loaded and processed using torchaudio. The waveform and sample rate of an audio file are loaded for exploratory data analysis.

```
Shape of waveform torch.Size([1, 220500])
```

```
Sample_rate : 44100
```



4. Custom Dataset and DataLoader Classes:

- Custom classes are defined to handle dataset creation and data loading.
- CustomDataset class preprocesses audio data and splits it into smaller samples (9 frames and each frame with 16000 samples) to be fed into the neural network for training.
- CustomDataModule class sets up dataloaders for training, validation, and testing. It handles data splitting and batching, ensuring efficient data loading during training.
- We later combine all the frames into one big tensor of size 144000.

5. Training and Evaluation Functions:

- Functions for training (train) and evaluating (validate) the model are defined.
- These functions utilize PyTorch's capabilities for training and evaluating neural networks.
- Training and validation loops are implemented, calculating loss and accuracy metrics to monitor model performance during training.

6. Model Training :

- A function (run_train_eval) is defined to train and evaluate the model on the train and validation set.
- The model is trained for a specified number of epochs using the training and validation functions.
- Training progress is logged using Weights & Biases (wandb) for visualization and tracking of metrics like loss and accuracy.

7. Testing Functionality:

- A function (run_test) is defined to evaluate the model on the test set.
- Metrics like accuracy, F1-score, confusion matrix, and ROC curve are computed and displayed to assess the model's performance on unseen data.

Architecture 1:- CNN Base

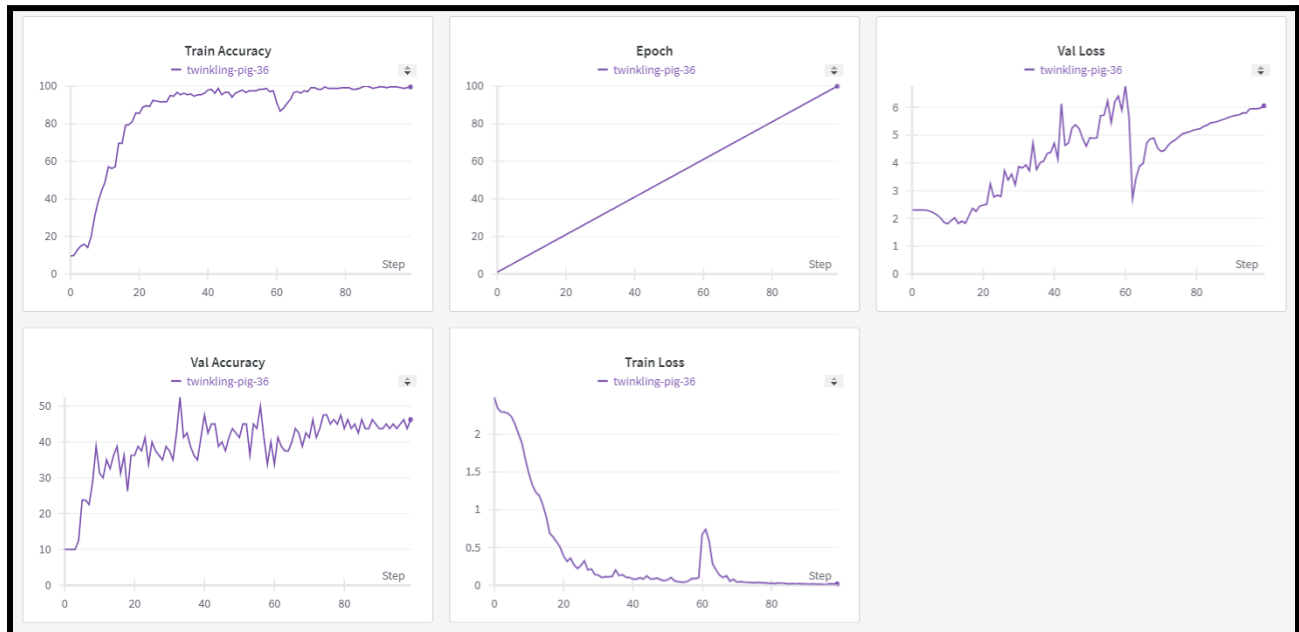
- The architecture consists of several convolutional blocks followed by fully connected layers.
- Each convolutional block consists of a convolutional layer, ReLU activation, and max-pooling layer to extract features from the audio input.

Layer (type:depth-idx)	Output Shape	Param #
Conv1DNet	[32, 10]	--
└─ConvBlock: 1-1	[32, 64, 35999]	128
└─Conv1d: 2-1	[32, 64, 71999]	512
└─ReLU: 2-2	[32, 64, 71999]	--
└─MaxPool1d: 2-3	[32, 64, 35999]	--
└─Dropout: 1-2	[32, 64, 35999]	--
└─ConvBlock: 1-3	[32, 32, 9000]	64
└─Conv1d: 2-4	[32, 32, 18000]	10,272
└─ReLU: 2-5	[32, 32, 18000]	--
└─MaxPool1d: 2-6	[32, 32, 9000]	--
└─Dropout: 1-4	[32, 32, 9000]	--
└─ConvBlock: 1-5	[32, 16, 2250]	32
└─Conv1d: 2-7	[32, 16, 4501]	1,552
└─ReLU: 2-8	[32, 16, 4501]	--
└─MaxPool1d: 2-9	[32, 16, 2250]	--
└─Dropout: 1-6	[32, 16, 2250]	--
└─FCBlock: 1-7	[32, 128]	--
└─Linear: 2-10	[32, 128]	4,608,128
└─ReLU: 2-11	[32, 128]	--
└─Linear: 1-8	[32, 10]	1,290
Total params: 4,621,978		
Trainable params: 4,621,978		
Non-trainable params: 0		
Total mult-adds (G): 7.47		
Input size (MB): 18.43		
Forward/backward pass size (MB): 1345.56		
Params size (MB): 18.49		
Estimated Total Size (MB): 1382.48		

Task 4 :- Report total trainable and non-trainable parameters.

- Total Trainable parameters 46,21,978
- Total Non-Trainable parameters 0
- Major of the parameters are because of the linear layer FC1.

Task 1:- Train for 100 epochs : Test Fold : 1 Validation Fold : 2



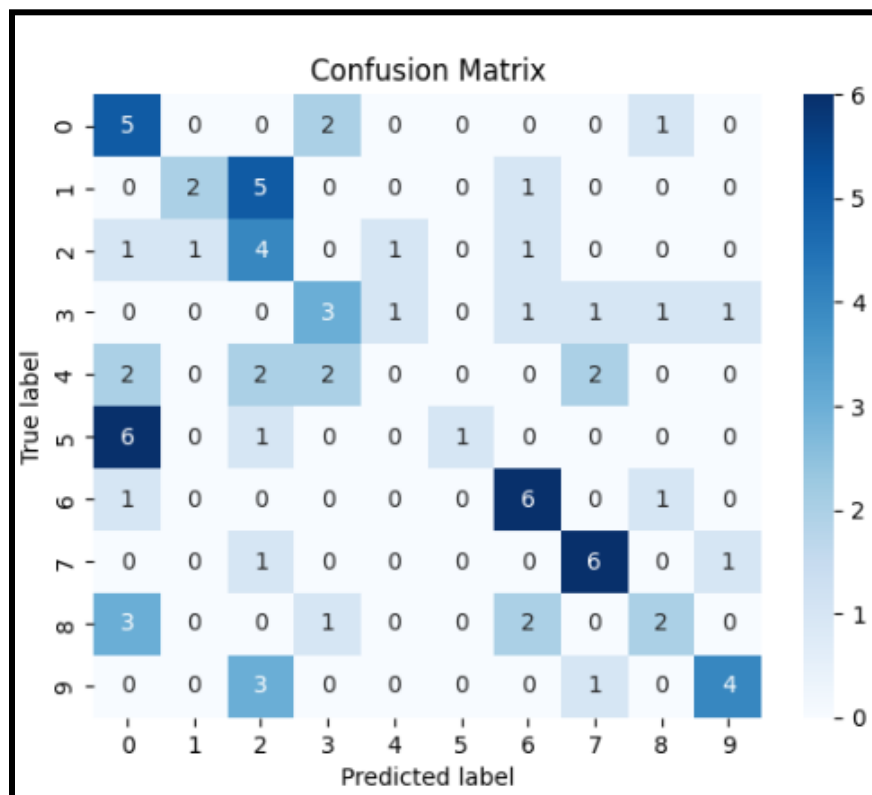
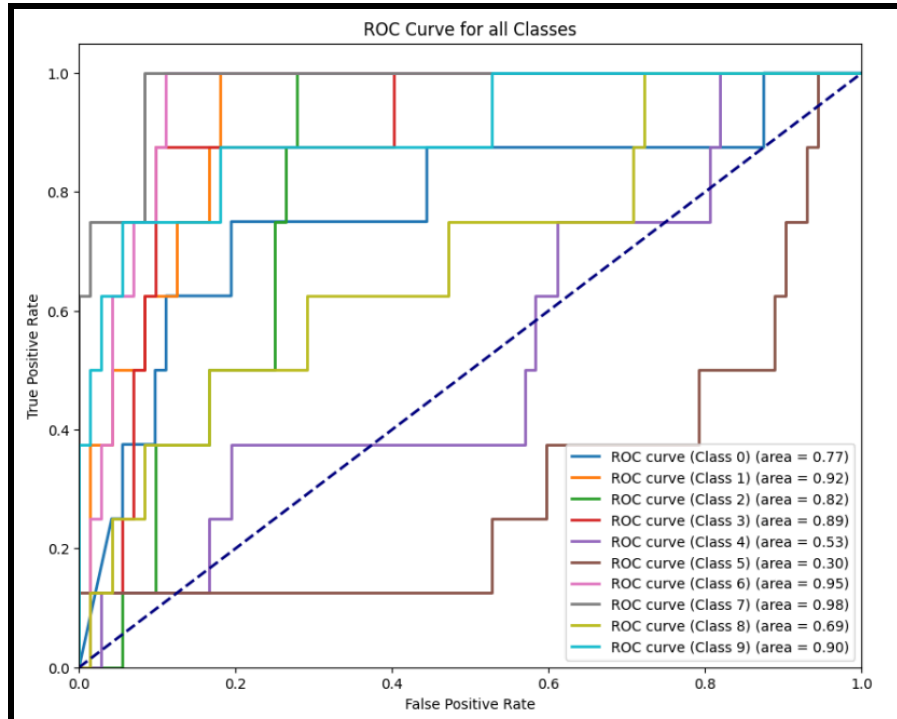
Overall Accuracy: 41.25

Overall F1-Score: 0.38561737969632703

Class-wise Metrics:

Class 0: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.7725694444444444
Class 1: Accuracy=0.25, F1-Score=0.13333333333333333, AUC-ROC=0.9201388888888888
Class 2: Accuracy=0.5, F1-Score=0.13333333333333333, AUC-ROC=0.8177083333333334
Class 3: Accuracy=0.375, F1-Score=0.0909090909090909, AUC-ROC=0.8888888888888889
Class 4: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5277777777777777
Class 5: Accuracy=0.125, F1-Score=0.07407407407407407, AUC-ROC=0.3020833333333333
Class 6: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.9479166666666666
Class 7: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.9774305555555556
Class 8: Accuracy=0.25, F1-Score=0.1, AUC-ROC=0.6875
Class 9: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.8993055555555556

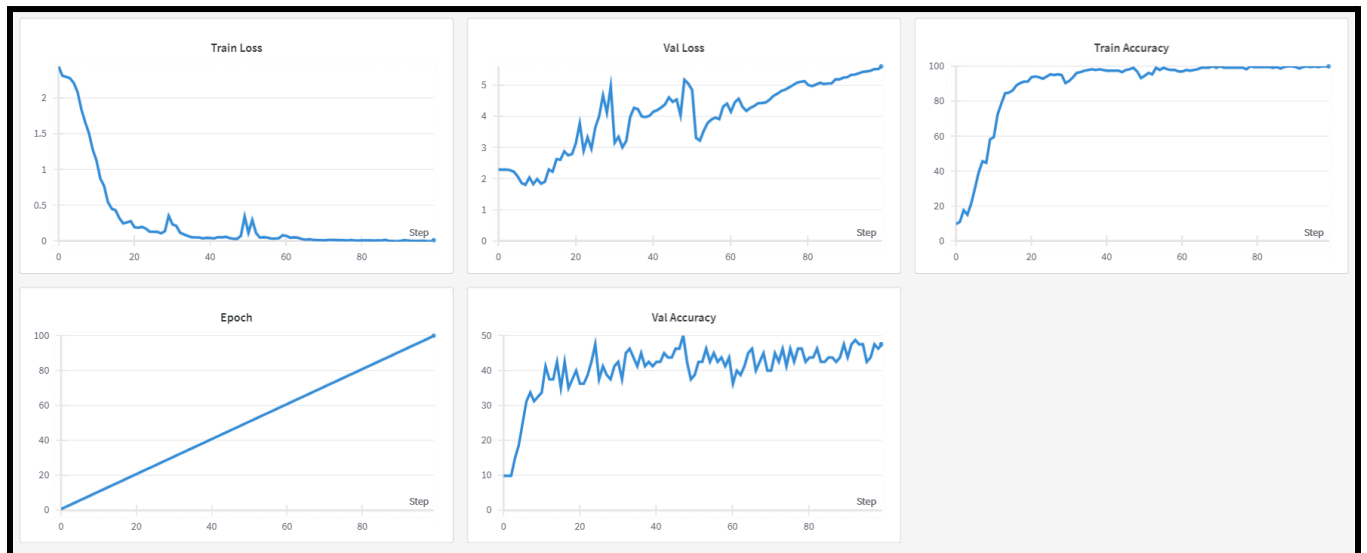
The model seems to overfit since it achieves 99% train accuracy and only around 40% validation accuracy even after taking measures like using dropout. This may occur due to small dataset or complex model architecture.



From the confusion matrix we can observe that the classes [0,2,6,7,9] are classified quite correctly while the model quite struggles with classes like [1,3,4,5,8]. Same inference can be made by looking at the auc-roc curve.

Task 2:- Perform k-fold validation, for k=4.

- For Fold 2:



Train Accuracy:100.0 Train Loss:0.0103 Val Accuracy:47.5 Val Loss:5.6

- For Fold 3:



Train Accuracy:99.58 Train Loss:0.0307 Val Accuracy:35.0 Val Loss:7.49677

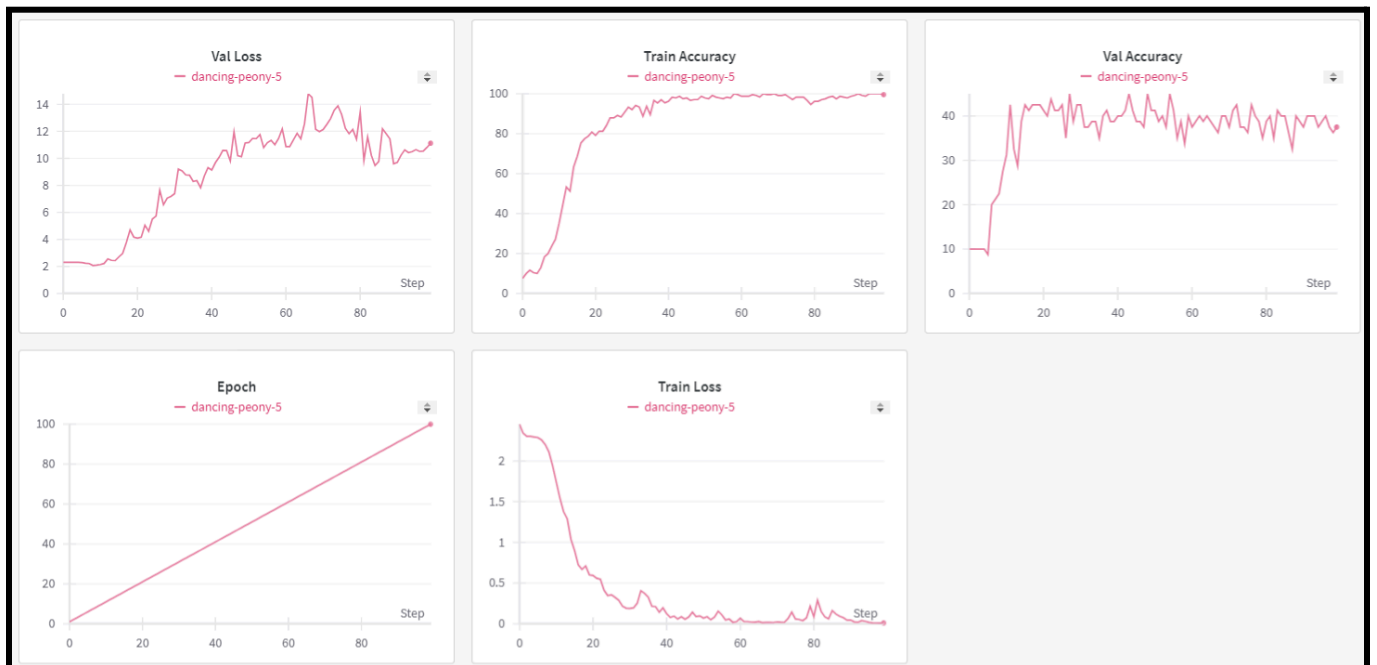
Model Overfitting is easily noticeable. The sharp zig-zag pattern of val accuracy and shows that after a certain epoch the model is stuck in local minima.

- For Fold 4:



Train Accuracy:97.9 Train Loss:0.064 Val Accuracy:41.25 Val Loss:5.26

- For Fold 5:

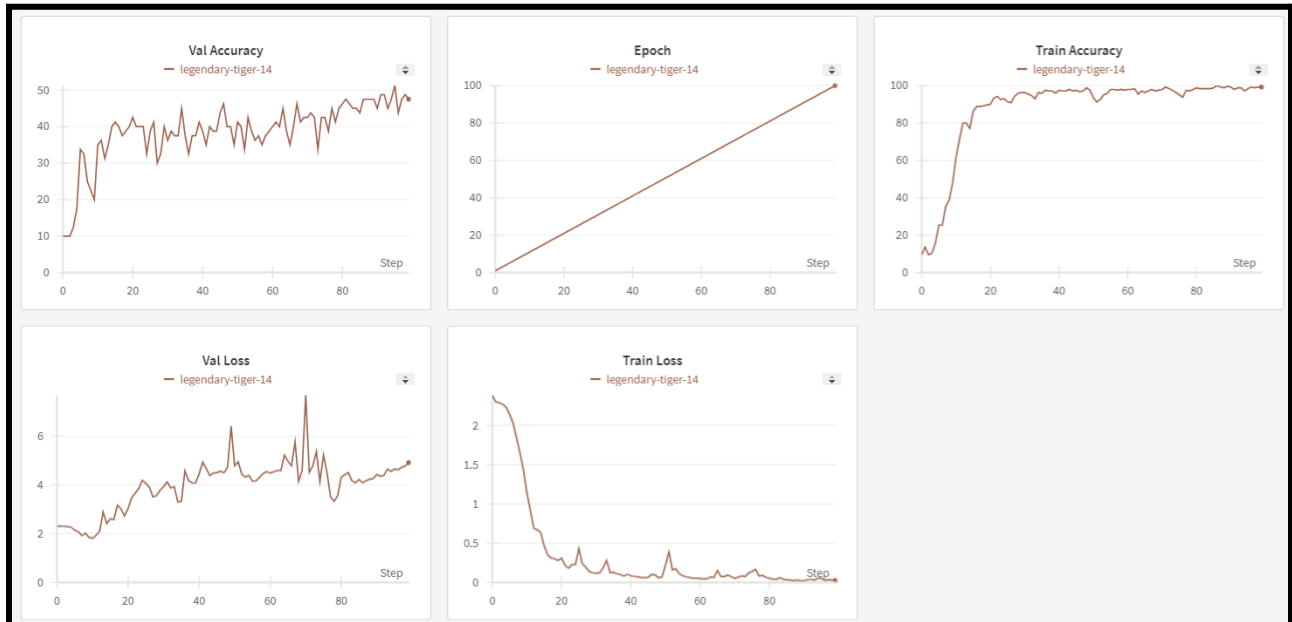


Train Accuracy:99.583 Train Loss:0.0106 Val Accuracy:37.5 Val Loss:11.21

Average accuracy over 4 folds: 40.3125

Task 3 : - Prepare Metrics on Test set

- Combination 1: - Learning rate - 0.001 , batch_size = 32



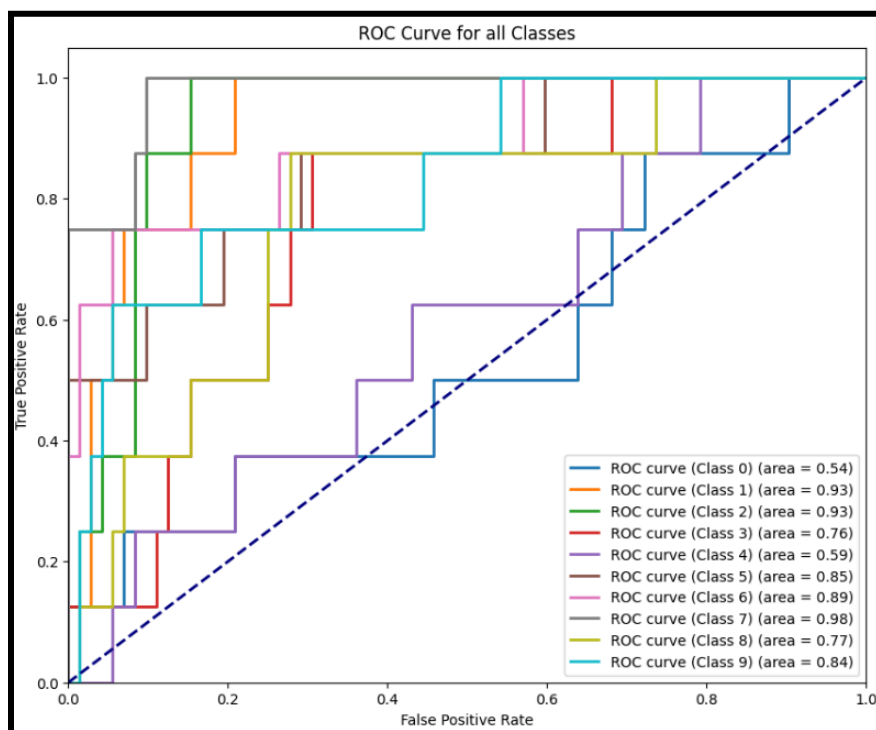
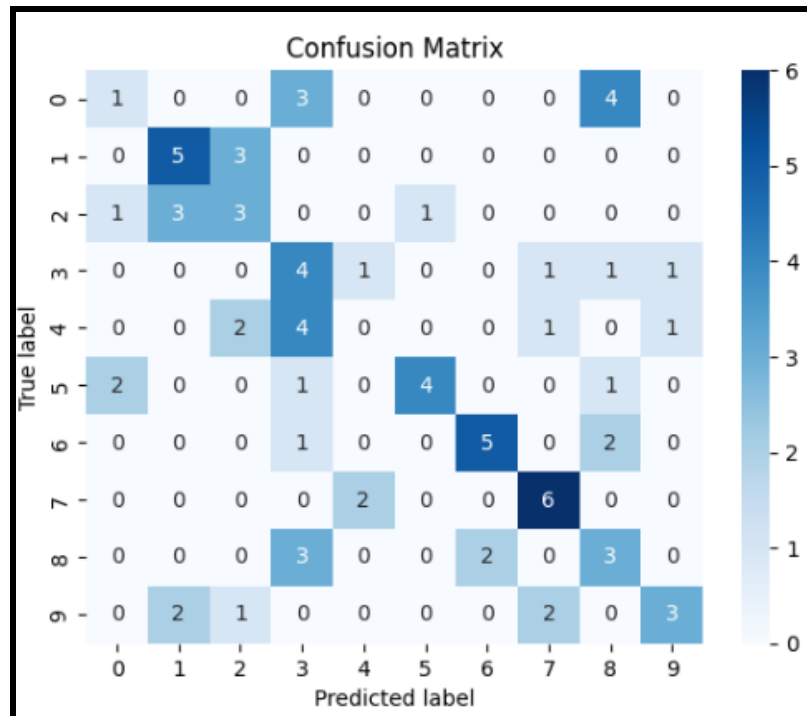
Overall Accuracy: 42.5

Overall F1-Score: 0.4134542615966764

Class-wise Metrics:

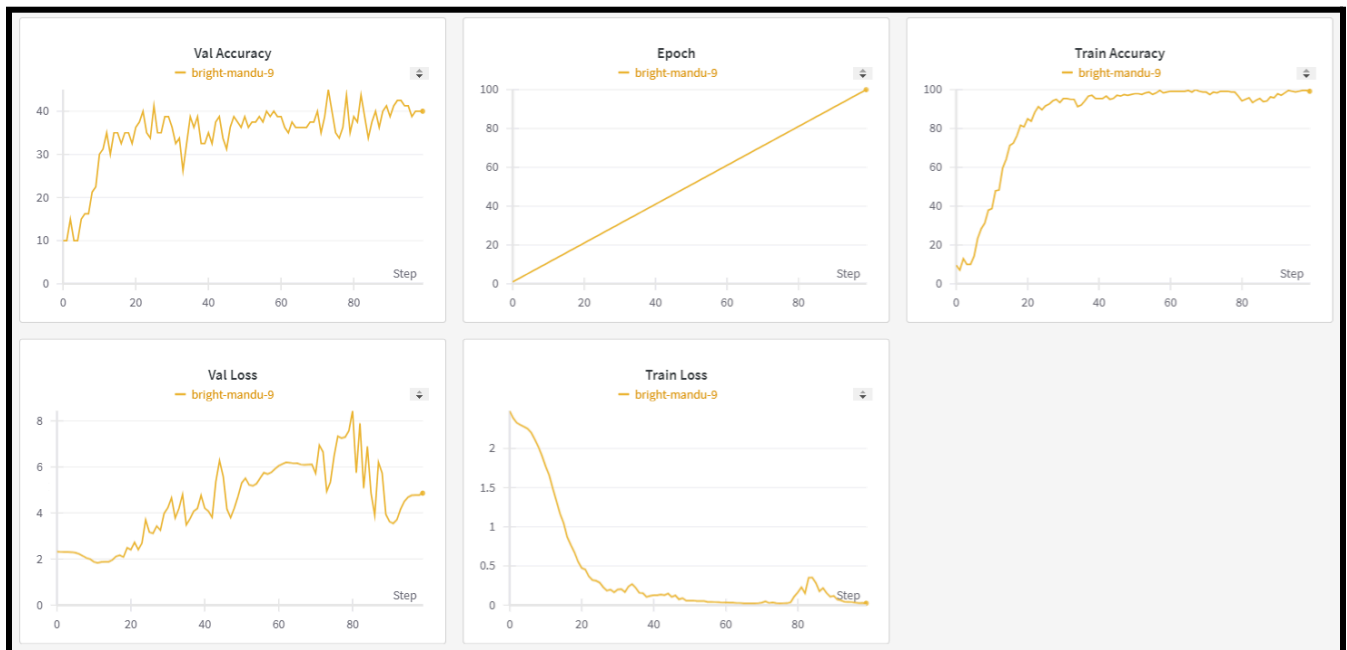
```
Class 0: Accuracy=0.125, F1-Score=0.07407407407407407, AUC-ROC=0.5399305555555555
Class 1: Accuracy=0.625, F1-Score=0.38461538461538464, AUC-ROC=0.9270833333333333
Class 2: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.9305555555555556
Class 3: Accuracy=0.5, F1-Score=0.13333333333333333, AUC-ROC=0.7621527777777778
Class 4: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5920138888888888
Class 5: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.8524305555555556
Class 6: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.8854166666666667
Class 7: Accuracy=0.75, F1-Score=0.42857142857142855, AUC-ROC=0.9774305555555556
Class 8: Accuracy=0.375, F1-Score=0.1818181818181818, AUC-ROC=0.7743055555555556
Class 9: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.8368055555555556
```

With Learning Rate 0.001 and Batch Size 32 , the achieved test accuracy is 42.5 and an overall F1- score is 0.4134 . The model Class-wise accuracy is also pretty okay , almost every class is classified correctly with more than 37.5% accuracy. Around 0.8 auc-roc value for each class signifies the model has a high discriminative power and is better at distinguishing between the positive and negative classes.



Looking at the AUC-ROC curve and the confusion matrix we can notice that the model quite struggles with classes like [0,4,8], while the model perfectly classifies classes 1,7 with auc-roc values 0.93, 0.98.

- Combination 2: - Learning rate - 0.001 , batch_size = 64



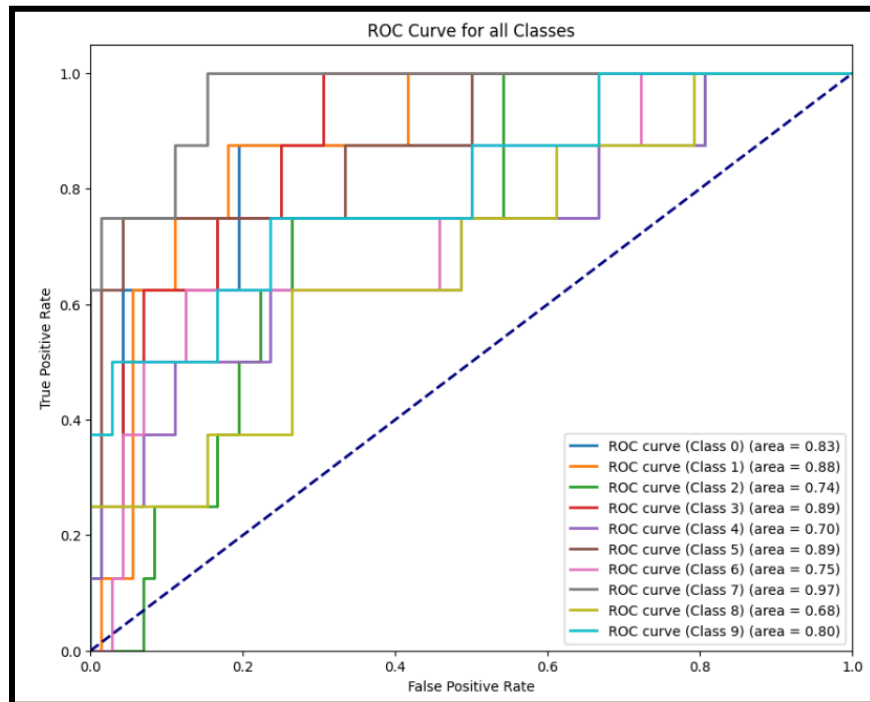
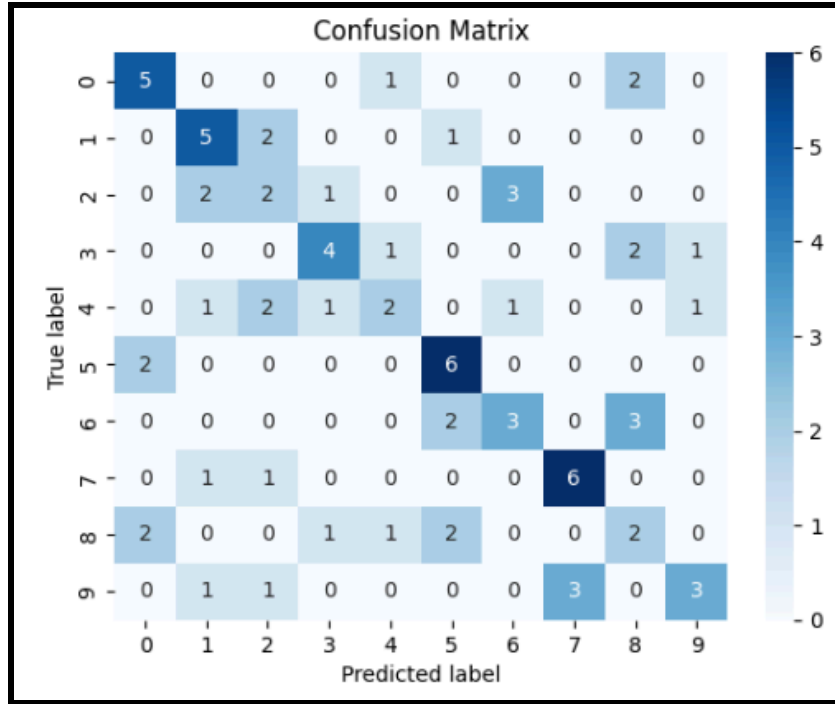
Overall Accuracy: 47.5

Overall F1-Score: 0.4669110370193962

Class-wise Metrics:

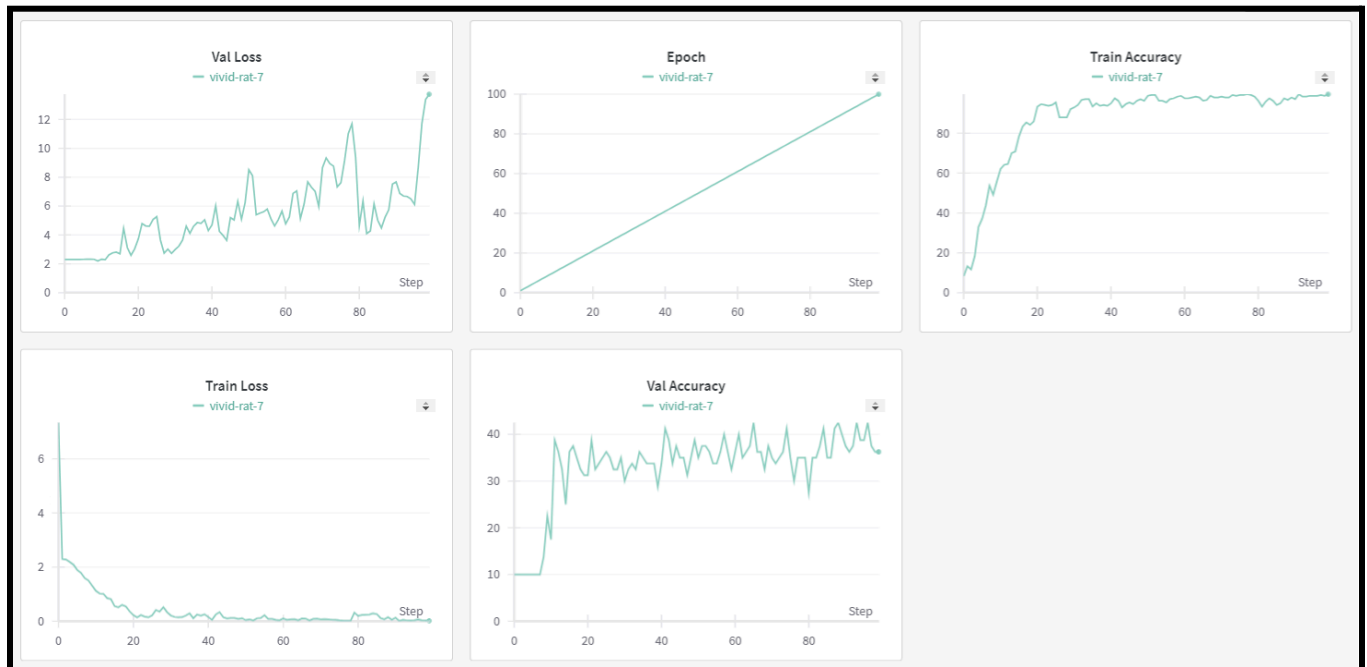
Class 0: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.8333333333333335
 Class 1: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.8819444444444444
 Class 2: Accuracy=0.25, F1-Score=0.1, AUC-ROC=0.7395833333333334
 Class 3: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.890625
 Class 4: Accuracy=0.25, F1-Score=0.06666666666666667, AUC-ROC=0.7013888888888888
 Class 5: Accuracy=0.75, F1-Score=0.42857142857142855, AUC-ROC=0.8854166666666666
 Class 6: Accuracy=0.375, F1-Score=0.1818181818181818, AUC-ROC=0.7517361111111111
 Class 7: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.9652777777777778
 Class 8: Accuracy=0.25, F1-Score=0.08, AUC-ROC=0.6788194444444444
 Class 9: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.8003472222222222

With Learning Rate 0.001 and Batch Size 64 , the achieved test accuracy is 47.5 and an overall F1- score is 0.4669 . The model Class-wise accuracy is also pretty good , almost every class is classified correctly with more than 50% accuracy. Around 0.8 auc-roc value for each class signifies the model has a high discriminative power and is better at distinguishing between the positive and negative classes. The model is overfitting but is better than other cnn models.



Looking at the AUC-ROC curve and the confusion matrix we can notice that the model quite struggles with classes like [2,4,6,8], while the model perfectly classifies classes 3,5,7 with auc-roc value of 0.89,0.89 and 0.97.

- Combination 3 : With Learning Rate 0.01 and Batch_size : 32



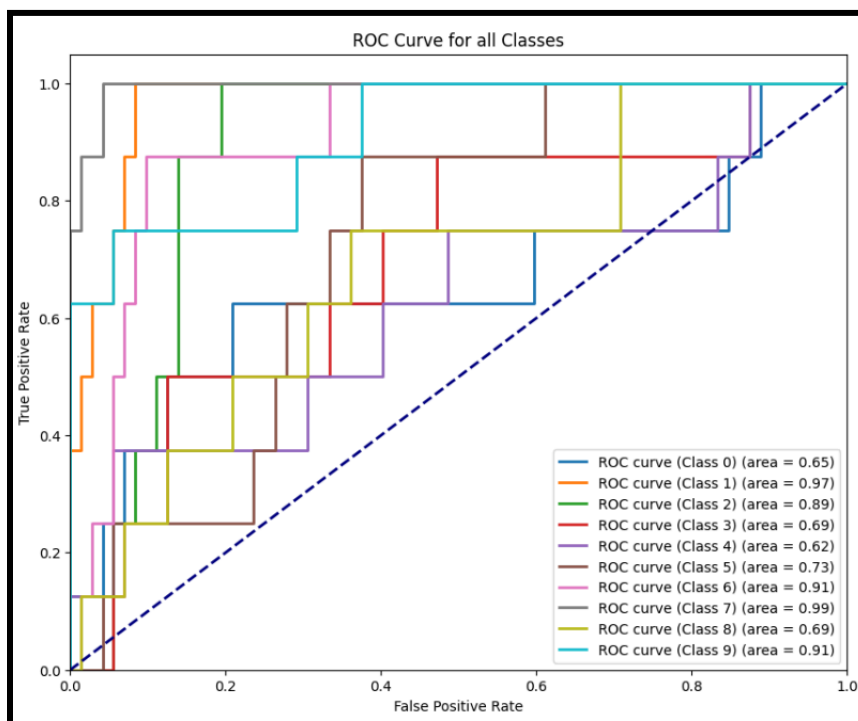
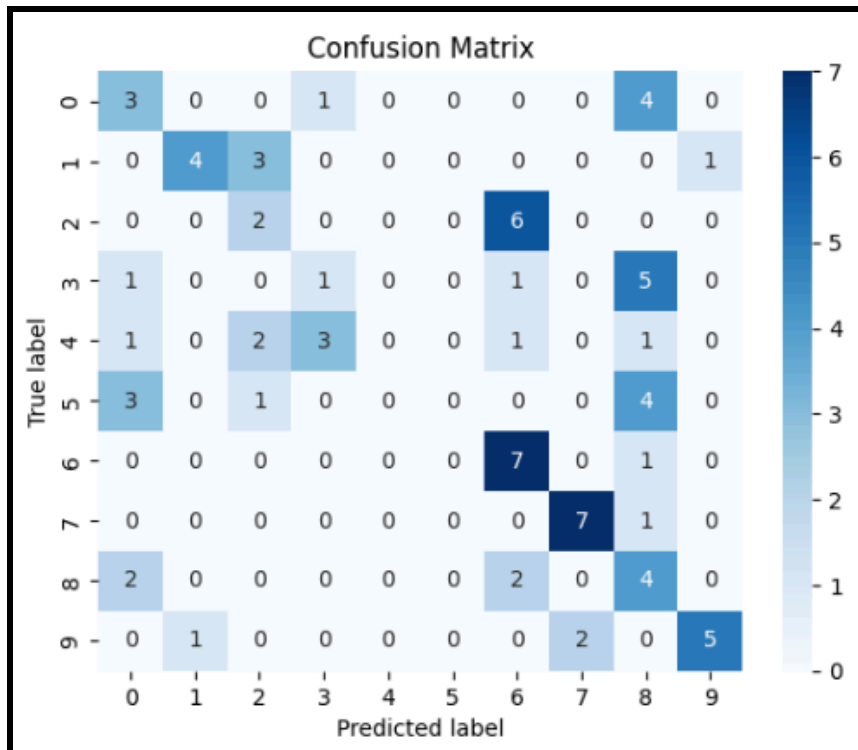
Overall Accuracy: 41.25

Overall F1-Score: 0.3736093514328809

Class-wise Metrics:

Class 0: Accuracy=0.375, F1-Score=0.1818181818181818, AUC-ROC=0.6527777777777778
 Class 1: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.96875
 Class 2: Accuracy=0.25, F1-Score=0.2, AUC-ROC=0.8854166666666666
 Class 3: Accuracy=0.125, F1-Score=0.05555555555555555, AUC-ROC=0.6927083333333333
 Class 4: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.6232638888888888
 Class 5: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.7256944444444444
 Class 6: Accuracy=0.875, F1-Score=0.4666666666666667, AUC-ROC=0.9079861111111111
 Class 7: Accuracy=0.875, F1-Score=0.4666666666666667, AUC-ROC=0.9930555555555556
 Class 8: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.6875
 Class 9: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9097222222222222

With Learning Rate 0.01 and Batch Size 32 , the achieved test accuracy is 41.25 and an overall F1- score is 0.373 . The model Class-wise accuracy is pretty bad , almost every class is classified correctly with less than 50% accuracy. Around 0.5 auc-roc value for each class signifies the model has a low discriminative power and is inaccurate at distinguishing between the positive and negative classes. Even classes 4 and 5 achieved 0 accuracy.



Looking at the AUC-ROC curve and the confusion matrix we can notice that the model is incapable at classifying classes like [4,5], while the model perfectly classifies classes [1,7] with auc-roc value of 0.97, and 0.99. Not a good model.

- Combination 4 : With Learning Rate 0.01 and Batch_size : 64

```
Overall Accuracy: 10.0  
Overall F1-Score: 0.01818181818181818  
Class-wise Metrics:  
Class 0: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 1: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 2: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 3: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 4: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 5: Accuracy=1.0, F1-Score=1.0, AUC-ROC=0.5  
Class 6: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 7: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 8: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5  
Class 9: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.5
```

The model didn't learn anything , maybe because of the high learning rate and large batch size.

- Larger batch sizes can lead to more noise in the gradient estimates due to fewer updates per epoch. This increased noise can cause the optimization process to become unstable or less effective, especially with a higher learning rate.
- With a high learning rate the optimization algorithm (e.g., stochastic gradient descent, Adam) takes large steps in the parameter space during optimization. This can lead to instability and overshooting of the optimal solution.

Task 5:- Perform hyper-parameter tuning

As mentioned above , with learning rate 0.01 we didn't receive any significant accuracy . So the best hyperparameters based on validation dataset is :-

Best Hyperparameters :- learning_rate : 0.001, batch_size : 64
Best Validation Accuracy : 47.5

Architecture 2:- Transformer Encoder with CNN Base

1. Position Embedding Function (position_embedding):

- This function generates positional encodings for input sequences in a transformer model.
- It uses a formula to calculate sinusoidal positional embeddings based on position and embedding dimensions.
- The result is a tensor of positional encodings.

2. Layer Normalization Class (LayerNormalization):

- This class implements layer normalization, which normalizes the activations of a layer across the feature dimension.
- It contains learnable parameters (gamma and beta) for scaling and shifting the normalized values.

3. Scaled Dot Product Attention Function (scaled_dot_product):

- This function performs scaled dot-product attention, a key component of the transformer architecture.
- It computes attention scores between query and key vectors, scales them, applies a softmax function, and computes a weighted sum of value vectors.

4. Multi-Head Attention Class (MultiHeadAttention):

- This class implements multi-head attention, which splits the input into multiple heads and performs attention independently in each head.
- It uses linear transformations to project the input into query, key, and value vectors for each head, then applies the scaled dot product attention.

5. Positionwise FeedForward Class

(PositionwiseFeedForward) :

- This class defines a position-wise feedforward network, which applies linear transformations followed by a non-linear activation function (ReLU) independently at each position in the sequence.

6. Linear Layer Class (LinearLayer) :

- This class defines a simple linear layer with a weight matrix and bias vector.

7. Attention Block Class (AttentionBlock) :

- This class implements an attention block, which combines multi-head attention with layer normalization and dropout.

8. FeedForward Block Class (FeedForwardBlock) :

- This class implements a feedforward block, which combines a position-wise feedforward network with layer normalization and dropout.

9. Transformer Encoder Block Class

(TransformerEncoderBlock) :

- This class defines a transformer encoder block, which consists of two attention blocks and a feedforward block.

10. Transformer Encoder Class (TransformerEncoder) :

- This class implements a transformer encoder, which consists of multiple transformer encoder blocks and a final linear layer for classification.

11. Combined Model Class (CombinedModel):

- This class combines a convolutional neural network (CNN) model and a transformer encoder model for joint training and inference.

Transformer Encoder with CNN Base Architecture :

Layer (type:depth-idx)	Output Shape	Param #
CombinedModel	[32, 10]	--
└─TransConv1DNet: 1-1	[32, 16, 2250]	--
└─ConvBlock: 2-1	[32, 64, 35999]	--
└─Conv1d: 3-1	[32, 64, 71999]	512
└─ReLU: 3-2	[32, 64, 71999]	--
└─MaxPool1d: 3-3	[32, 64, 35999]	--
└─ConvBlock: 2-2	[32, 32, 9000]	--
└─Conv1d: 3-4	[32, 32, 18000]	10,272
└─ReLU: 3-5	[32, 32, 18000]	--
└─MaxPool1d: 3-6	[32, 32, 9000]	--
└─ConvBlock: 2-3	[32, 16, 2250]	--
└─Conv1d: 3-7	[32, 16, 4501]	1,552
└─ReLU: 3-8	[32, 16, 4501]	--
└─MaxPool1d: 3-9	[32, 16, 2250]	--
└─TransformerEncoder: 1-2	[32, 10]	16
└─TransformerEncoderBlock: 2-4	[32, 2251, 16]	--
└─AttentionBlock: 3-10	[32, 2251, 16]	1,120
└─AttentionBlock: 3-11	[32, 2251, 16]	1,120
└─AttentionBlock: 3-12	[32, 2251, 16]	1,120
└─AttentionBlock: 3-13	[32, 2251, 16]	1,120
└─FeedForwardBlock: 3-14	[32, 2251, 16]	33,840
└─Linear: 2-5	[32, 10]	170
Total params: 50,842		
Trainable params: 50,842		
Non-trainable params: 0		
Total mult-adds (G): 7.32		
Input size (MB): 18.43		
Forward/backward pass size (MB): 2138.45		
Params size (MB): 0.20		
Estimated Total Size (MB): 2157.09		

Task 4:- Report total trainable and Non-trainable parameters :

Total Trainable parameters : 50,842

Total Non-Trainable parameters : 0

As can be seen from the above image , very few parameters were there for the Attention Blocks , most of the parameters belonged to the feed-forward network.

Task 1:- Train for 100 epochs : Test Fold : 1 Validation Fold : 2

Fixed Params : Learning Rate : 0.001 , Batch Size = 32 , d_model = 16 ,
hidden_dim = 1024 , dropout_prob = 0.1 , max_sequence_length = 2251

1. Num_Heads = 1



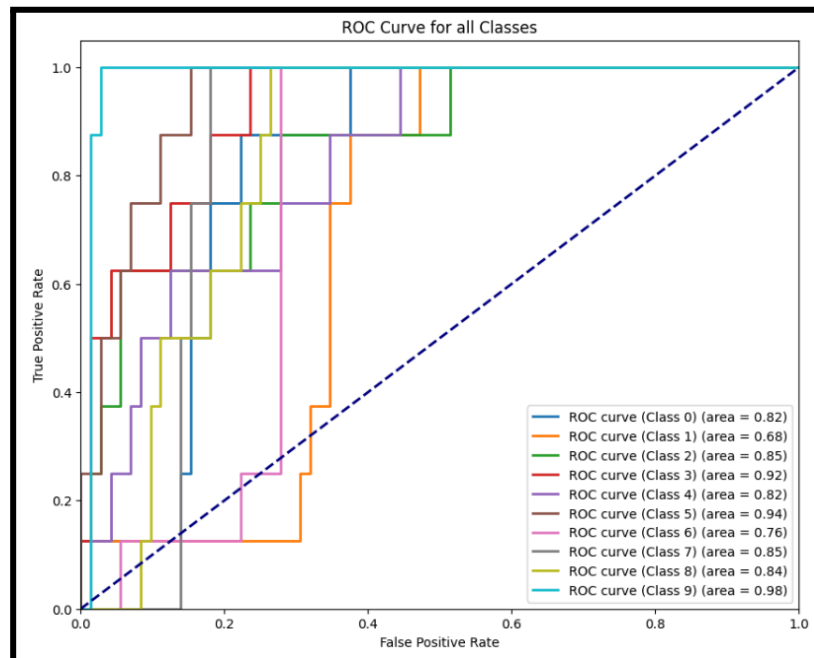
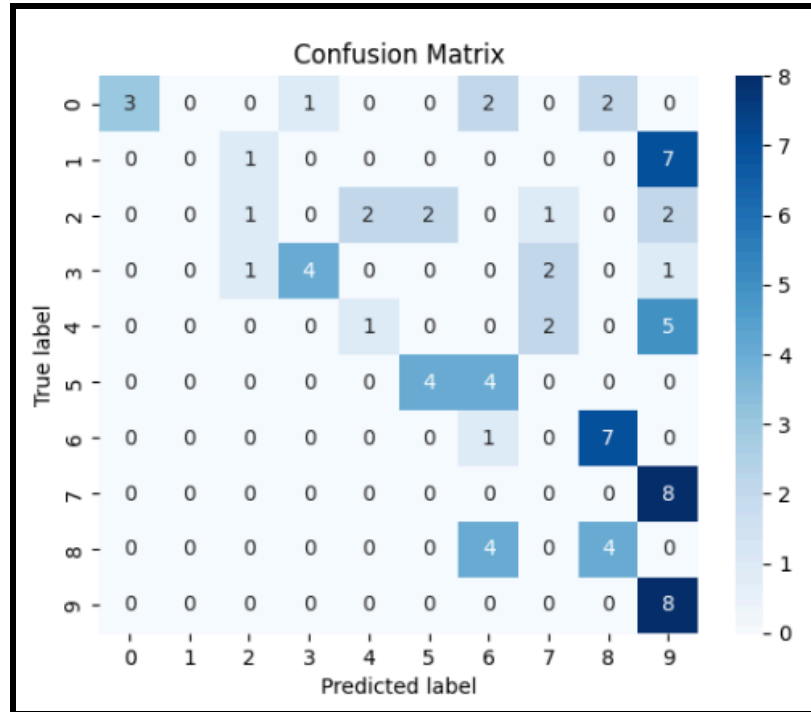
Overall Accuracy: 32.5

Overall F1-Score: 0.2992376045007624

Class-wise Metrics:

```
Class 0: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.8246527777777778
Class 1: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.6840277777777778
Class 2: Accuracy=0.125, F1-Score=0.04444444444444444, AUC-ROC=0.8506944444444444
Class 3: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.921875
Class 4: Accuracy=0.125, F1-Score=0.07407407407407407, AUC-ROC=0.8246527777777778
Class 5: Accuracy=0.5, F1-Score=0.3333333333333333, AUC-ROC=0.9444444444444444
Class 6: Accuracy=0.125, F1-Score=0.11111111111111111, AUC-ROC=0.7569444444444445
Class 7: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.8472222222222222
Class 8: Accuracy=0.5, F1-Score=0.3333333333333333, AUC-ROC=0.8368055555555557
Class 9: Accuracy=1.0, F1-Score=1.0, AUC-ROC=0.984375
```

The model performs poorly , even the CNN base performed better with 42.5 % accuracy . This may be due to the fact that the number of heads is only 1. The F1-Score is also only 0.3 which is very less . Some classes like 1,7 were not predicted and some classes like 9 were correctly classified with accuracy 100%. Also it can be seen that the model is not overfitting here i.e. cnn with encoder.



From the confusion matrix and auc-roc curve we can see that model was incapable of classifying classes like 1 and 7 but it did pretty well at classifying classes like 3,5 and 9 with auc-roc score of 0.92,0.94 and 0.98 which is very good. But the model as a whole performs very bad.

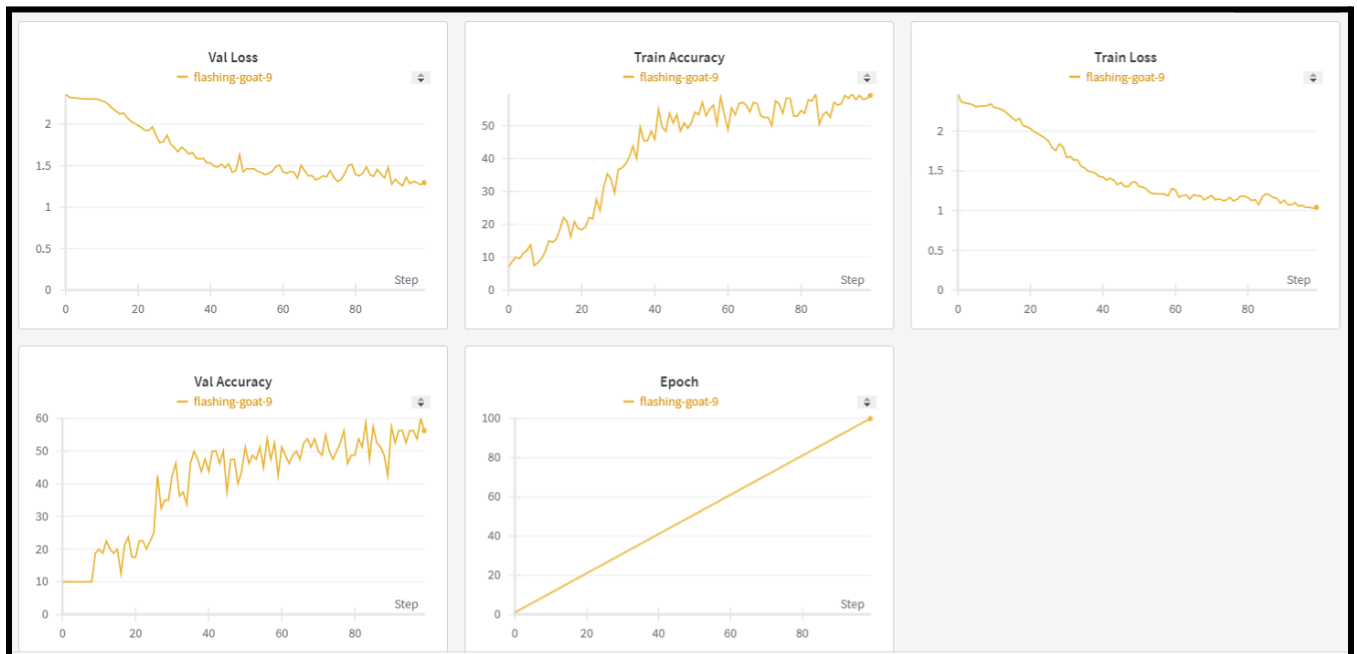
2. Num_Heads = 2 :

Overall Accuracy: 55.00000000000001

Overall F1-Score: 0.5233484848484848

Class-wise Metrics:

Class 0: Accuracy=0.375, F1-Score=0.18181818181818, AUC-ROC=0.8975694444444444
Class 1: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.9184027777777777
Class 2: Accuracy=0.125, F1-Score=0.04444444444444444, AUC-ROC=0.8611111111111112
Class 3: Accuracy=0.625, F1-Score=0.38461538461538464, AUC-ROC=0.9635416666666666
Class 4: Accuracy=0.375, F1-Score=0.1818181818181818, AUC-ROC=0.9149305555555555
Class 5: Accuracy=0.875, F1-Score=0.4666666666666667, AUC-ROC=0.9583333333333334
Class 6: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9166666666666667
Class 7: Accuracy=0.75, F1-Score=0.42857142857142855, AUC-ROC=0.9357638888888888
Class 8: Accuracy=0.125, F1-Score=0.07407407407407407, AUC-ROC=0.8993055555555556
Class 9: Accuracy=0.875, F1-Score=0.4666666666666667, AUC-ROC=0.9548611111111111

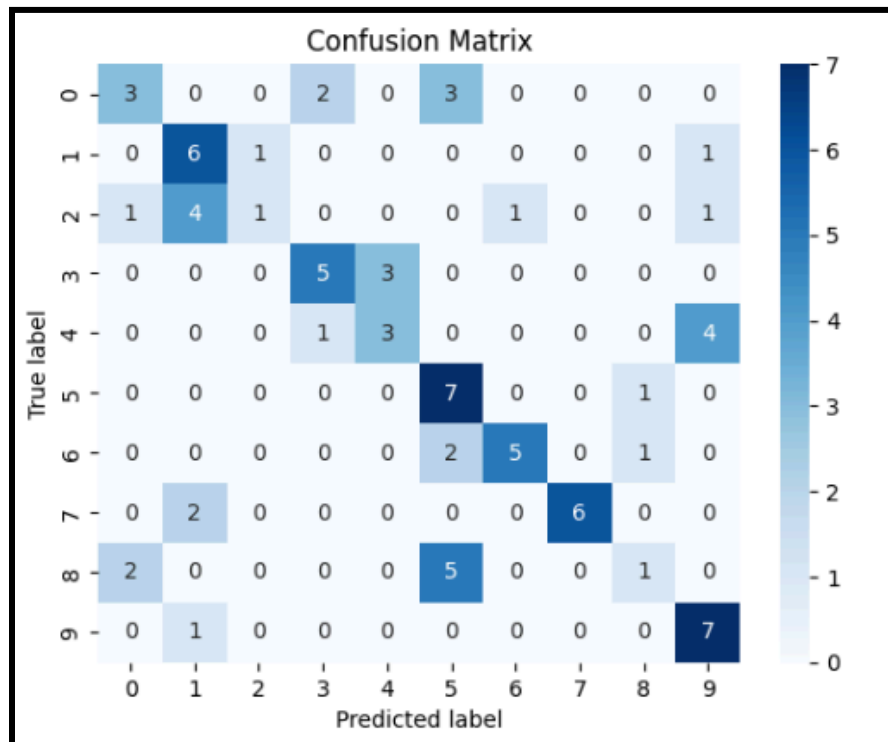
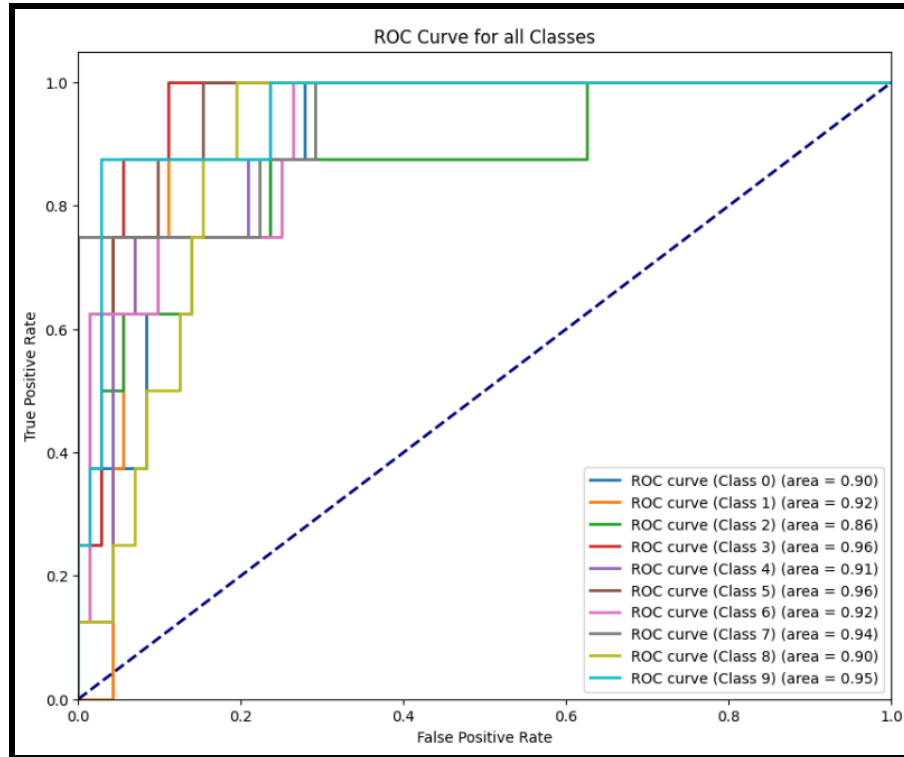


With the number of heads 2, the model performs pretty well .

The validation accuracy reaches around 56.25 % and the test accuracy is 55 % which shows that the model is quite in sync now and not overfitting.

The F1-Score is also only 0.52 which is better than the cnn base .

The model accuracy improved quite gradually , unlike CNN where the accuracy peaked at initial and then flattened. The significant difference in performance for the number of heads 1 and 2 is easily noticeable.



From the auc-roc curve and the confusion matrix it is evident that this model performs better than the previous one by classifying classes like 1 and 7 with great accuracy .

3. Num_Heads = 4

Overall Accuracy: 57.49999999999999

Overall F1-Score: 0.5719978354978356

Class-wise Metrics:

Class 0: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.90625

Class 1: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.8975694444444445

Class 2: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.9097222222222222

Class 3: Accuracy=0.625, F1-Score=0.19230769230769232, AUC-ROC=0.9548611111111111

Class 4: Accuracy=0.375, F1-Score=0.1818181818181818, AUC-ROC=0.9583333333333333

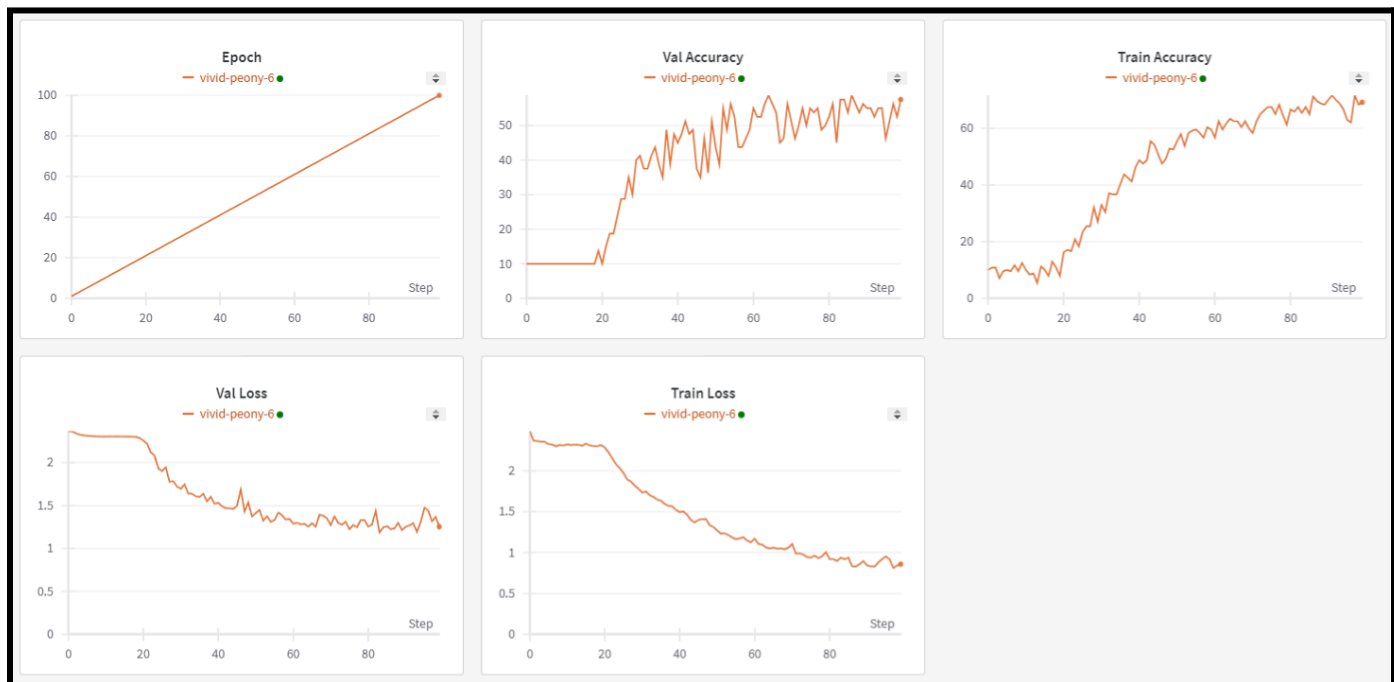
Class 5: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9704861111111112

Class 6: Accuracy=1.0, F1-Score=1.0, AUC-ROC=0.9722222222222222

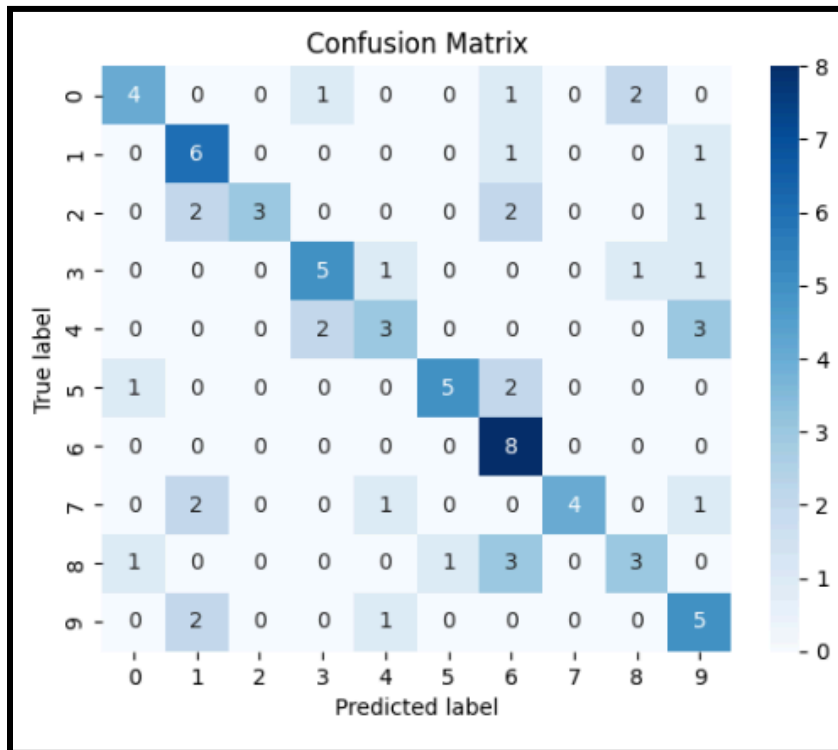
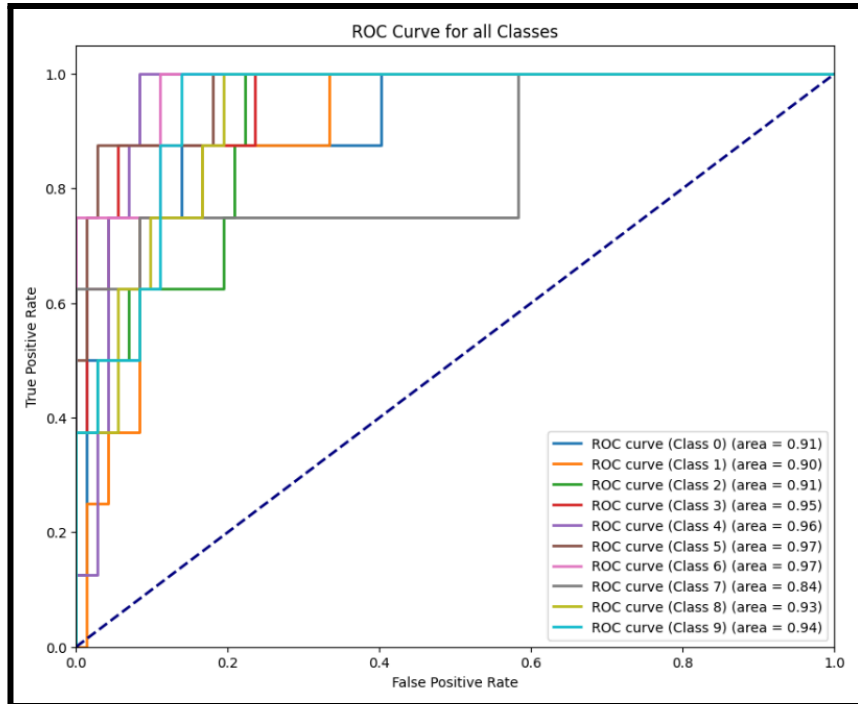
Class 7: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.84375

Class 8: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.9288194444444444

Class 9: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9409722222222223

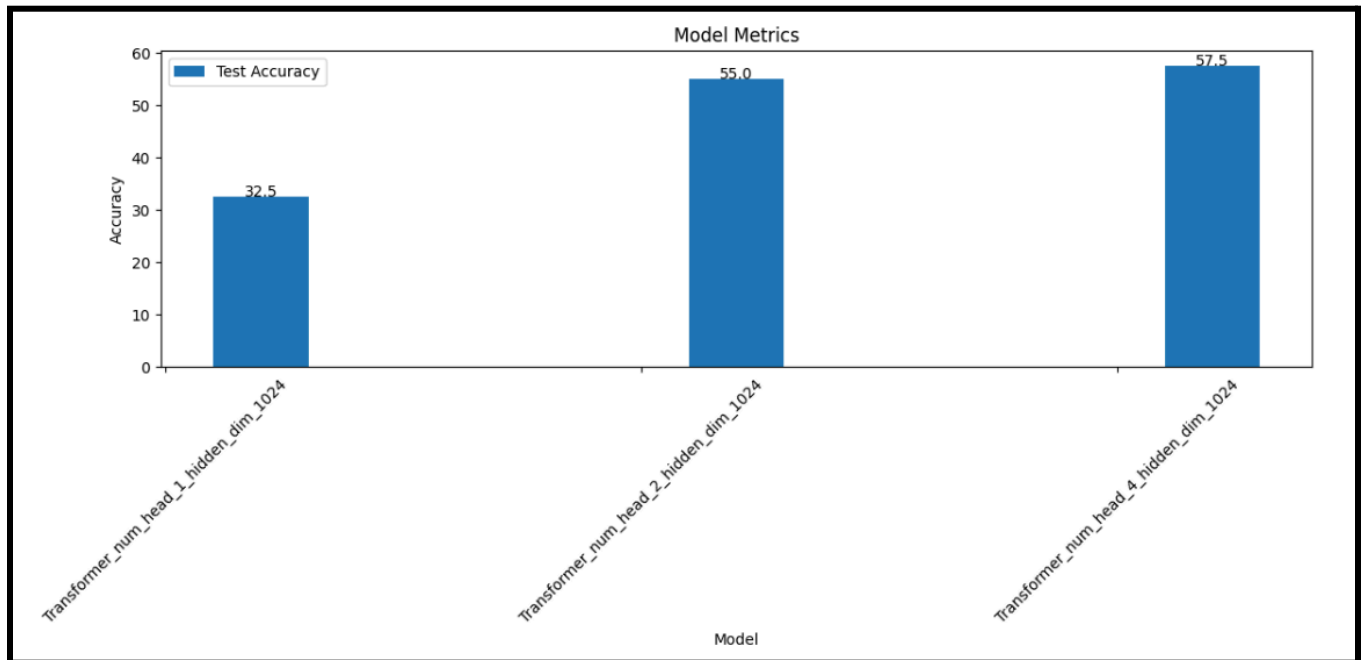


This model performs the best in comparison to the above two models . The model achieves a train accuracy of 69% , val accuracy of 57% and test accuracy of 57.5% . This significant performance can be directly related to the number of heads . As the number of heads are increasing we can see a significant increase in accuracy . The overall F1-score is 0.57 which is pretty great , the best so far.



From the auc-roc curve and the confusion matrix it is evident that this model performs way better than any previous model by classifying almost all the classes with great accuracy. AUC-ROC scores for all the above classes are above 0.9 and the misclassifications of classes are also very less.

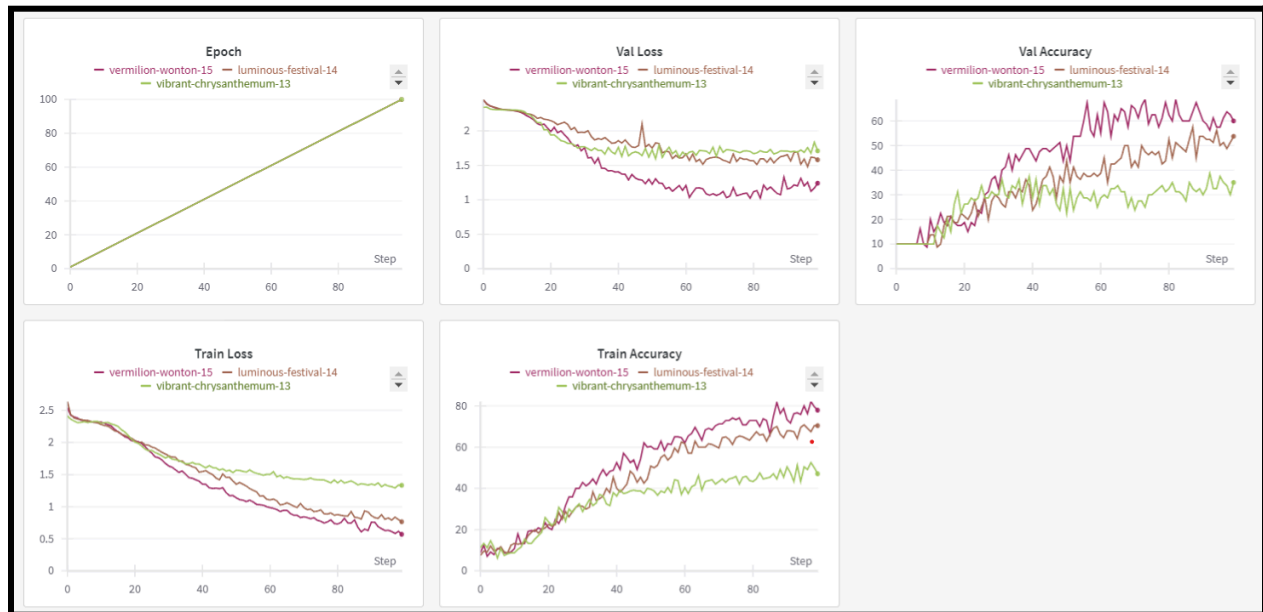
Analysis :-



- Among the three models, the one with 4 attention heads achieves the highest test accuracy of 57.5%
-
- Increasing the number of attention heads from 1 to 4 leads to a significant improvement in test accuracy.
- The hidden dimension size (1024) remains constant across all models, so the differences in performance are primarily due to the variation in the number of attention heads.
- The transformer model with 4 attention heads likely achieves better performance because it can capture more complex patterns and dependencies in the data compared to models with fewer attention heads.
- Increasing the number of attention heads allows the model to attend to more diverse aspects of the input sequence simultaneously, potentially enhancing its ability to learn and represent the underlying features in the data.

Task 2: Perform k-fold validation, for k=4.

Fold = 2 : Num_Head = 1,2,4



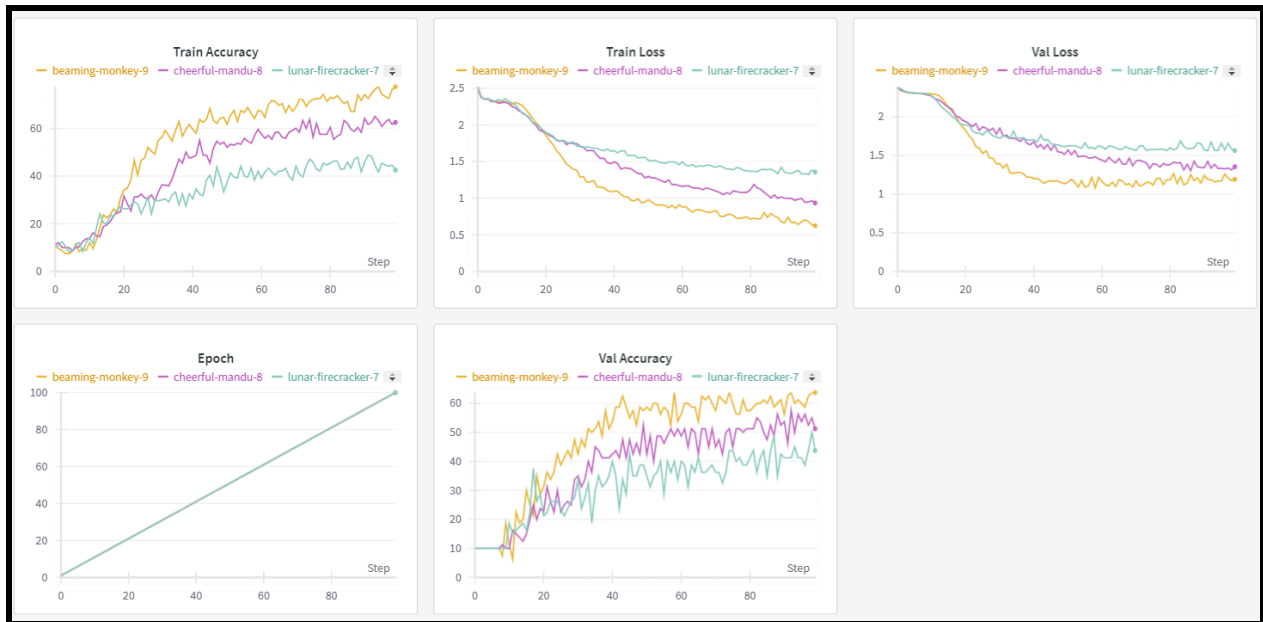
Num_Head 1: Train Accuracy:47	Train Loss:1.3	Val Accuracy:35	Val Loss:1.7
Num_Head 2: Train Accuracy:70	Train Loss:0.7	Val Accuracy:53	Val Loss:1.5
Num_Head 4: Train Accuracy:77	Train Loss:0.5	Val Accuracy:63	Val Loss:1.2

Fold = 3 : Num_Head = 1,2,4



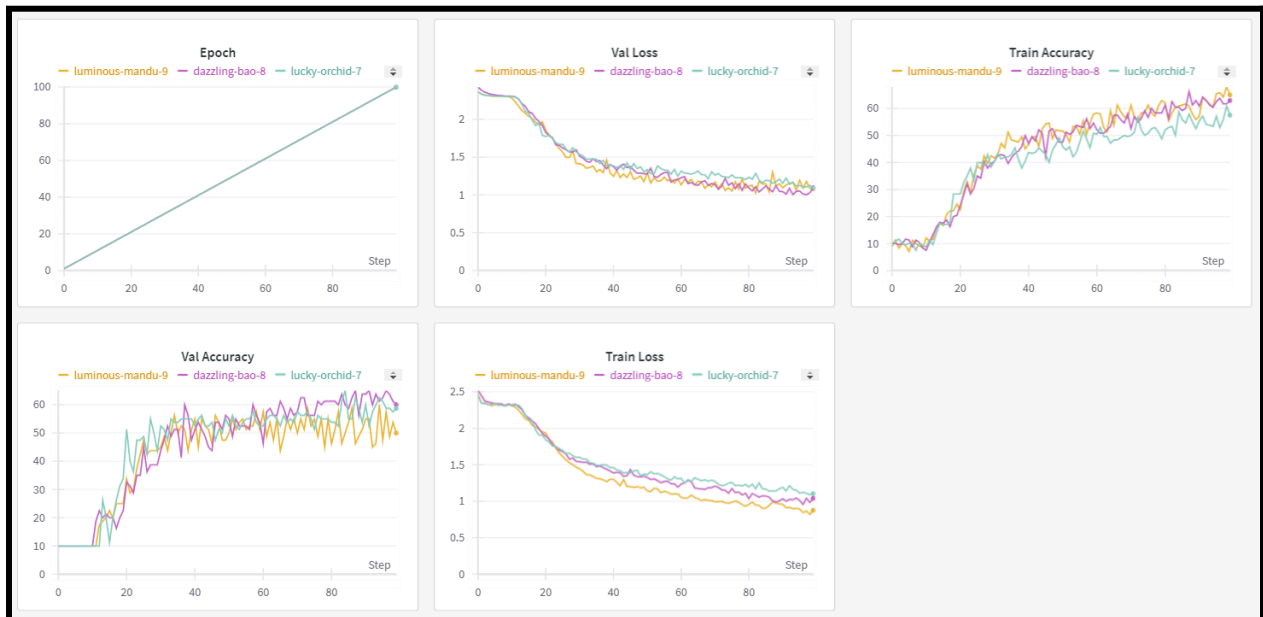
Num_Head 1: Train Accuracy:59	Train Loss:1.02	Val Accuracy:43	Val Loss:1.39
Num_Head 2: Train Accuracy:61	Train Loss:1.09	Val Accuracy:48	Val Loss:1.37
Num_Head 4: Train Accuracy:72	Train Loss:0.8	Val Accuracy:56	Val Loss:1.1

Fold = 4 : Num_Head = 1,2,4



Num_Head 1: Train Accuracy:42	Train Loss:1.3	Val Accuracy:42	Val Loss:1.5
Num_Head 2: Train Accuracy:62	Train Loss:0.9	Val Accuracy:51	Val Loss:1.3
Num_Head 4: Train Accuracy:77	Train Loss:0.6	Val Accuracy:63	Val Loss:1.1

Fold = 5 : Num_Head = 1,2,4



Num_Head 1: Train Accuracy:57	Train Loss:1.1	Val Accuracy:50	Val Loss:1.09
Num_Head 2: Train Accuracy:62	Train Loss:1.0	Val Accuracy:58	Val Loss:1.08
Num_Head 4: Train Accuracy:65	Train Loss:0.8	Val Accuracy:60	Val Loss:1.08

1. Average accuracy for Num_head 1 over 4 folds: 45.31
2. Average accuracy for Num_head 2 over 4 folds: 49.37
3. Average accuracy for Num_head 4 over 4 folds: 52.08

Task 5:- Perform hyper-parameter tuning

1. Num_Head = 1 , Hidden_dim = 512

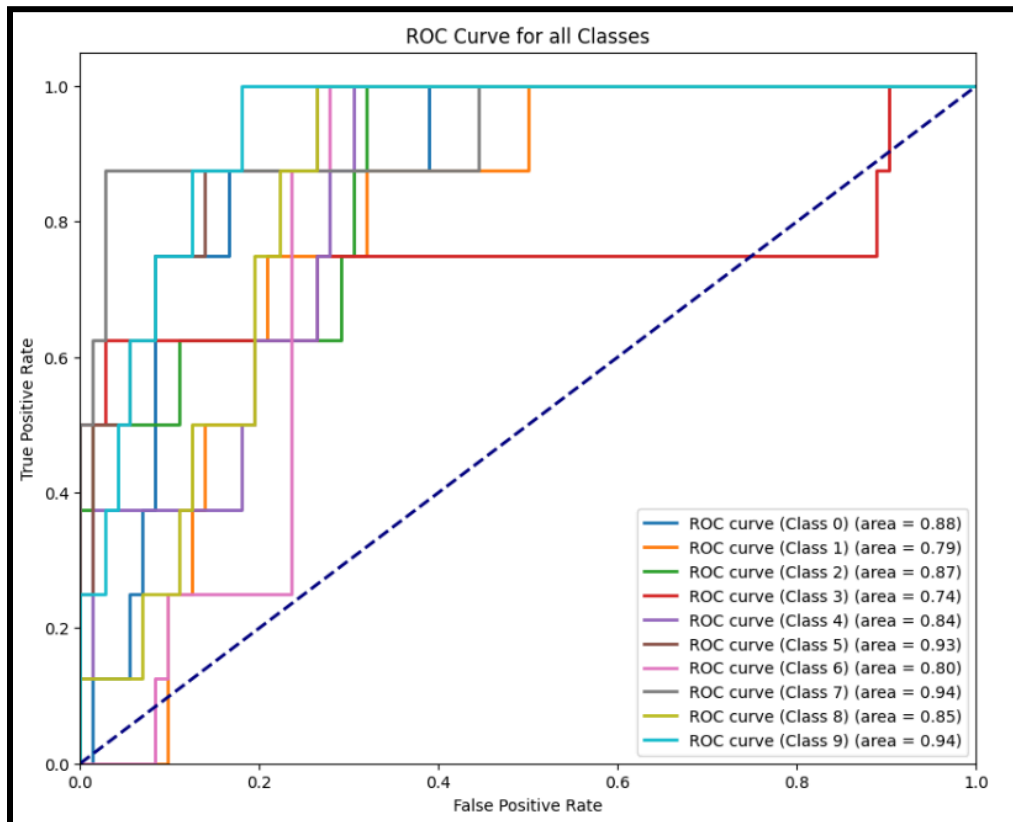
```

Overall Accuracy: 42.5

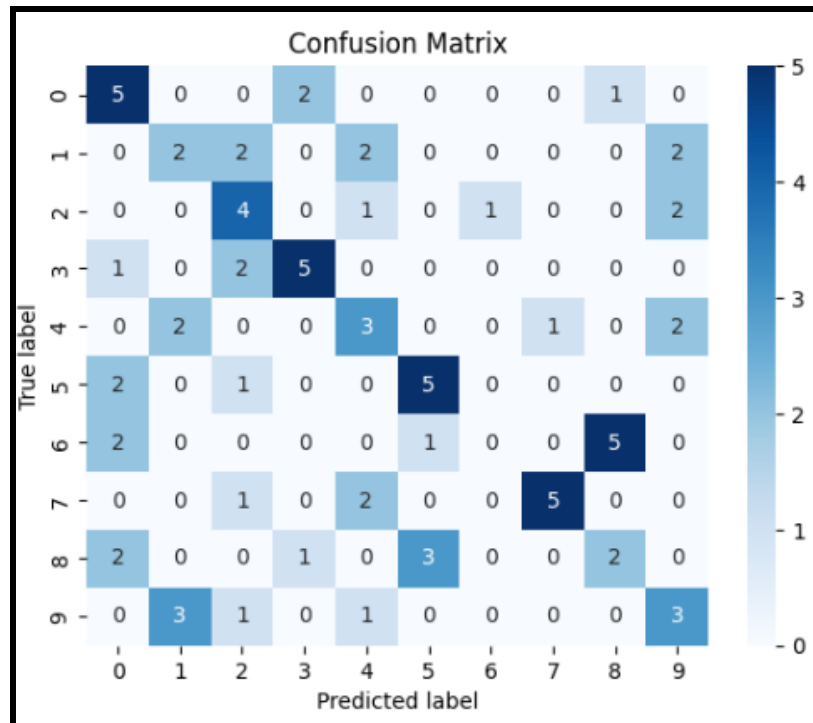
Overall F1-Score: 0.4071122659590151

Class-wise Metrics:
Class 0: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.8819444444444445
Class 1: Accuracy=0.25, F1-Score=0.1, AUC-ROC=0.7899305555555556
Class 2: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.8697916666666667
Class 3: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.7395833333333333
Class 4: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.84375
Class 5: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9288194444444445
Class 6: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.7951388888888888
Class 7: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9357638888888888
Class 8: Accuracy=0.25, F1-Score=0.1, AUC-ROC=0.8524305555555556
Class 9: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.9357638888888888

```



For Num_Head = 1 and hidden_dim = 512 , the model achieves test accuracy of 42.5 . Not much improvement with respect to CNN base model.



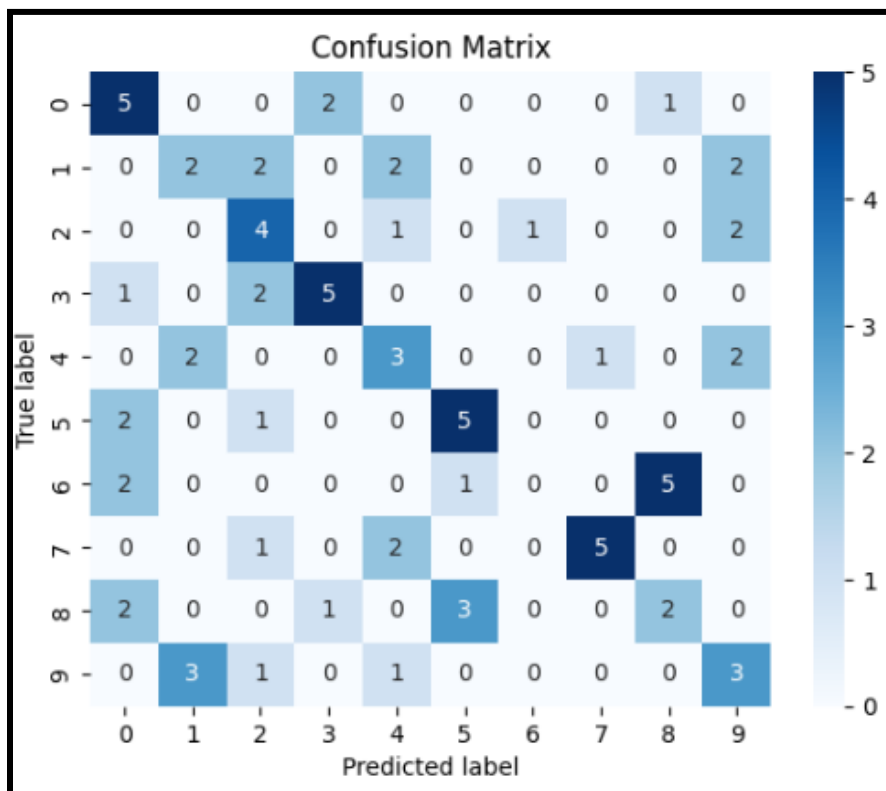
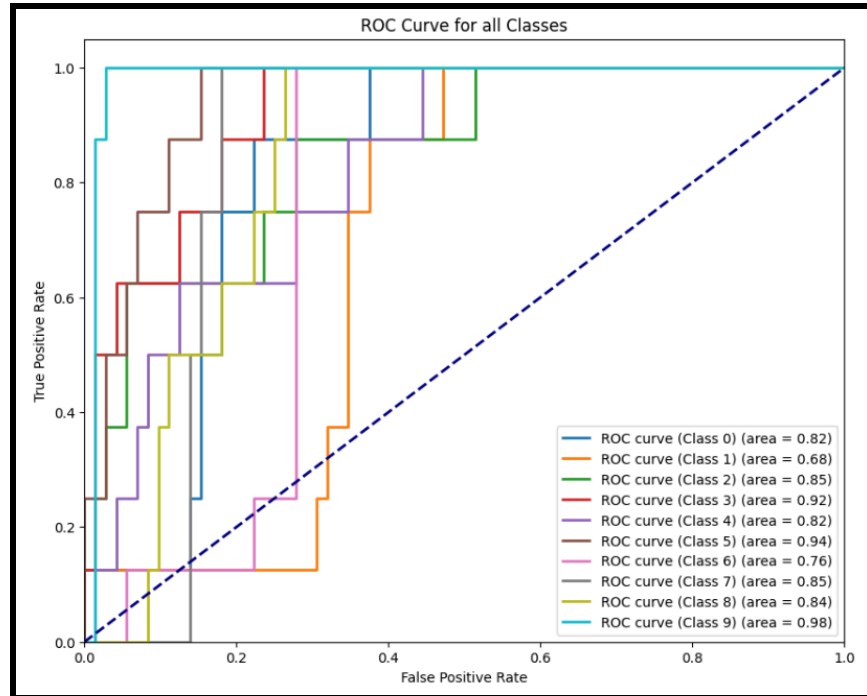
As observable from the above confusion matrix , the model is not able to classify the label 6 at all and all labels like 1,4 and 8 are highly misclassified . Leading to a low overall F1-score of only 0.4. Classes like 0 , 3 , 5 are quite correctly classified , similar inference can be drawn by looking at each class AUC_ROC curve.

2. Num_Head = 1 , Hidden_dim = 1024

```
Overall Accuracy: 32.5
Overall F1-Score: 0.2992376045007624

Class-wise Metrics:
Class 0: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.8246527777777778
Class 1: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.6840277777777778
Class 2: Accuracy=0.125, F1-Score=0.04444444444444444, AUC-ROC=0.8506944444444444
Class 3: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.921875
Class 4: Accuracy=0.125, F1-Score=0.07407407407407407, AUC-ROC=0.8246527777777778
Class 5: Accuracy=0.5, F1-Score=0.3333333333333333, AUC-ROC=0.9444444444444444
Class 6: Accuracy=0.125, F1-Score=0.1111111111111111, AUC-ROC=0.7569444444444445
Class 7: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.8472222222222222
Class 8: Accuracy=0.5, F1-Score=0.3333333333333333, AUC-ROC=0.8368055555555557
Class 9: Accuracy=1.0, F1-Score=1.0, AUC-ROC=0.984375
```

Num_Head = 1 with hidden_dim = 1024 performs even worse compared to the model with hidden_dim = 512 . The test accuracy achieved is only 32.5 and F1-score is 0.299.



From the confusion matrix and auc-roc curve we can see that model was incapable of classifying classes like 1 and 7 but it did pretty well at classifying classes like 3,5 and 9 with auc-roc score of 0.92,0.94 and 0.98 which is very good. But the model as a whole performs very bad.

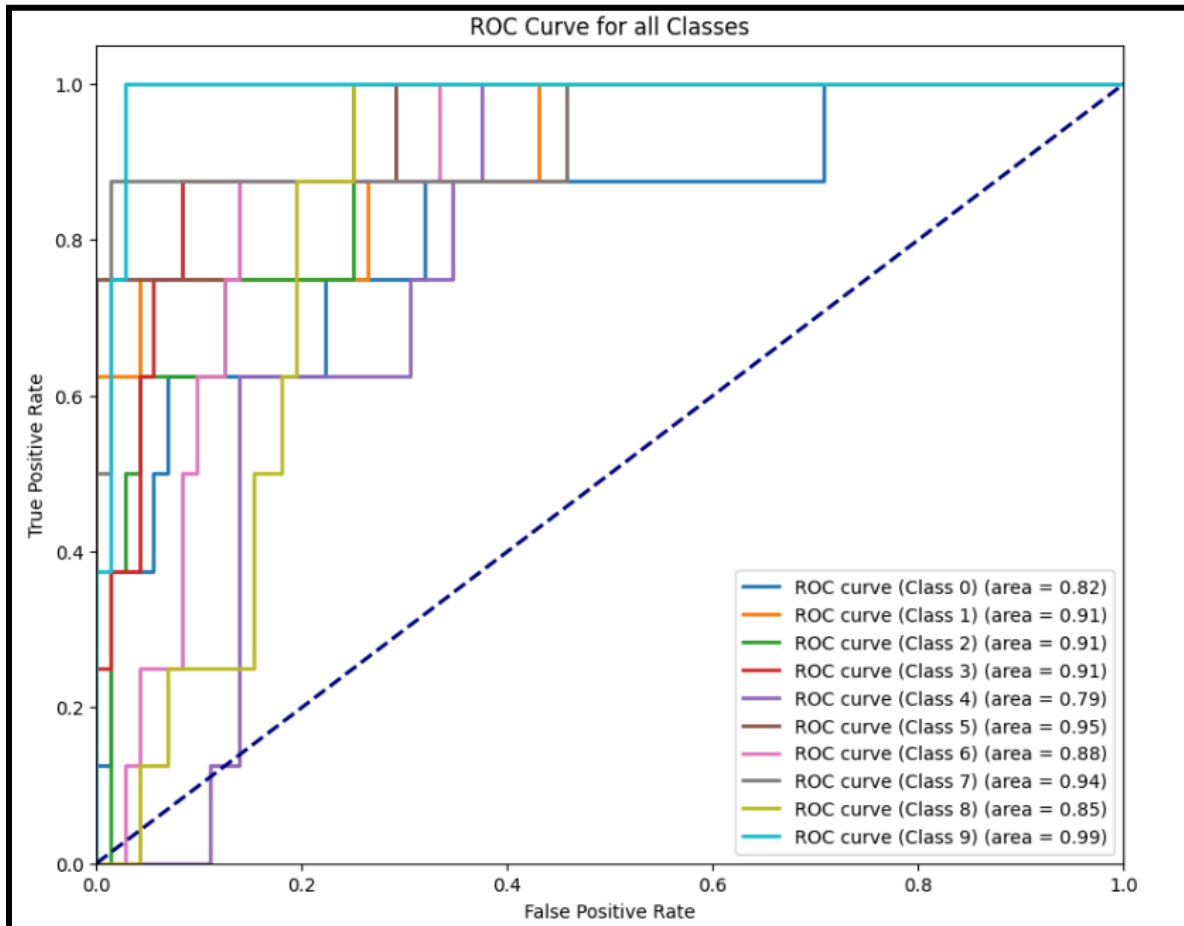
3. Num_Head = 2 , Hidden_dim = 512

Overall Accuracy: 51.24999999999999

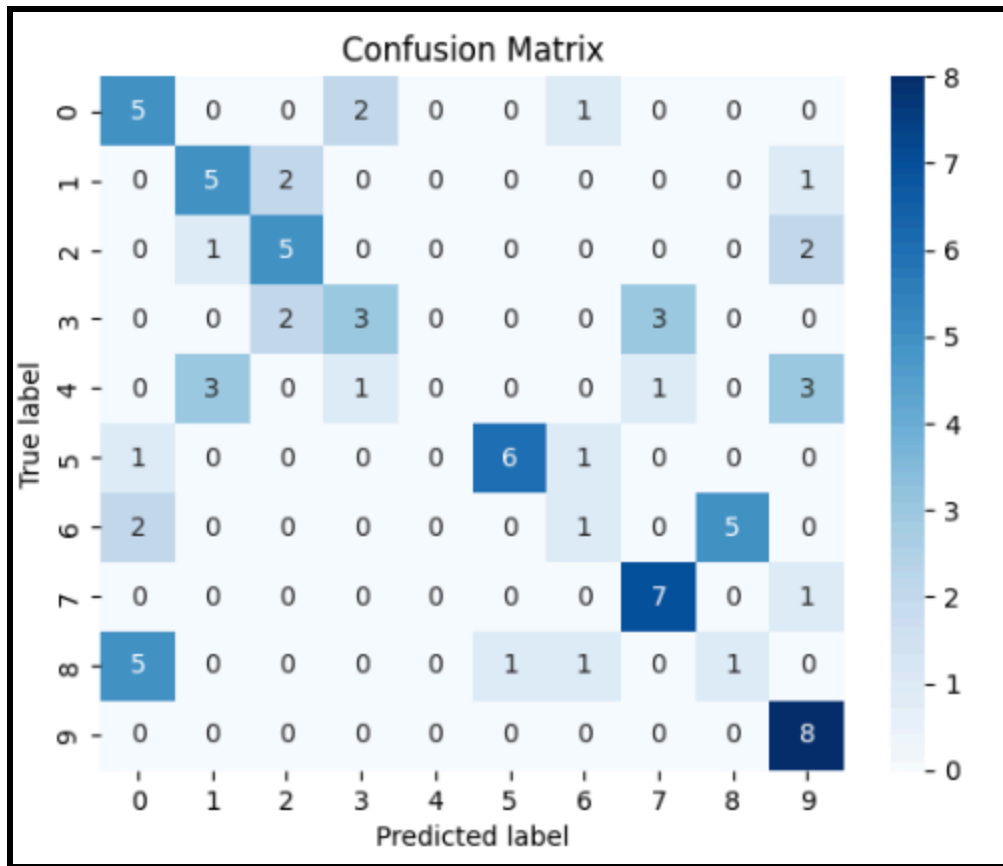
Overall F1-Score: 0.4623250581697209

Class-wise Metrics:

Class 0: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.8246527777777778
Class 1: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9079861111111112
Class 2: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9079861111111112
Class 3: Accuracy=0.375, F1-Score=0.1818181818181818, AUC-ROC=0.9131944444444445
Class 4: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.7881944444444444
Class 5: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.9461805555555556
Class 6: Accuracy=0.125, F1-Score=0.07407407407407407, AUC-ROC=0.8836805555555556
Class 7: Accuracy=0.875, F1-Score=0.4666666666666667, AUC-ROC=0.9375
Class 8: Accuracy=0.125, F1-Score=0.05555555555555555, AUC-ROC=0.8454861111111112
Class 9: Accuracy=1.0, F1-Score=1.0, AUC-ROC=0.9878472222222222



The model with Num_head = 2 and hidden_dim = 512 , achieves a greater accuracy , reason as previously discussed that num_heads = 2 performs way better than num_heads=1 . As a result the overall F1-score is also 0.46 quite better than previous models.



It can be observed from the above confusion matrix that though the model performs great on most of the labels , it still fails to classify label 8 and 4 .

4. Num_Head = 2 , Hidden_dim = 1024

```
Overall Accuracy: 51.249999999999999
Overall F1-Score: 0.4817239623121976

Class-wise Metrics:
Class 0: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.9184027777777778
Class 1: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.8940972222222223
Class 2: Accuracy=0.25, F1-Score=0.1, AUC-ROC=0.8871527777777778
Class 3: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9045138888888889
Class 4: Accuracy=0.0, F1-Score=0.0, AUC-ROC=0.8923611111111112
Class 5: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.9461805555555555
Class 6: Accuracy=0.875, F1-Score=0.4666666666666667, AUC-ROC=0.9618055555555556
Class 7: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.8559027777777778
Class 8: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.9270833333333334
Class 9: Accuracy=0.75, F1-Score=0.42857142857142855, AUC-ROC=0.9288194444444445
```

By changing the hidden_dim from 512 to 1024 we don't see any significant improvement in terms of accuracy. Though the model now was able to classify label 8 which it was incapable of when hidden_dim = 512.

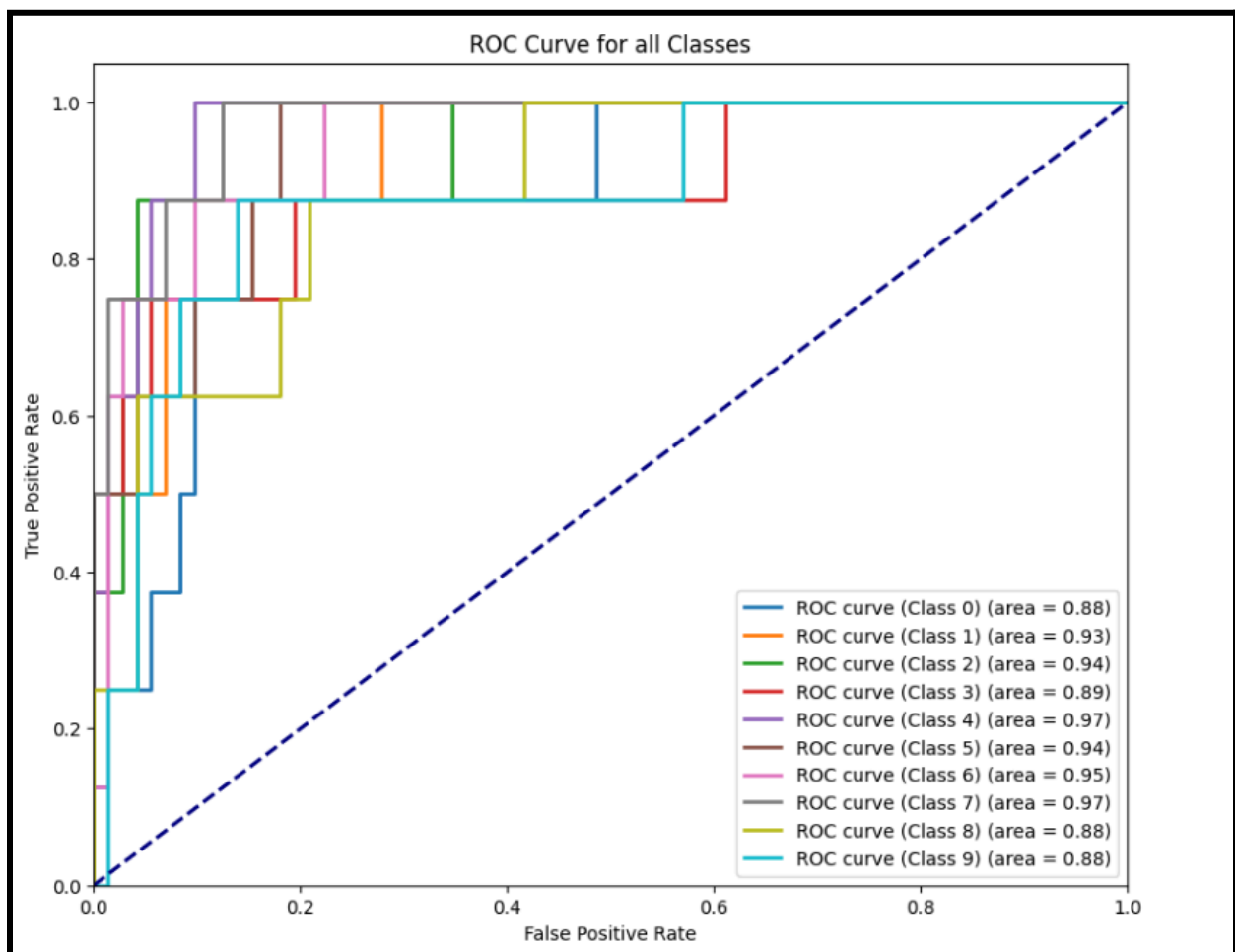
5. Num_Head = 4 , Hidden_dim = 512

Overall Accuracy: 57.49999999999999

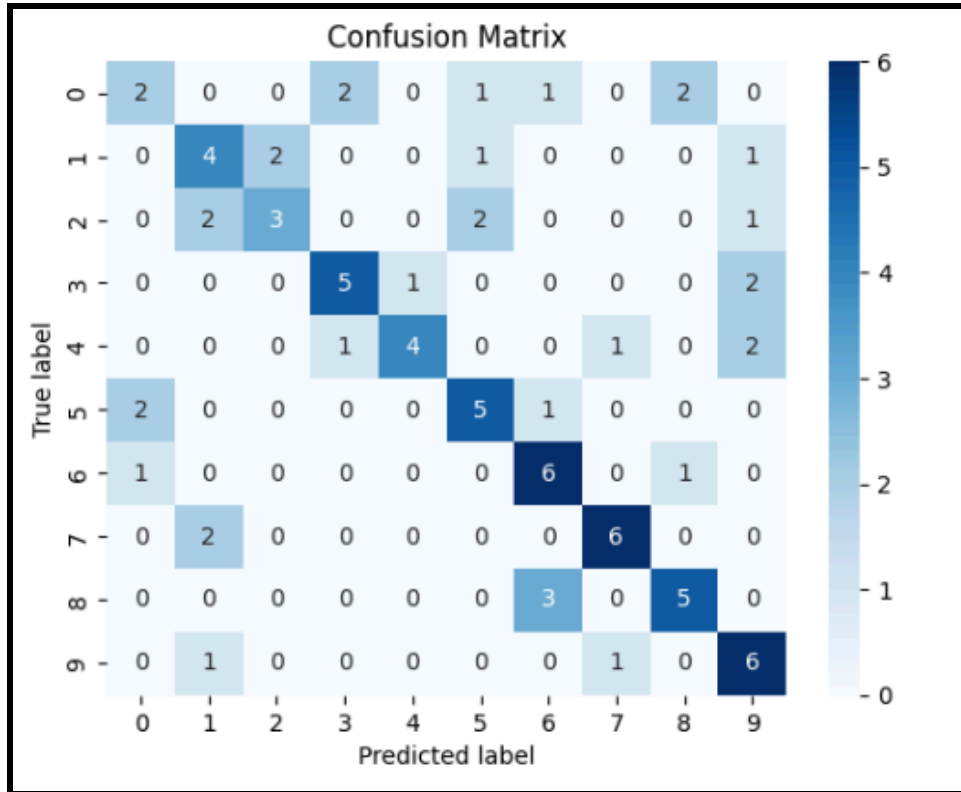
Overall F1-Score: 0.567501786139557

Class-wise Metrics:

Class 0: Accuracy=0.25, F1-Score=0.08, AUC-ROC=0.8767361111111111
Class 1: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.9288194444444444
Class 2: Accuracy=0.375, F1-Score=0.13636363636363635, AUC-ROC=0.9392361111111112
Class 3: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.8871527777777778
Class 4: Accuracy=0.5, F1-Score=0.16666666666666666, AUC-ROC=0.9722222222222222
Class 5: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.9409722222222222
Class 6: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.9496527777777778
Class 7: Accuracy=0.75, F1-Score=0.42857142857142855, AUC-ROC=0.9722222222222222
Class 8: Accuracy=0.625, F1-Score=0.38461538461538464, AUC-ROC=0.8836805555555556
Class 9: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.8802083333333334



The model with Num_head = 4 and hidden_dim = 512 , achieves a greater accuracy , reason as previously discussed that num_heads = 4 performs way better than num_heads=1 and 2. As a result the overall F1-score is also 0.56 way better than previous models and the test accuracy was 57.49% .



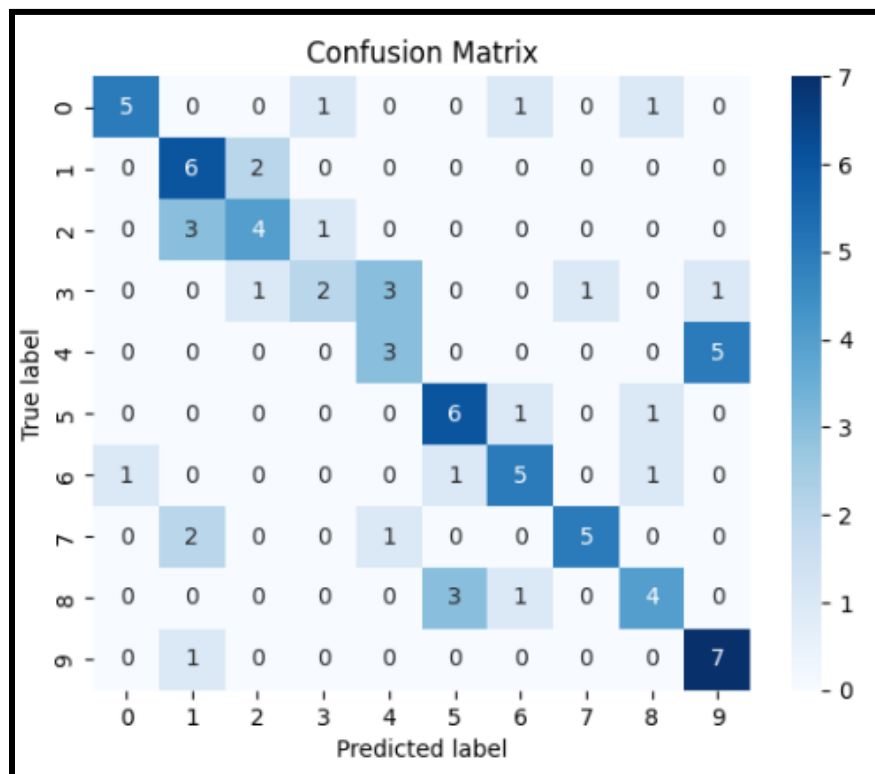
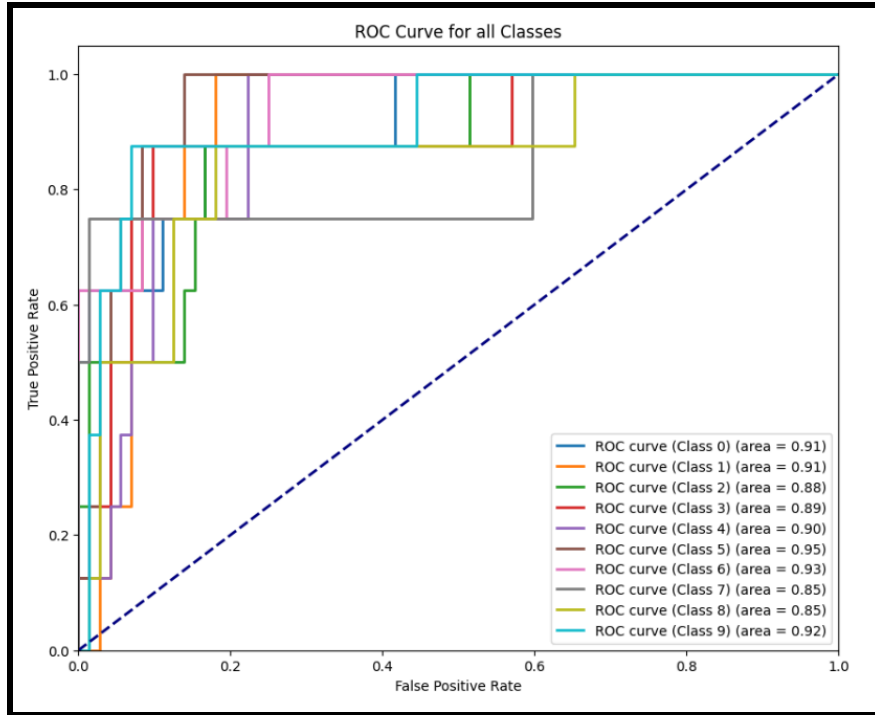
As can be observed from the above AUC-ROC curve and the confusion matrix that model performs quite well . Almost all the classes have accuracy greater than 50% . Though there's still some misclassification of labels like 0 and 3

6. Num_Head = 4 , Hidden_dim = 1024

```
Overall Accuracy: 58.75
Overall F1-Score: 0.5786904761904762

Class-wise Metrics:
Class 0: Accuracy=0.625, F1-Score=0.19230769230769232, AUC-ROC=0.9114583333333333
Class 1: Accuracy=0.75, F1-Score=0.42857142857142855, AUC-ROC=0.9149305555555556
Class 2: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.875
Class 3: Accuracy=0.25, F1-Score=0.08, AUC-ROC=0.8871527777777778
Class 4: Accuracy=0.375, F1-Score=0.2727272727272727, AUC-ROC=0.8975694444444444
Class 5: Accuracy=0.75, F1-Score=0.2857142857142857, AUC-ROC=0.9513888888888889
Class 6: Accuracy=0.625, F1-Score=0.19230769230769232, AUC-ROC=0.9340277777777778
Class 7: Accuracy=0.625, F1-Score=0.25641025641025644, AUC-ROC=0.8472222222222222
Class 8: Accuracy=0.5, F1-Score=0.2222222222222222, AUC-ROC=0.8524305555555555
Class 9: Accuracy=0.875, F1-Score=0.4666666666666667, AUC-ROC=0.9166666666666667
```

This is the best model so far , it achieves test accuracy of 58.75 and F1-score of 0.58 . The accuracy can be justified given the num_head = 4 and hidden_dim = 1024 . As observed earlier these numbers tend to provide better results.

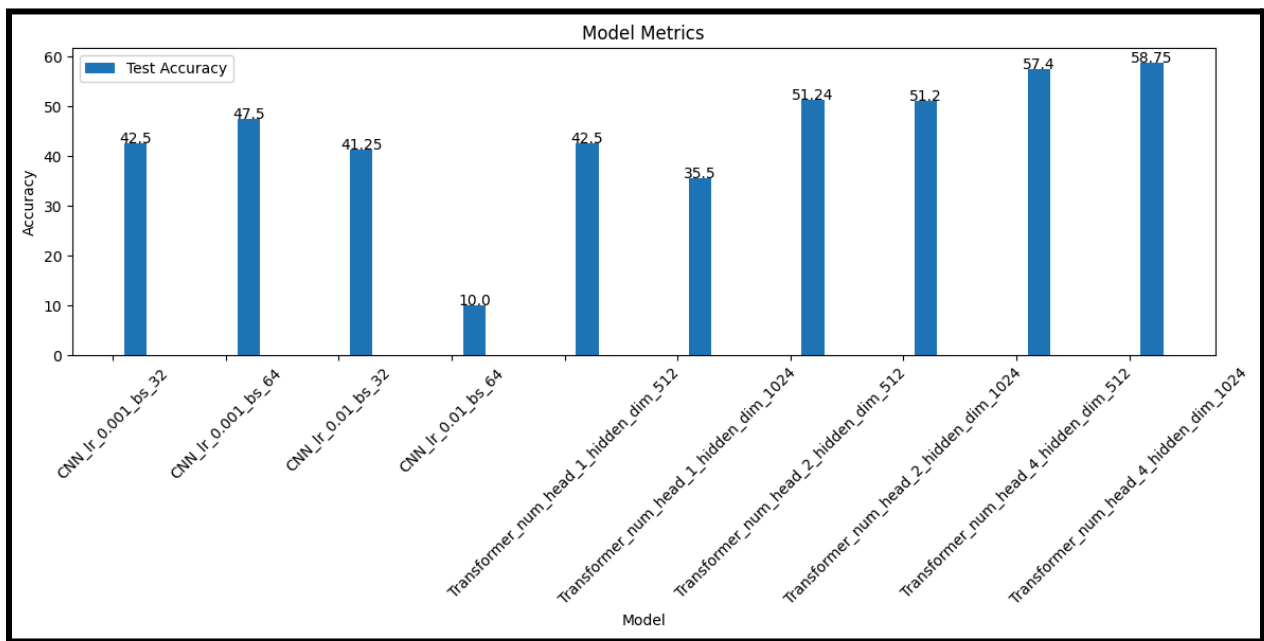


As can be observed from the above AUC-ROC curve and the confusion matrix that model performs quite well . Almost all the classes have accuracy greater than 50% . Some of the labels like 0, 3, 8 and 4 which were misclassified by the previous models are now getting correctly classified by this superior model.

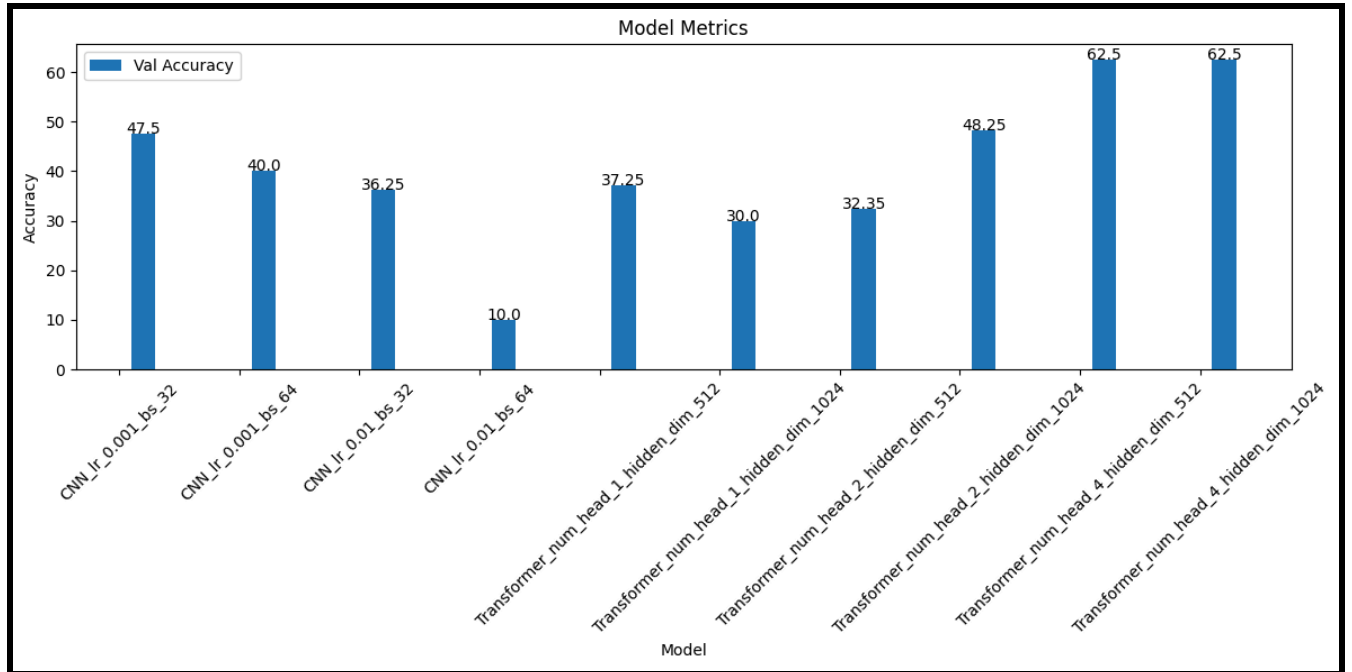
Best Hyperparameters :- num_head: 4, hidden_dim: 512 , 1024
Best Validation Accuracy:- 62.5

Comparative Table :-

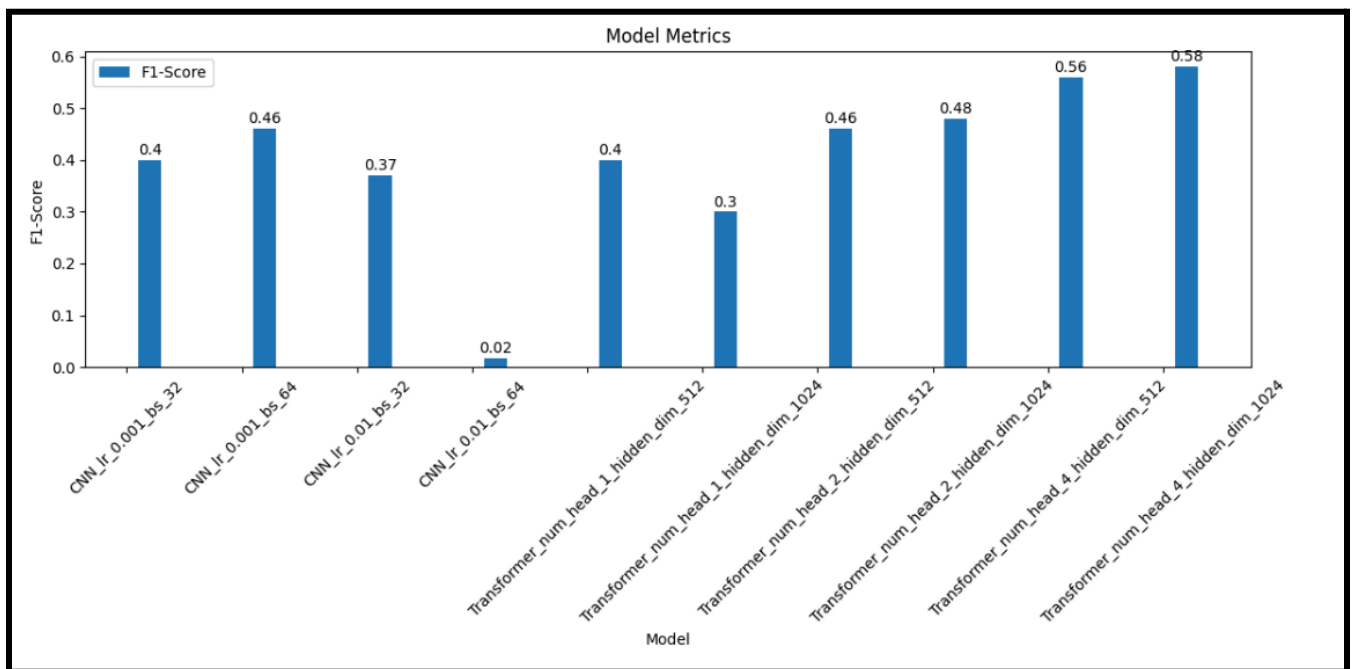
	Model Name	Test Accuracy	Val Accuracy	F1-Score
0	CNN_lr_0.001_bs_32	42.50	47.50	0.400
1	CNN_lr_0.001_bs_64	47.50	40.00	0.460
2	CNN_lr_0.01_bs_32	41.25	36.25	0.370
3	CNN_lr_0.01_bs_64	10.00	10.00	0.018
4	Transformer_num_head_1_hidden_dim_512	42.50	37.25	0.400
5	Transformer_num_head_1_hidden_dim_1024	35.50	30.00	0.300
6	Transformer_num_head_2_hidden_dim_512	51.24	32.35	0.460
7	Transformer_num_head_2_hidden_dim_1024	51.20	48.25	0.480
8	Transformer_num_head_4_hidden_dim_512	57.40	62.50	0.560
9	Transformer_num_head_4_hidden_dim_1024	58.75	62.50	0.580



This figure depicts the test accuracy of different models



This figure depicts the Validation accuracy of different models



This figure depicts the F1-score of different models

As Evident from the above figure : -

1. The Transformer model with 4 attention heads and hidden dimension 1024 consistently outperforms other models in terms of test accuracy, validation accuracy, and F1-score.
2. Transformer models generally outperform CNN models in terms of test accuracy, validation accuracy, and F1-score, especially models with a higher number of attention heads and hidden dimension sizes.
3. Increasing the number of attention heads and hidden dimension size tends to improve model performance, likely due to the increased capacity to capture complex patterns and dependencies in the data.
4. CNN models with a learning rate of 0.01 and batch size of 64 perform poorly compared to other configurations, indicating sensitivity to hyperparameters.
5. Generally, validation accuracy is slightly higher than test accuracy for most models, indicating that the models may be slightly overfitting to the training data.

** Kaggle notebook was used to perform the whole task , reason being google colab was not getting disconnected again and again and hence was not able to perform the tasks in one go.