Comprehensive Examination of Transformer Networks' Performance on the ESC-10 Dataset Utilizing PyTorch Framework

Colab Link : DL Assignment 2 .ipynb

Objective:

In this assignment, we are required to implement a transformer network in Python using the Pytorch framework. By the end, we should have a working transformer network that can be trained on a simple audio dataset for multi-class classification.

DataSet:

The dataset utilized in this study is the ESC_10 dataset, which is a subset of the ESC_50 dataset. In the ESC_10 dataset, only entries with the ESC_10 flag set to True are included. It comprises 400 environmental audio recordings, categorized into 10 distinct classes. Furthermore, the dataset is divided into five folds for cross-validation purposes.

Data Pre-Processing:

In the ESC-10 dataset, audio files were initially captured at a sampling rate of 44 kHz. To mitigate computational requirements and align with conventional audio processing practices, the signals were downsampled to 16 kHz using the torchaudio.transforms.Resample function. Given that audio lengths within the dataset vary, a windowing strategy was implemented. Each audio file was fragmented into overlapping windows, each spanning one second in duration with a hop size of 0.5 seconds. This ensures that the model receives inputs of a consistent size while also maintaining some temporal context from longer sounds. Furthermore, labels were transformed from a range of 0 to 9.

Regarding data division for general training, validation, and testing, we have adopted the following approach: the training set comprises three subsets of data, while both the validation and test sets consist of one subset each.

Architecture 1: 1D Convolutional Feature Extractor with Classification Head

The core of Architecture 1 is a three-layer 1D convolutional neural network designed to extract discriminative features from raw audio input. Each convolutional layer is followed by a batch normalization layer to improve training stability. Max pooling with a stride of 4 is applied after the first two convolutional layers to reduce dimensionality. Adaptive average pooling is used before the classification head to ensure the input to the fully connected layers has a consistent shape.

- Convolutional Feature Extraction
    - Network Structure: The model uses a convolutional neural network (CNN) architecture with three 1D convolutional layers for feature extraction from raw audio waveforms.
    - Layer Specifications:
        - Conv1: Input channels (1), output channels (32), kernel size (15), stride (4), padding (7).
        - Conv2: Input channels (32), output channels (64), kernel size (11), stride (4), padding (5).
        - Conv3: Input channels (64), output channels (128), kernel size (7), stride (2), padding (3).
        - BatchNorm1D: Batch normalization layers are used to accelerate training and reduce sensitivity to initialization.
        - Pooling: MaxPool1d layers (kernel size 4, stride 4) are used for downsampling feature maps.
        - Adaptive Average Pooling: This layer is used to produce an output of fixed size (256) for the fully connected layers.
- Classification Head
    - Fully Connected Layers:
        - FC1: Input features (128 * 256, flattened), output features (512).
        - FC2: Input features (512), output features (equal to the number of classes).
    - Dropout: Dropout (with probability 0.5) is used as a regularization technique to prevent overfitting.
- Activations:
    - ReLU activation functions are used throughout the network.

Architecture 2: CNN-Transformer with Self-Attention

Architecture 2 builds upon the foundation of Architecture 1, enhancing it with a Transformer-based encoder to leverage the power of self-attention for improved audio classification. Here's a breakdown of its constituent parts:

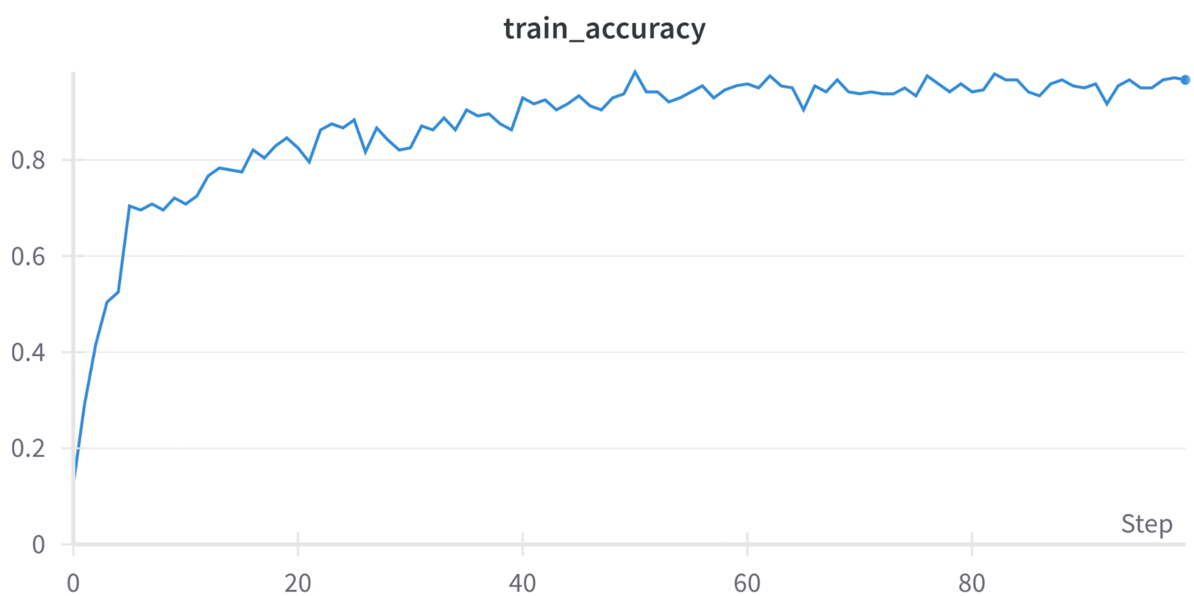
Convolutional Feature Extraction:

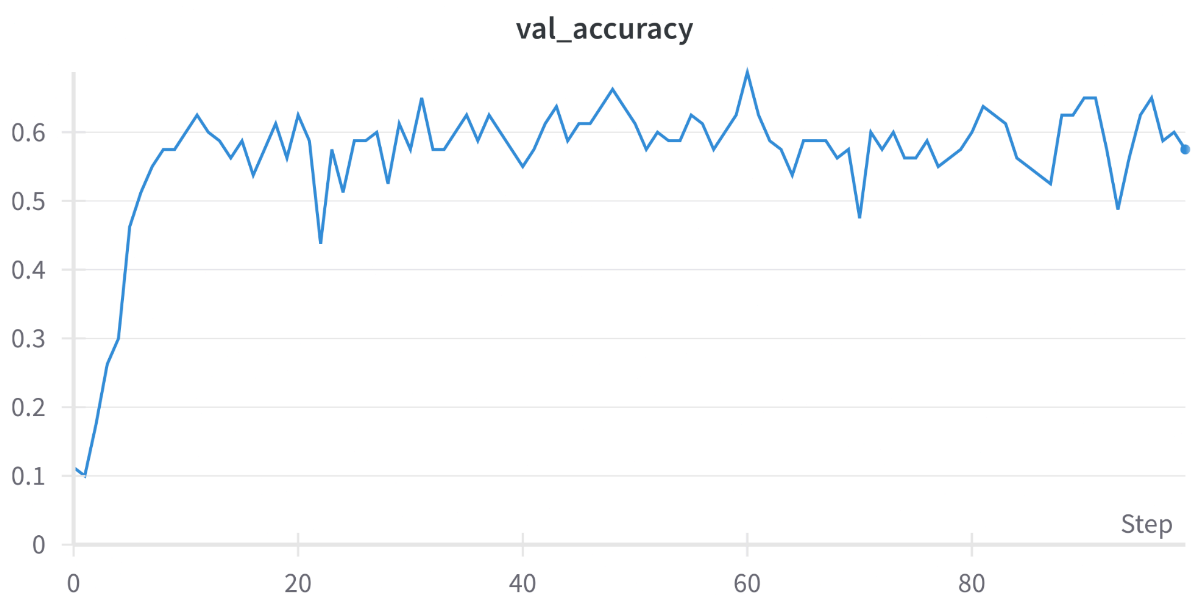
- Backbone: Three-layer convolutional network
- Conv1: Input channels (1), output channels (32), kernel size (15), stride (4), padding (7)
- Conv2: Input channels (32), output channels (64), kernel size (11), stride (4), padding (5)
- Conv3: Input channels (64), output channels (embed_size), kernel size (7), stride (2), padding (3)
- Purpose: Extracts local, discriminative features from the raw audio input
- Output Adaptation: Adjusts the final convolutional layer to produce an output with embed_size channels

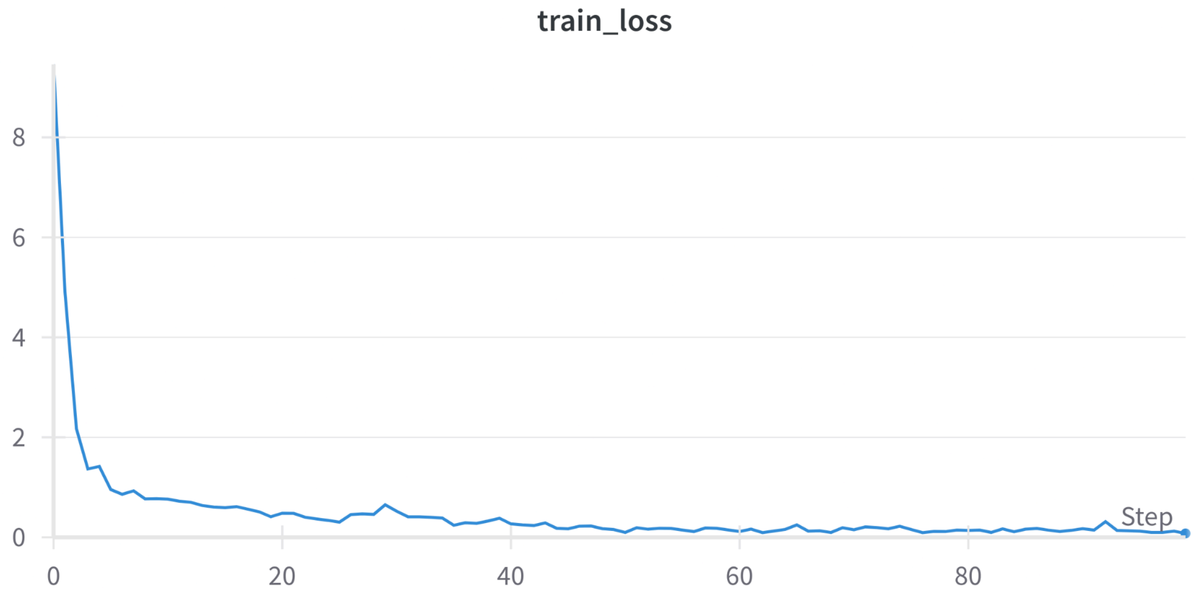
- **Transformer Encoder:**
  - Special <cls> Token: Prepended to the feature sequence extracted by the CNN
  - Positional Encodings: Learned positional encodings added to the CNN output
  - Transformer Blocks: Multiple stacked Transformer blocks
  - Multi-Head Self-Attention: Computes attention weights between different parts of the input sequence
  - Normalization (LayerNorm): Helps stabilize training and improve convergence
  - Feed-Forward Network: Provides further processing of the attention-weighted features
  - Residual Connections and Dropout: Employed for regularization and to enhance gradient flow
- **MLP Classification Head:**
  - Structure: A simple linear layer maps the final representation of the <cls> token to a vector of class scores (logits)
- **Key Advantages of Architecture 2:**
  - Long-Range Dependencies: The Transformer excels at modeling long-range dependencies within the audio feature sequences
  - Global Contextual Information: The <cls> token aggregates global information relevant for the final classification

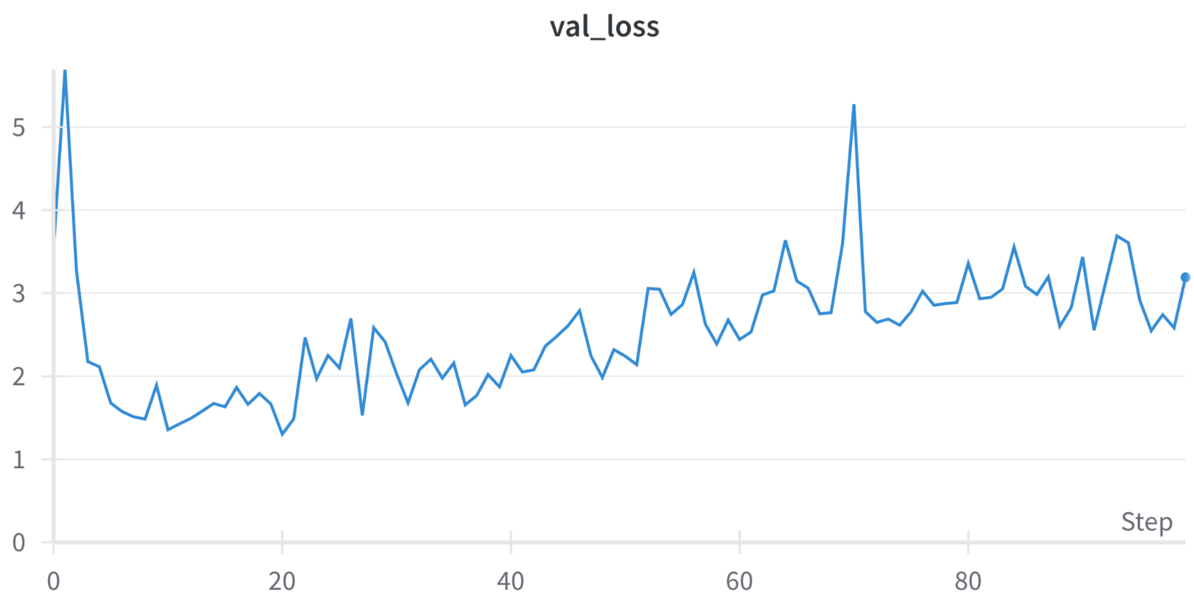
Tasks for Architecture I:

A. Train for 100 epochs. Plot accuracy and loss per epoch on Weight and Biases (WandB) platform.

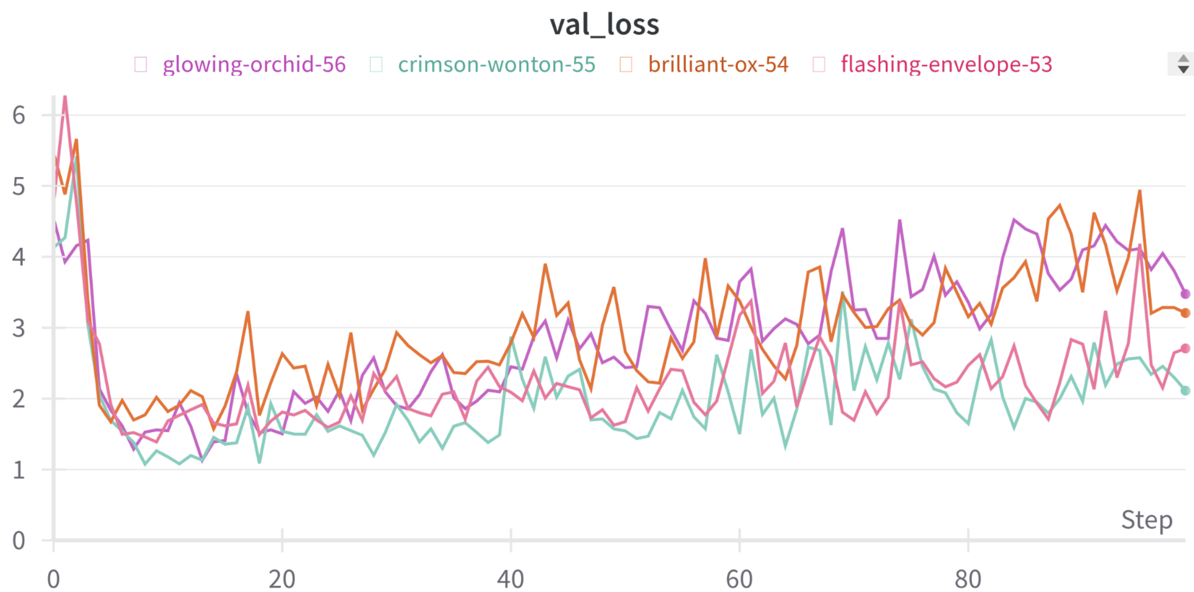


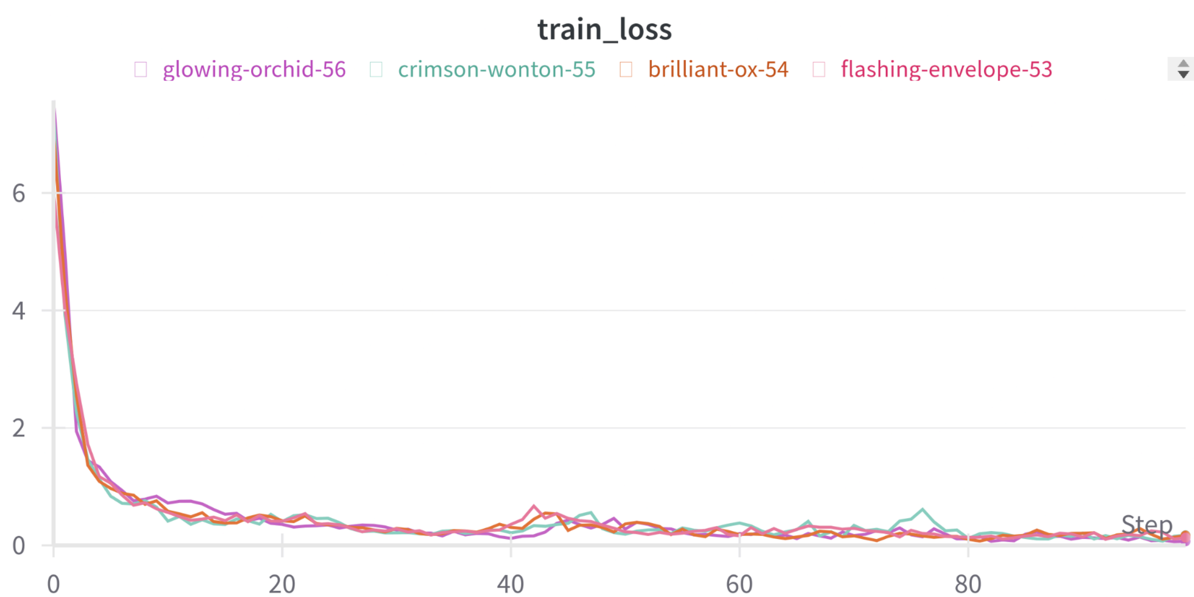


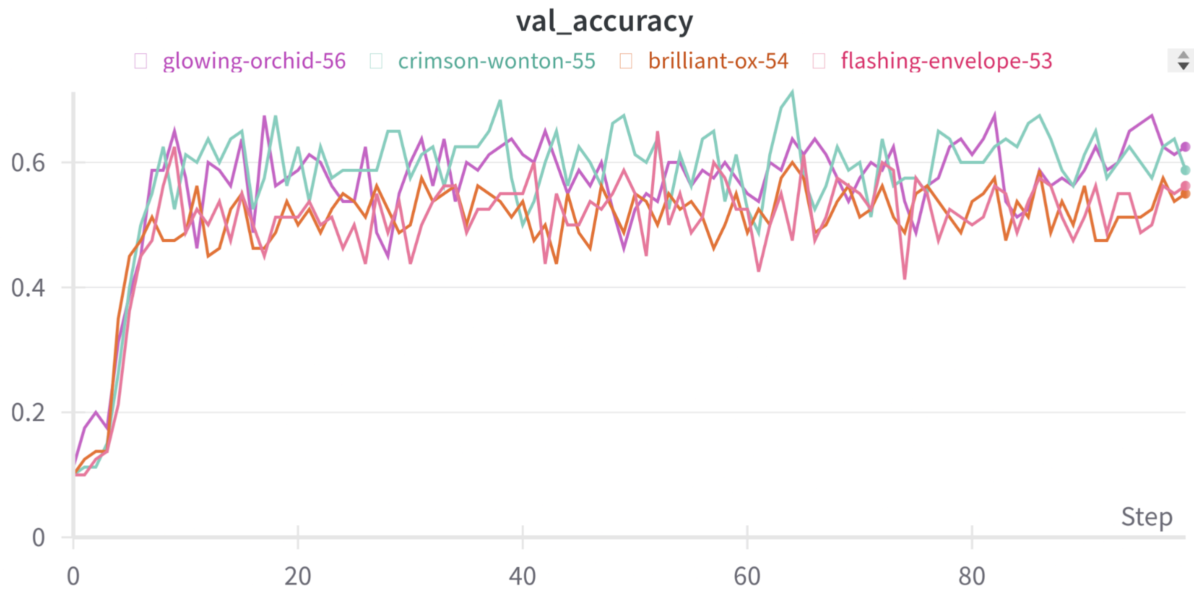

train_loss
8
6
4
2
0
Step
0
20
40
60
80

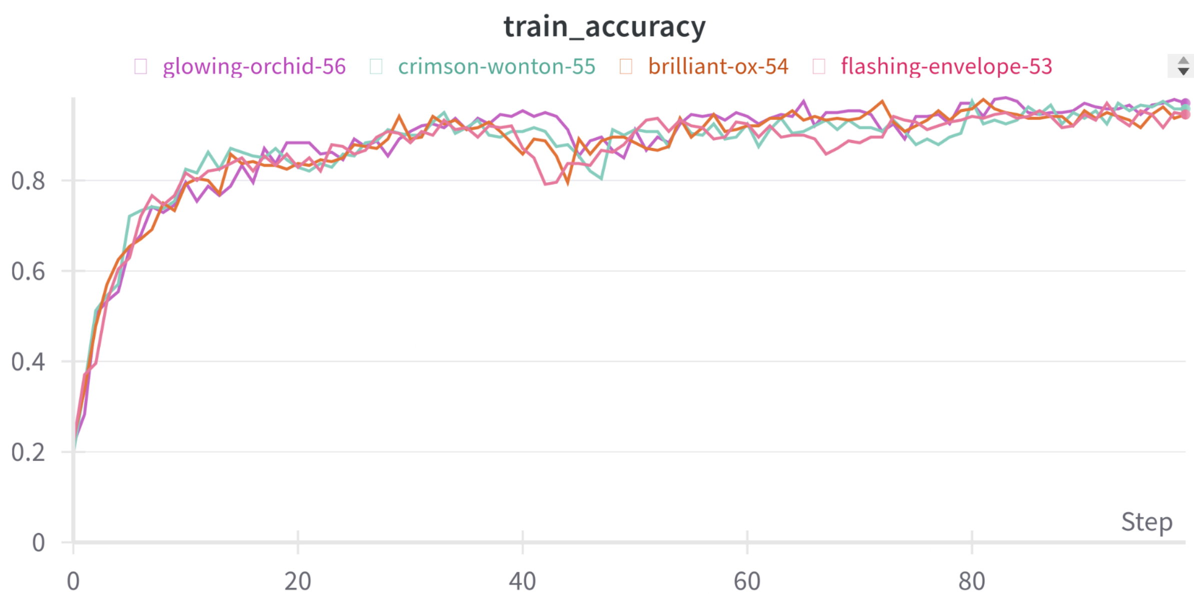

val_loss
5
4
3
2
1
0
Step
0
20
40
60
80

B. Perform k-fold validation, for k=4.



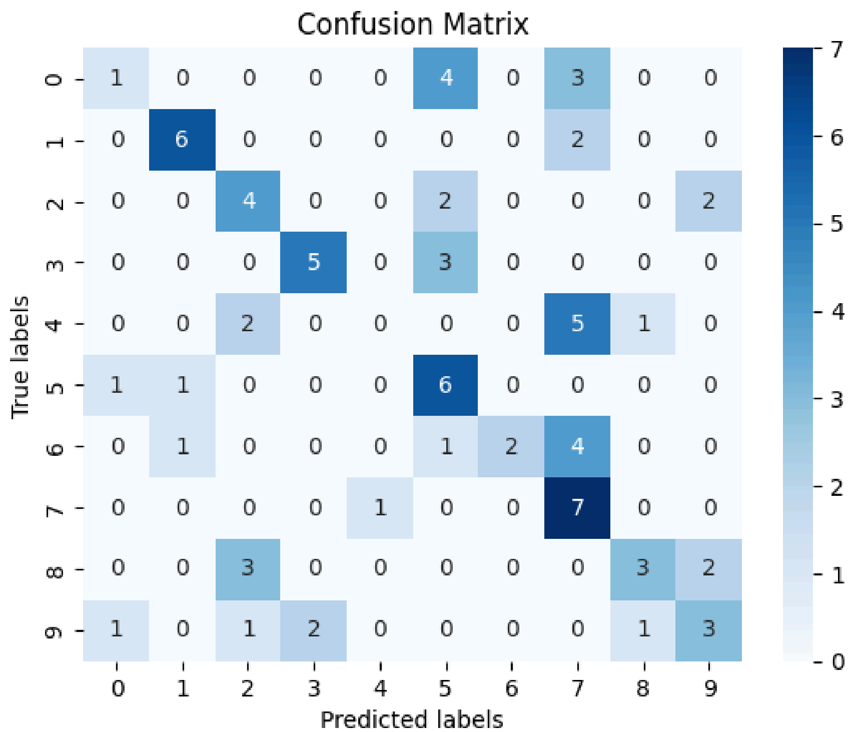


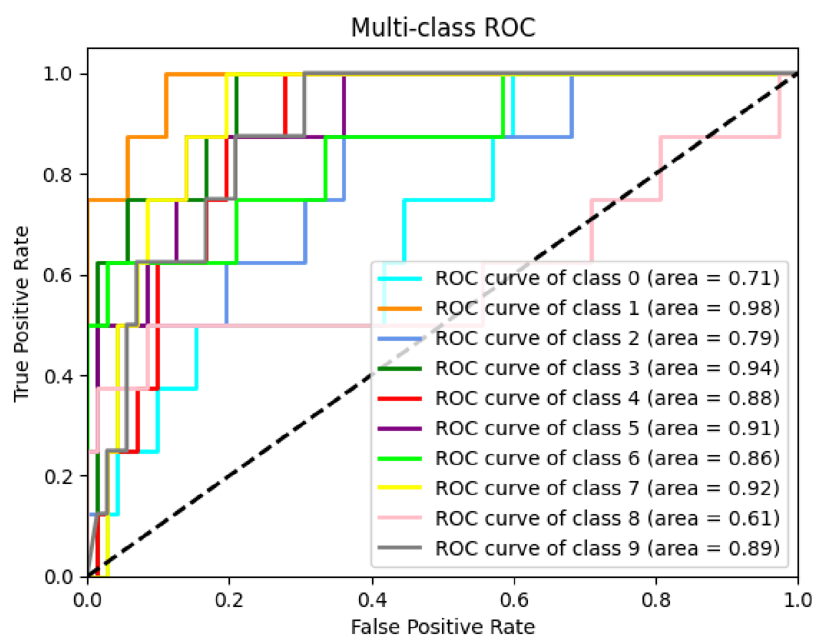




C. Prepare an Accuracy, Confusion matrix, F1-scores, and AUC-ROC curve for the test set

Accuracy: 0.4625
F1-Score (Macro): 0.428722637515741
F1-Score (Weighted): 0.428722637515741
ROC-AUC (One vs Rest): 0.8480034722222223
ROC-AUC (One vs One): 0.8480034722222223



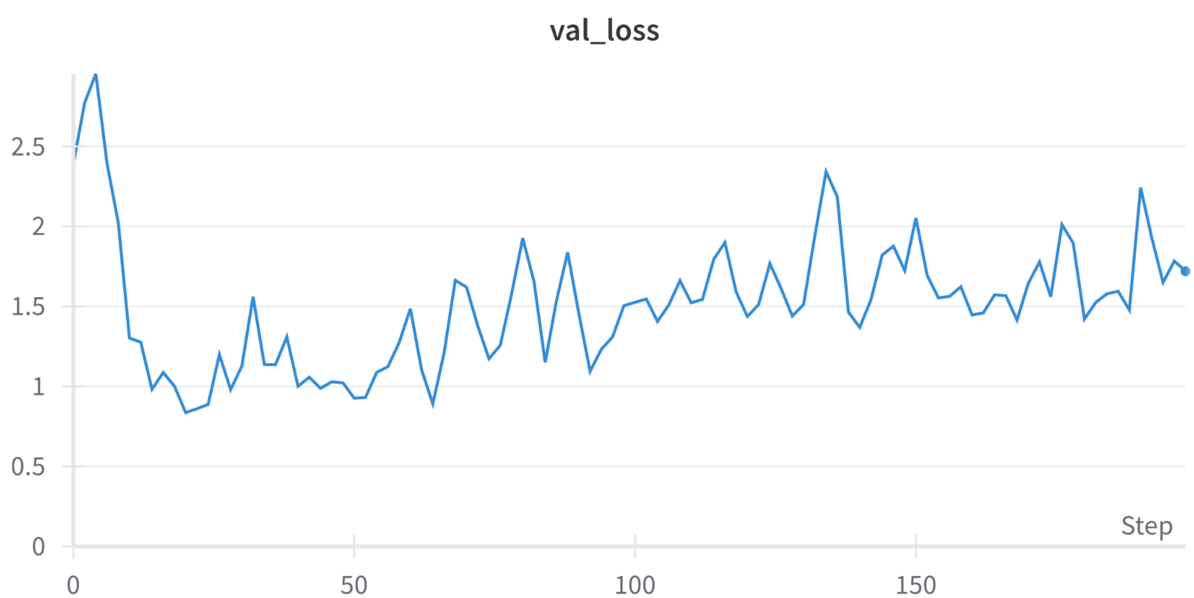


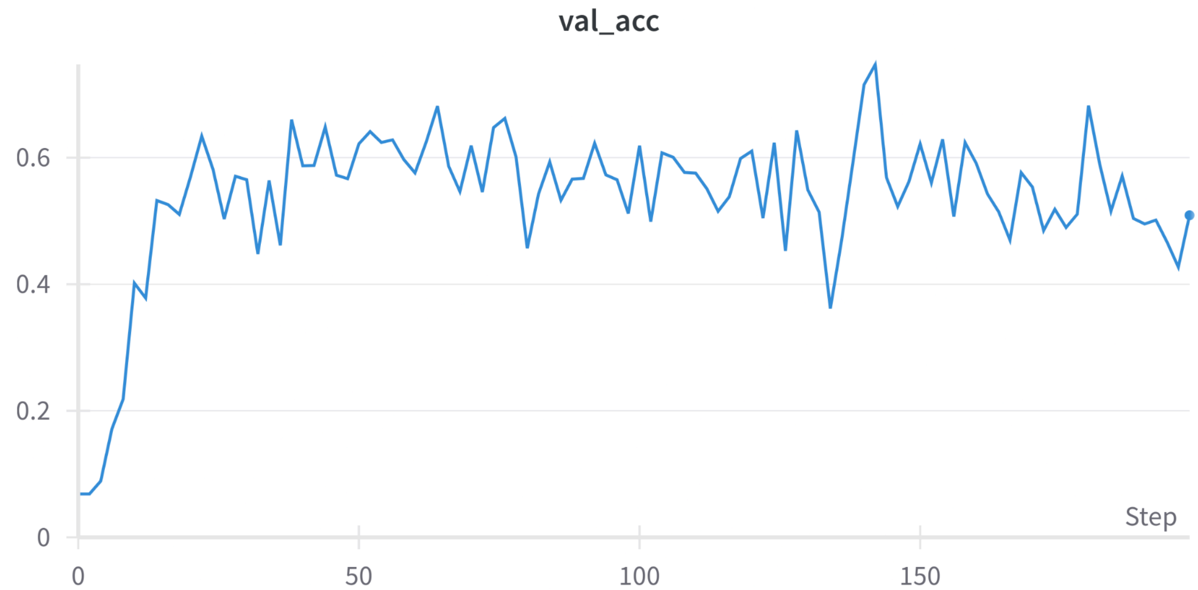
Total Parameters: 16863882
Trainable Parameters: 16863882
Non-Trainable Parameters: 0

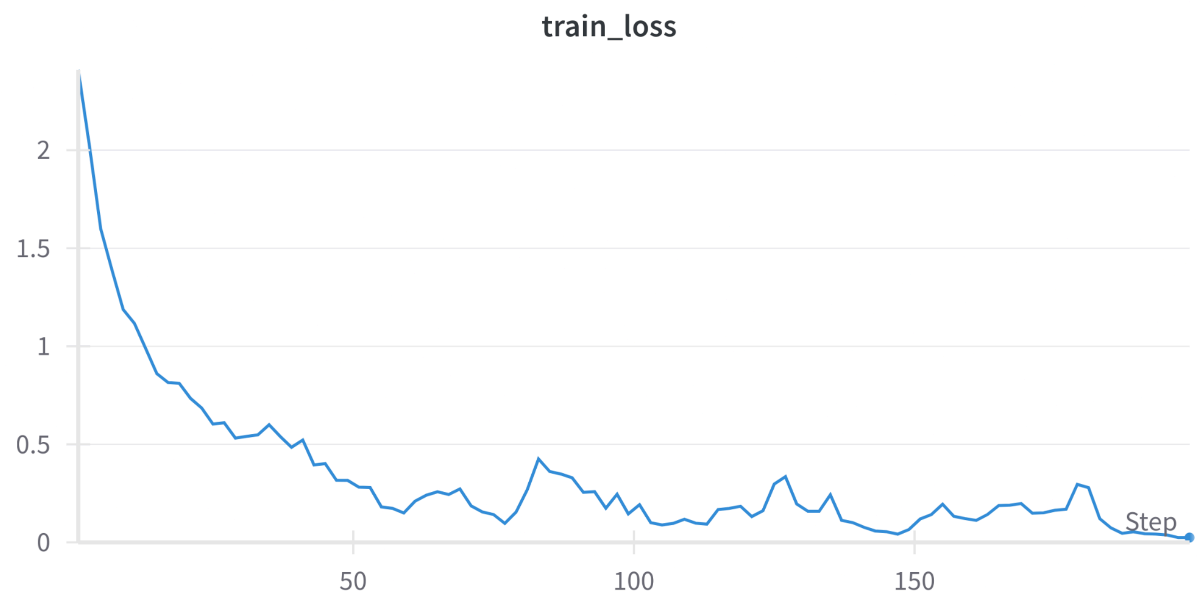
Task for Architecture II:

A. Train for 100 epochs. Plot accuracy and loss per epoch on Weight and Biases (WandB) platform.
B. Prepare an Accuracy, Confusion matrix, F1-scores, and AUC-ROC curve for the test set

- No. Heads = 1

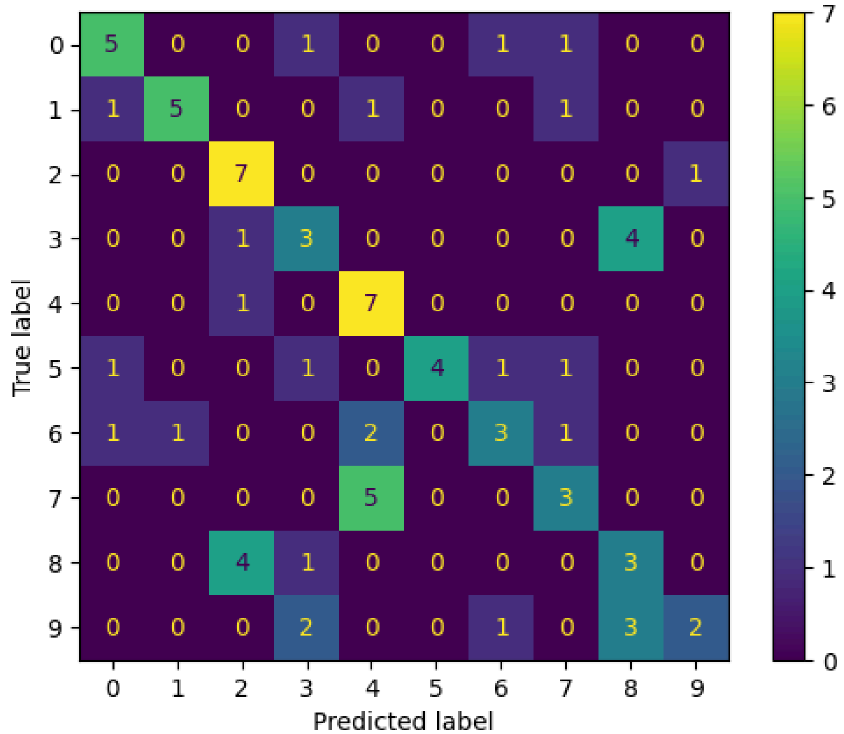




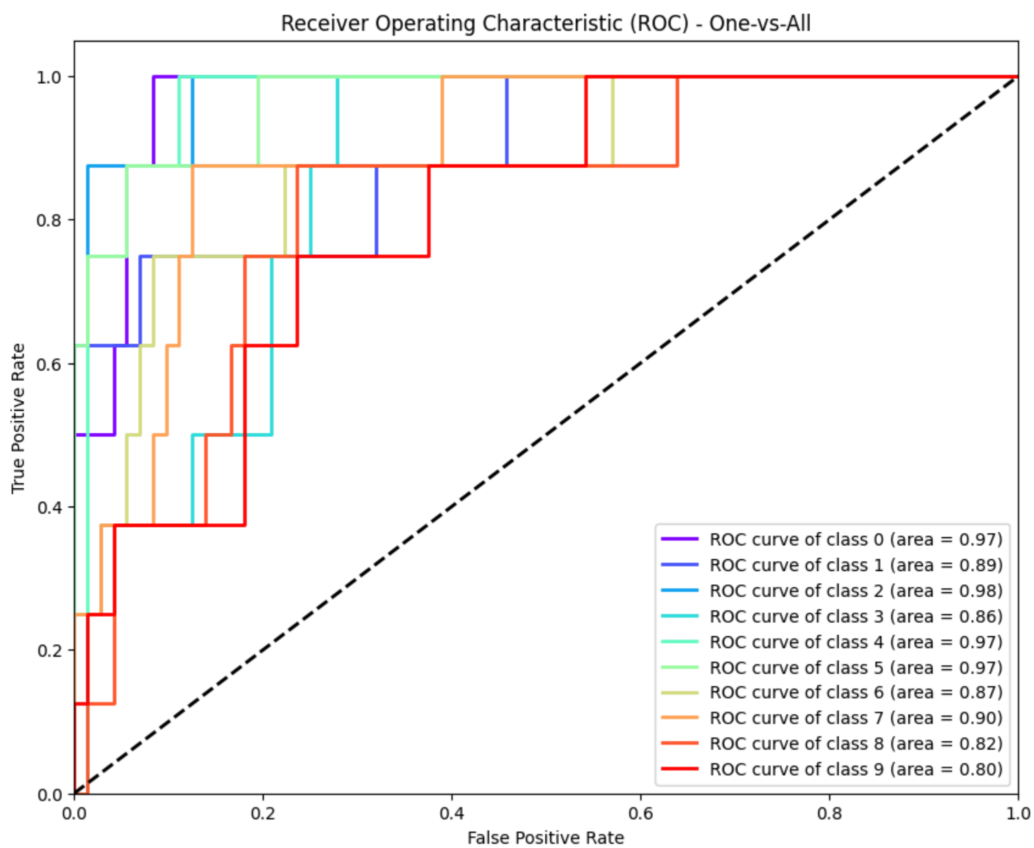


Test:

Accuracy: 0.525
F1 Score: 0.5181855825334086
AUC-ROC: 0.9032986111111112

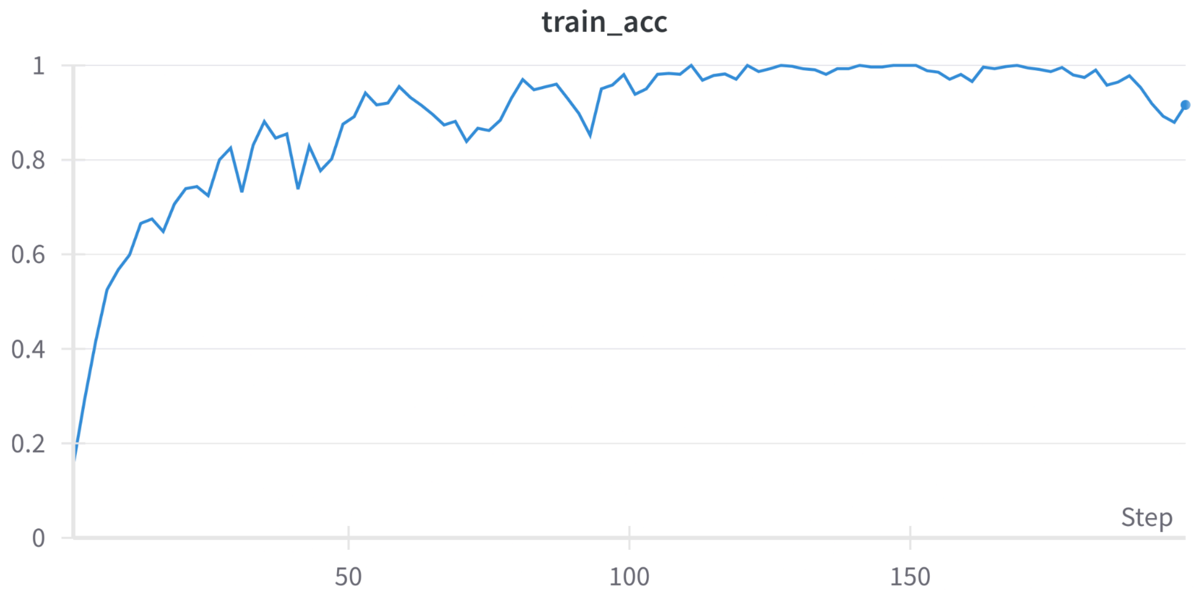


True label
Predicted label
0
1
2
3
4
5
6
7
8
9

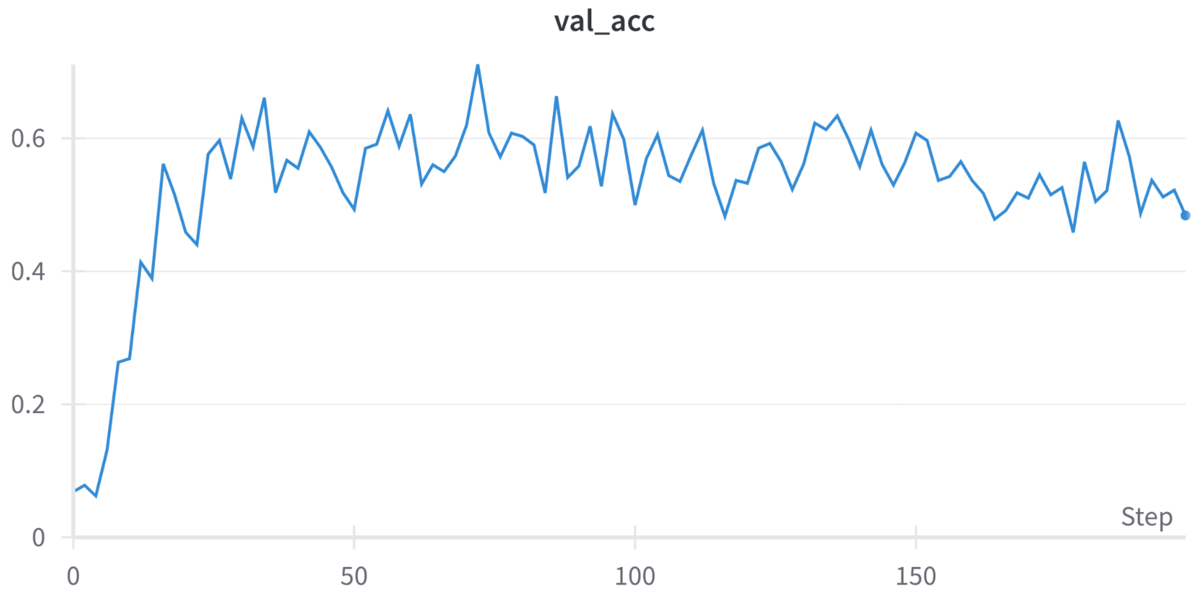


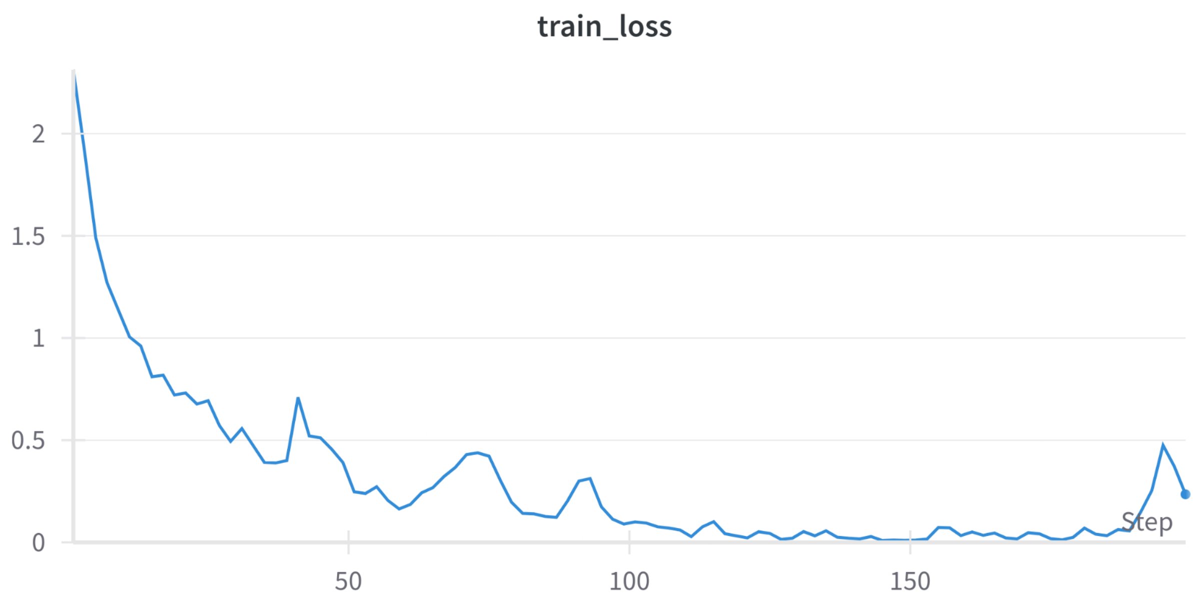
Receiver Operating Characteristic (ROC) - One-vs-All
True Positive Rate
False Positive Rate
ROC curve of class 0 (area = 0.97)
ROC curve of class 1 (area = 0.89)
ROC curve of class 2 (area = 0.98)
ROC curve of class 3 (area = 0.86)
ROC curve of class 4 (area = 0.97)
ROC curve of class 5 (area = 0.97)
ROC curve of class 6 (area = 0.87)
ROC curve of class 7 (area = 0.90)
ROC curve of class 8 (area = 0.82)
ROC curve of class 9 (area = 0.80)

- No. Heads = 2

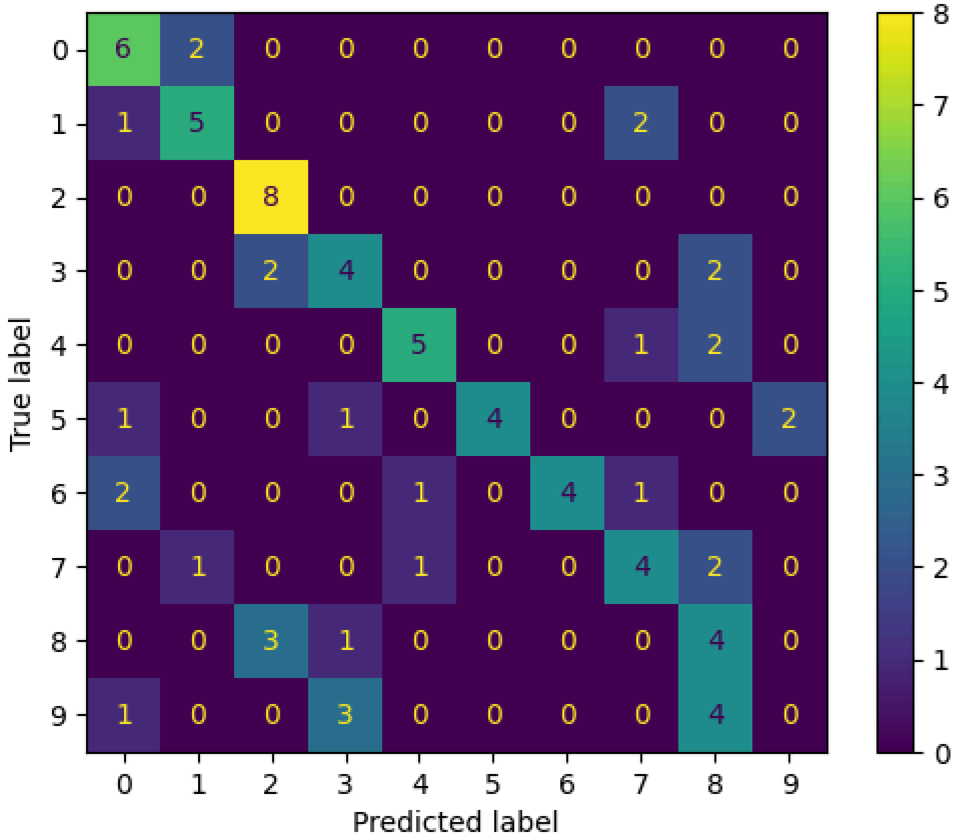

train_acc
1
0.8
0.6
0.4
0.2
0
Step
50
100
150

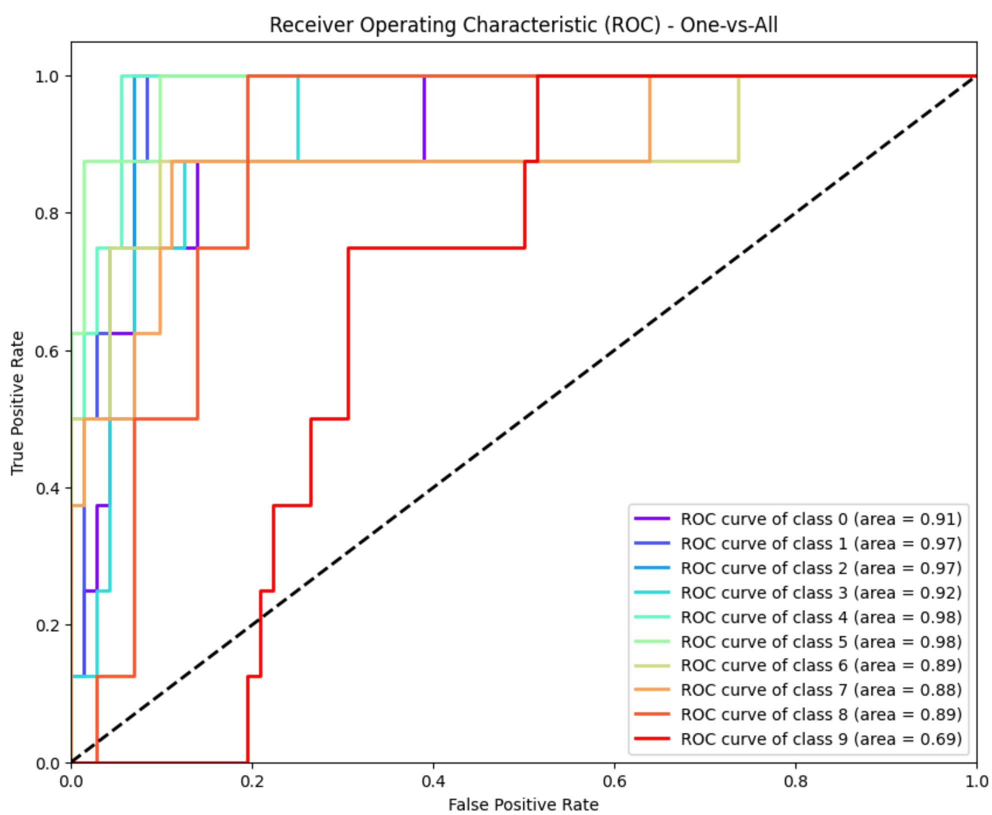

val_loss
3
2.5
2
1.5
1
0.5
0
Step
0
50
100
150



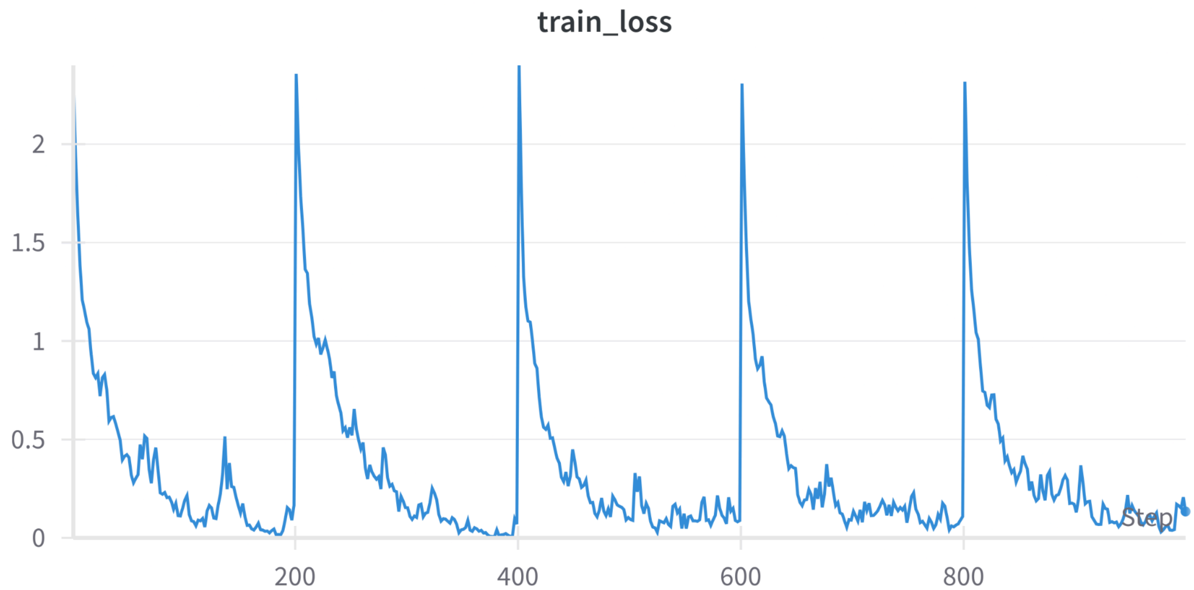


Test:
Accuracy: 0.55
F1 Score: 0.5352708308203664
AUC-ROC: 0.9076388888888889

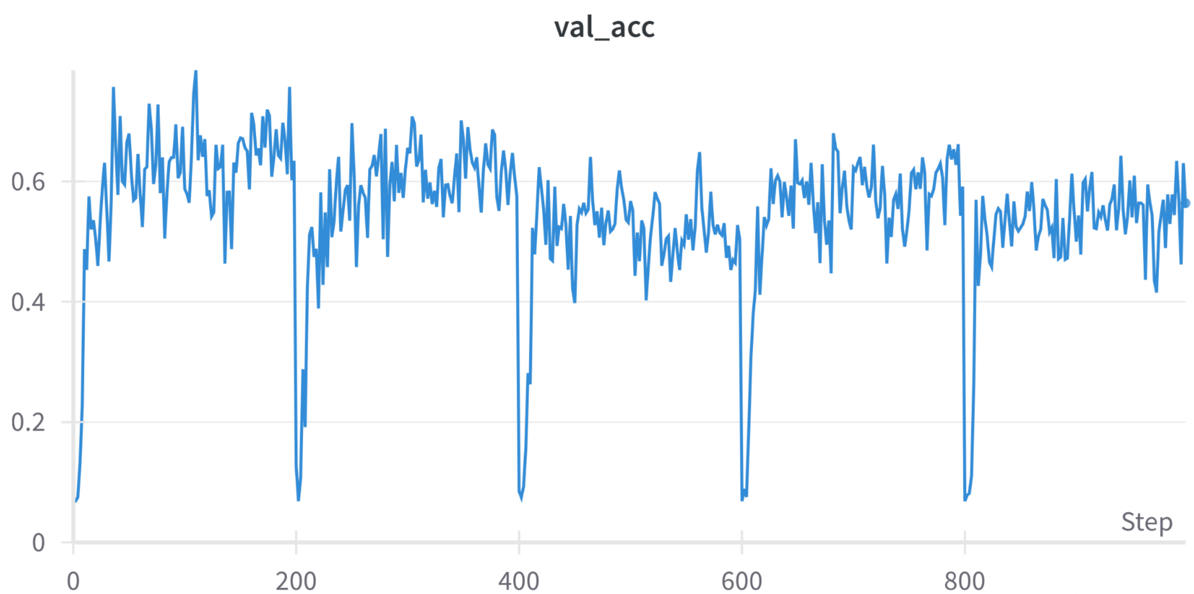

True label
Predicted label


Receiver Operating Characteristic (ROC) - One-vs-All
True Positive Rate
False Positive Rate
ROC curve of class 0 (area = 0.91)
ROC curve of class 1 (area = 0.97)
ROC curve of class 2 (area = 0.97)
ROC curve of class 3 (area = 0.92)
ROC curve of class 4 (area = 0.98)
ROC curve of class 5 (area = 0.98)
ROC curve of class 6 (area = 0.89)
ROC curve of class 7 (area = 0.88)
ROC curve of class 8 (area = 0.89)
ROC curve of class 9 (area = 0.69)

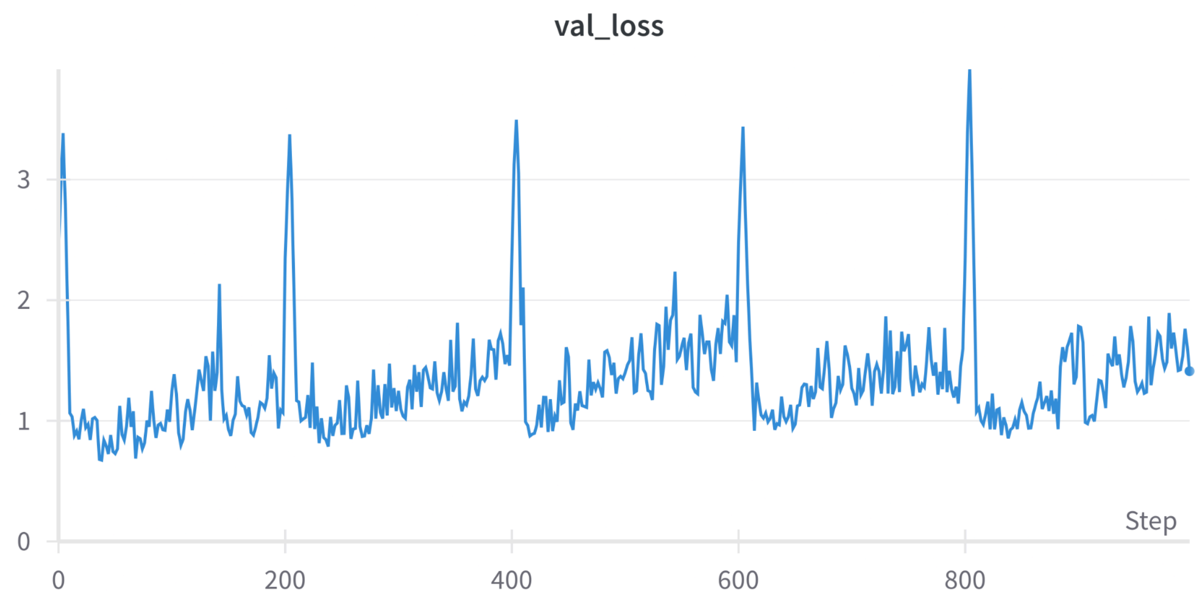
- No. Heads = 4

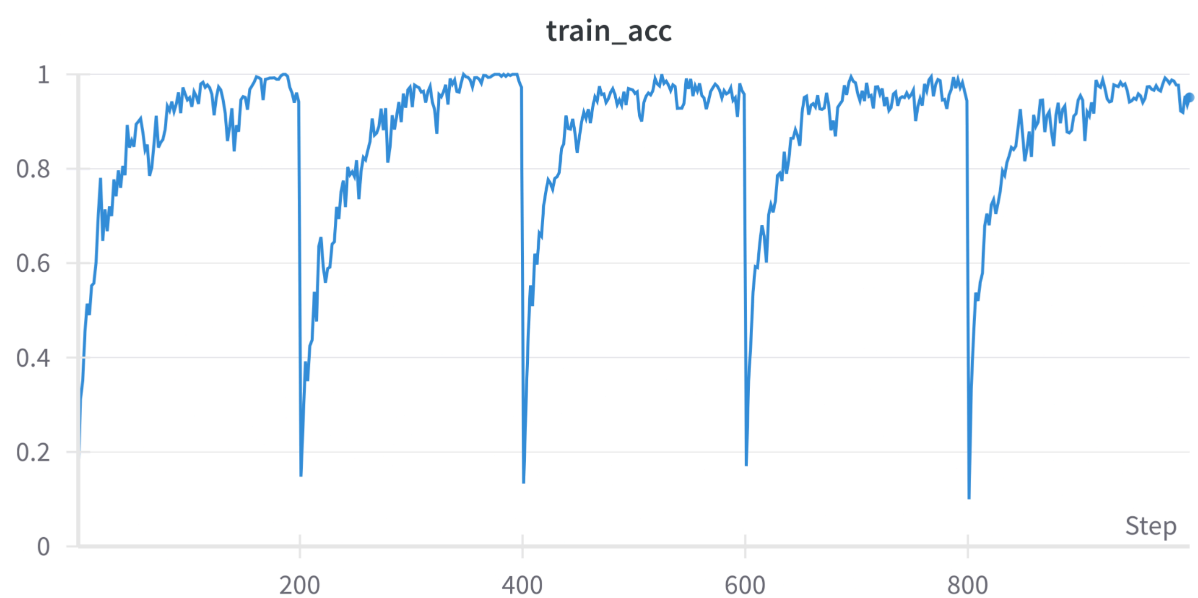


train_loss
2
1.5
1
0.5
0
200
400
600
800
Step

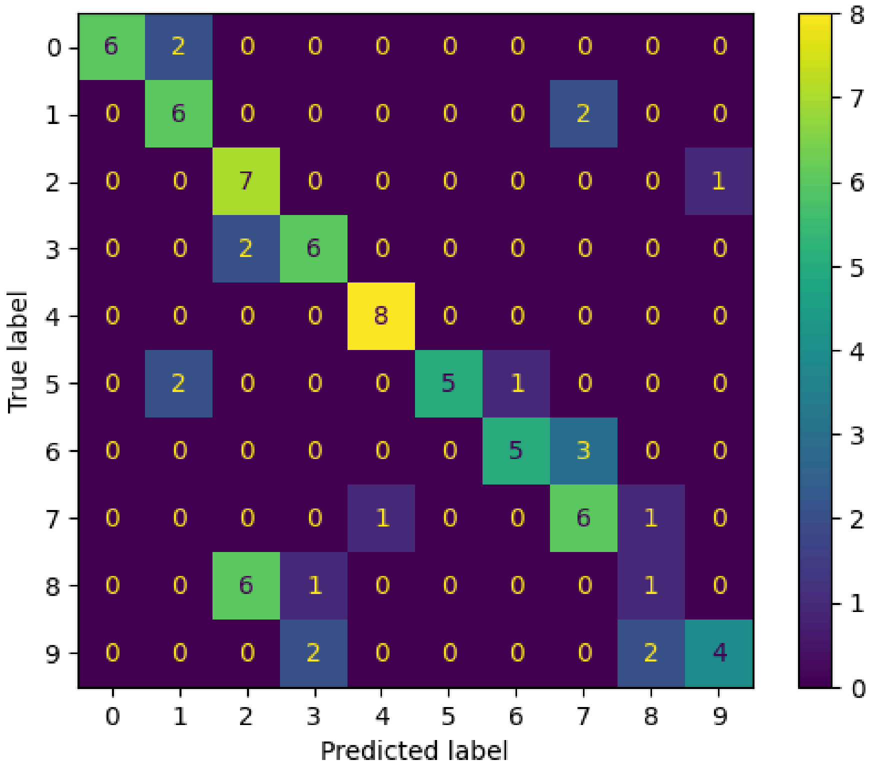


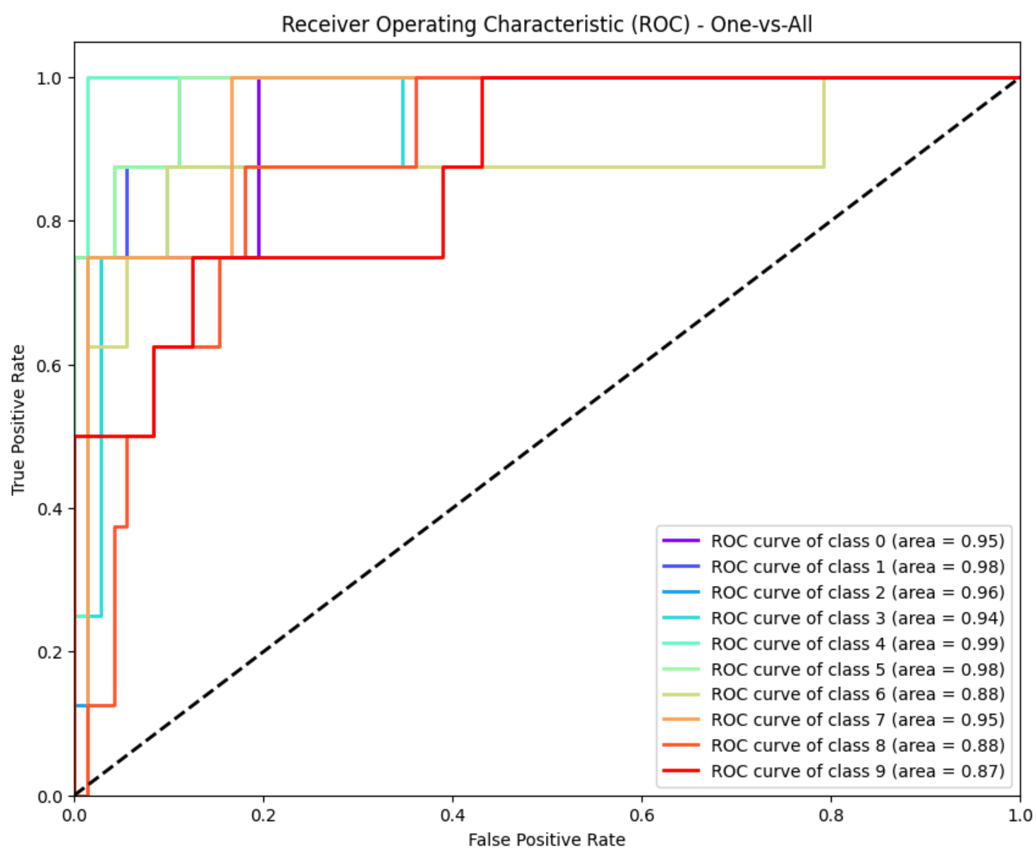
val_acc
0.6
0.4
0.2
0
0
200
400
600
800
Step



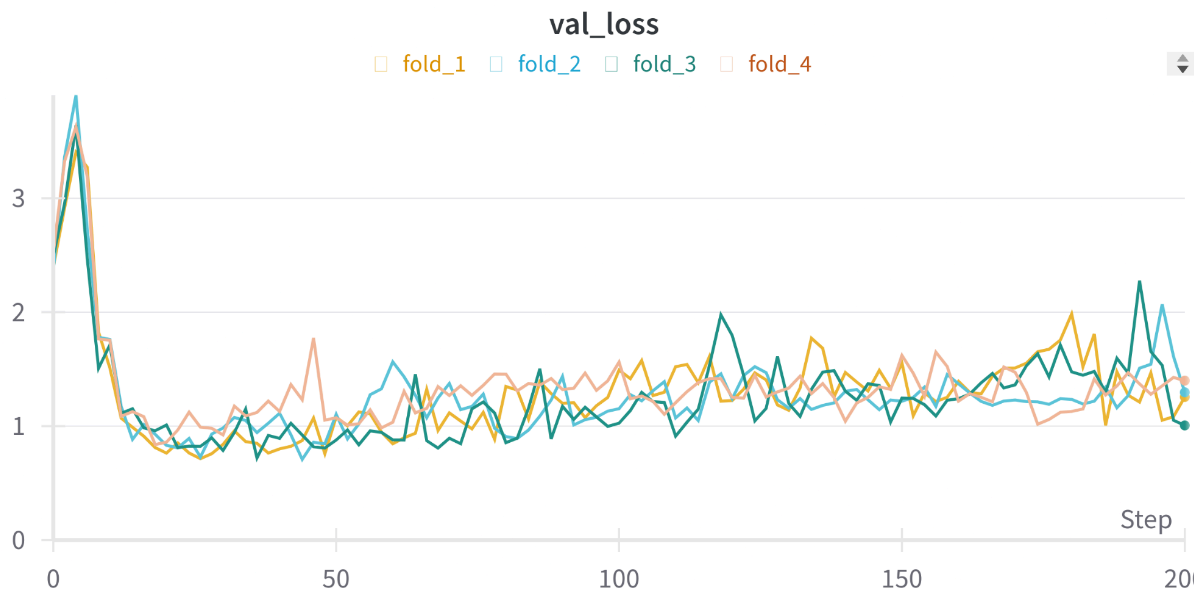


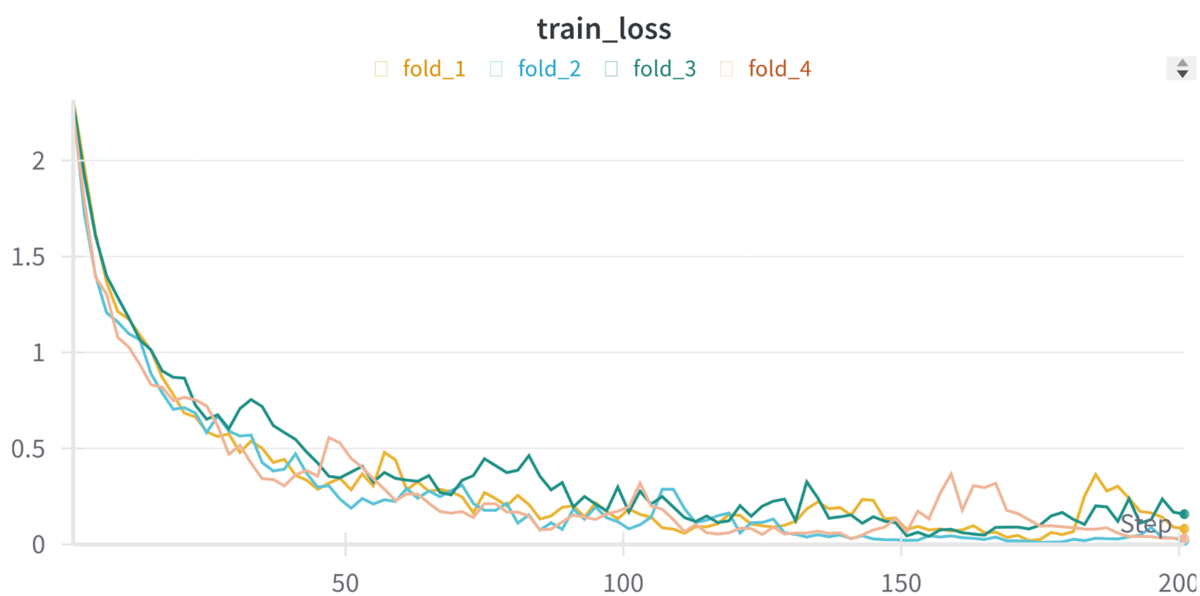
Test:
Accuracy: 0.675
F1 Score: 0.6676710712449037
AUC-ROC: 0.9383680555555556

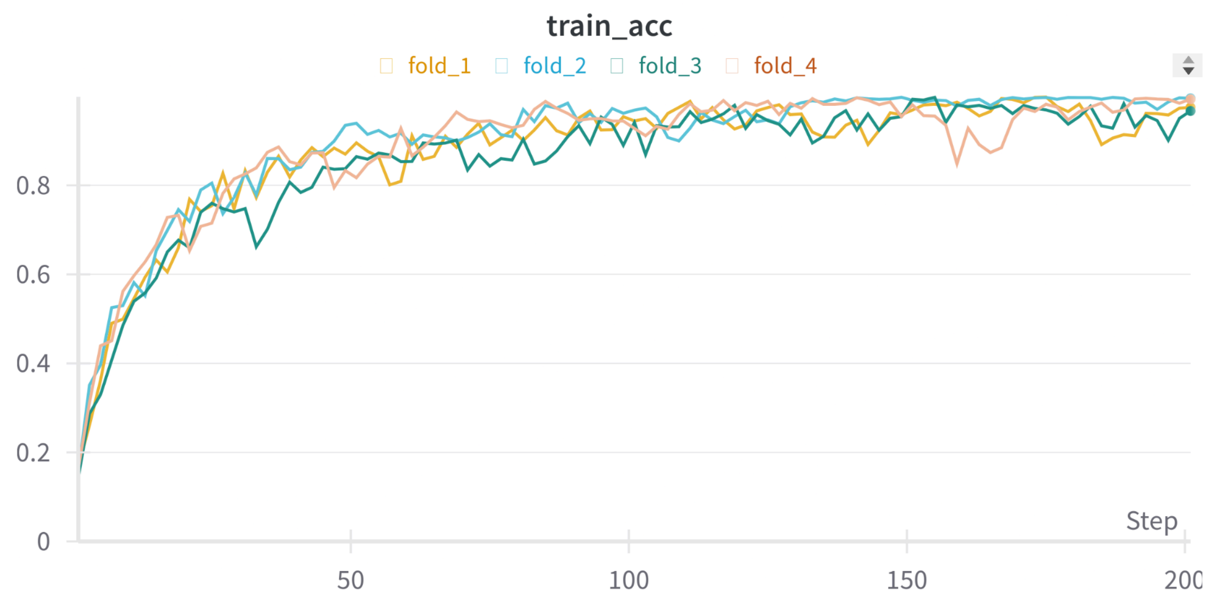

True label
Predicted label

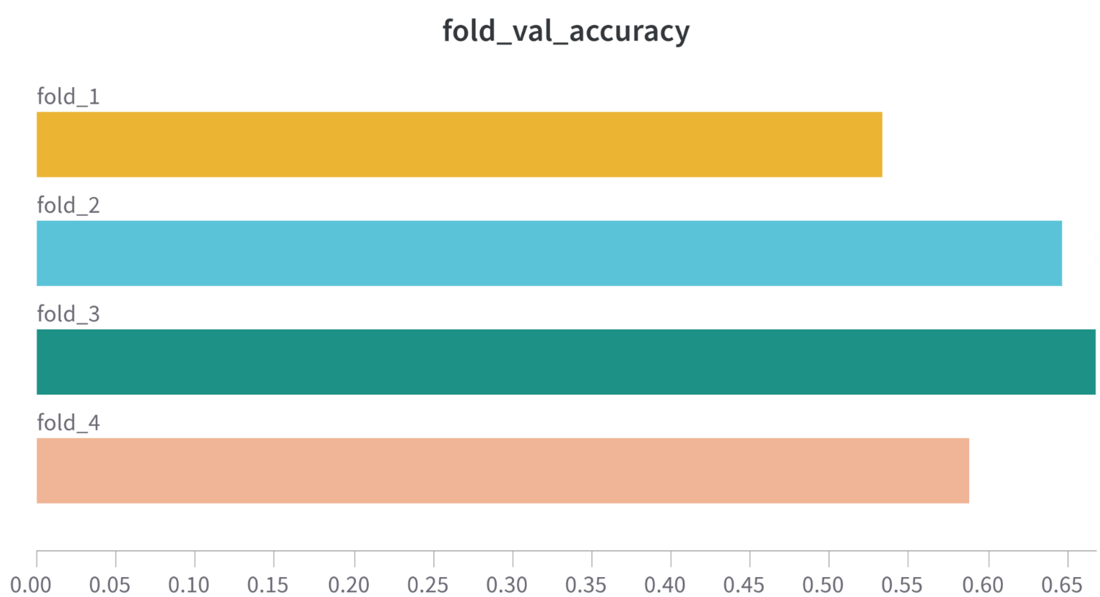

Receiver Operating Characteristic (ROC) - One-vs-All
True Positive Rate
False Positive Rate
ROC curve of class 0 (area = 0.95)
ROC curve of class 1 (area = 0.98)
ROC curve of class 2 (area = 0.96)
ROC curve of class 3 (area = 0.94)
ROC curve of class 4 (area = 0.99)
ROC curve of class 5 (area = 0.98)
ROC curve of class 6 (area = 0.88)
ROC curve of class 7 (area = 0.95)
ROC curve of class 8 (area = 0.88)
ROC curve of class 9 (area = 0.87)

C. Perform k-fold validation, for k=4 (heads = 1).









We generalize the train and validation accuracy so as to get a good idea about our models and then we choose any one model to evaluate the performance. One of the models was shown earlier with head = 1 earlier.

Total Parameters: 511114
Trainable Parameters: 511114
Non-Trainable Parameters: 0

Results:

Architecture I:

Train Loss	Train Accuracy	Val Loss	Val Accuracy	Test Accuracy
0.0781	0.9667	3.1912	0.575	0.4625

Fold	Train Loss	Train Accuracy	Val Loss	Val Accuracy
2	0.1323	0.9458	2.7078	0.5625
3	0.1678	0.9458	3.2079	0.55
4	0.1723	0.9583	2.1127	0.5875
5	0.0697	0.9708	3.476	0.625
Average	2.8761	0.5813	-	-

Architecture II:

No. of Heads	val_loss	val_acc	train_loss	train_acc
1	1.72	0.509	0.02457	0.9973
2	1.5	0.484	0.235	0.916
4	1.07	0.634	0.165	0.942

K fold

fold_train_accuracy	fold_val_accuracy	train_acc	train_loss	trainer	val_acc	val_los
0.9753	0.5337	0.9753	0.08148	807	0.5337	1.256
0.9956	0.6468	0.9956	0.01831	807	0.6468	1.297
0.967	0.6679	0.967	0.1577	807	0.6679	1.006
0.9933	0.5886	0.9933	0.03094	807	0.5886	1.4
-	-	-	-	-	-	-

Hyper Parameter Tunning:

MODEL	RATE	HEADS	ACCURACY
1DCONV	0.01	NA	0.42
1DCONV	0.005	NA	0.456
1DCONV	**0.001**	**NA**	**0.575**
Transformer	0.001	1	0.509
Transformer	0.01	1	0.1257
Transformer	0.005	1	0.2757
Transformer	0.001	2	0.484
Transformer	**0.001**	**4**	**0.634**

Observations:

We Observed that best performance was given to us by our transformer model with heads = 4

Model Architecture	Accuracy	F1-Score	ROC-AUC
Conv 1	0.4625	0.4287	0.848
Transformer (Head = 1)	0.525	0.5182	0.9033
Transformer (Head = 2)	0.55	0.5353	0.9076
Transformer (Head = 4)	**0.675**	**0.6677**	**0.9384**