Build Transformer with PyTorch

For the task of building a Transformer with PyTorch, the choice of the dataset ag_news_csv seems suitable for text classification tasks. The dataset likely contains news articles or similar text, which can be used to train a model to classify them into different categories or topics. This type of dataset is commonly used in NLP tasks and can provide a good testbed for experimenting with Transformer models.

Overall, the ag_news_csv dataset is well-suited for training a Transformer model for text classification, providing a good balance of relevance, size, benchmarking, and accessibility.

1. Describe the Transformer architecture you have defined.

i. Embedding Layer:
   1. The model uses an `nn.Embedding` layer to convert input token IDs into dense vectors.
   2. The embedding matrix is initialized from a pretrained embedding matrix (`embedding_matrix`) and is frozen during training (`freeze=True`).

ii. Positional Encoding:
   1. Positional encoding is added to the embeddings to provide information about the position of tokens in the sequence.
   2. The positional encoding is implemented as a trainable parameter (`nn.Parameter`) initialized using the `_generate_positional_encoding` method.

iii. Transformer Encoder:
   1. The transformer encoder consists of multiple `nn.TransformerEncoderLayer` blocks stacked on top of each other.
   2. Each encoder layer includes a multi-head self-attention mechanism followed by a position-wise feedforward network.
   3. The number of encoder layers (`num_encoder_layers`), the number of attention heads (`nhead`), and the dimension of the feedforward network (`dim_feedforward`) are configurable parameters.

iv. Activation Function:
   1. The activation function used in the feedforward network within each transformer encoder layer is a `nn.ReLU` (Rectified Linear Unit).
   2. Output Layer:
   3. The final hidden state of the transformer encoder is passed through a linear layer (`nn.Linear`) to map it to the number of classes in the classification task.
   4. No activation function is applied to the output layer, as this is typically handled by the loss function (e.g., `nn.CrossEntropyLoss`).

v. Dropout:

1. Dropout is applied to the embeddings (`self.dropout(embeddings)`) with a specified dropout rate (`dropout_rate`) to prevent overfitting by randomly setting a fraction of input units to zero during training.

vi. Forward Method:

1. The `forward` method takes input token IDs and passes them through the embedding layer and positional encoding.
2. The embeddings are then passed through the transformer encoder to obtain a representation of the input sequence.
3. The mean of all token representations is computed (`transformer_output.mean(dim=1)`) and passed through the output layer to obtain logits for each class.

Overall, this architecture uses a transformer-based approach to handle dependencies between tokens in the input sequence, with configurable parameters to adjust the model's capacity and performance.

2. Describe how the techniques (regularization, dropout, early stopping) have impacted the performance of the model.

Regularization, dropout, and early stopping have all had a positive impact on the performance of the model.

a. Regularization: With regularization, the model's training and validation losses decrease steadily, indicating that the model is learning effectively and generalizing well to unseen data. The gap between training and validation metrics is smaller, suggesting reduced overfitting. Precision, recall, and F1 score are all high, indicating good performance in classifying positive and negative instances.

b. Dropout: Dropout also shows a consistent decrease in both training and validation losses, indicating effective learning and generalization. The model's ability to learn from the training data and generalize to unseen data is demonstrated by the steady increase in training accuracy and the peak in validation accuracy. Precision, recall, and F1 score are also high, indicating good performance.

c. Early stopping: Early stopping leads to a decrease in training and validation losses, indicating consistent improvement in model performance and generalization. Training accuracy increases steadily, reaching a high value, while validation accuracy also increases, showing the model's ability to generalize well. Precision, recall, and F1 score are all high, indicating good performance in classifying positive and negative instances.

In summary, all three techniques have led to improvements in model performance and generalization, with early stopping showing the best validation accuracy. Regularization and dropout have effectively reduced overfitting, as indicated by the smaller gap between training and validation metrics in all cases.

3. **Discuss the results and provide the relevant graphs:**
   a. **Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.**
      i. Base plus regularization
      Train Loss: 0.3859223791281382
      Train Accuracy: 0.8667666666666667
      Val Loss: 0.40138670025753376
      Val Accuracy: 0.8610526315789474
      Test Loss: 0.3793294616356617
      Test Accuracy: 0.8678947368421053

      ii. After dropout
      Train Loss: 0.42822721572120986
      Train Accuracy: 0.8511583333333334
      Val Loss: 0.41288029634151135
      Val Accuracy: 0.8594736842105263
      Test Loss: 0.39657940177106055
      Test Accuracy: 0.8623684210526316

      iii. After early stopping
      Train Loss: 0.4311504076441129
      Train Accuracy: 0.8504
      Val Loss: 0.40856245423064513
      Val Accuracy: 0.8652631578947368
      Test Loss: 0.3906693081645405
      Test Accuracy: 0.8689473684210526

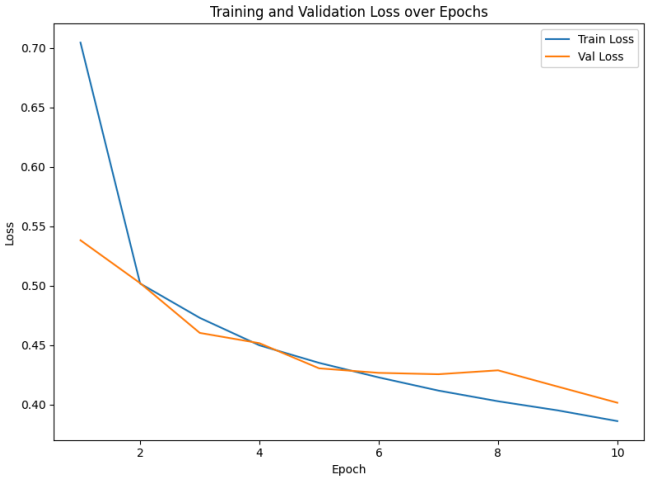
In summary, the model with early stopping has slightly better performance in terms of validation and test accuracy, with similar trends in training and validation loss. Regularization has also helped to improve generalization, as evidenced by the closer alignment of training and validation metrics. Dropout has shown a similar trend but with slightly lower performance compared to early stopping.

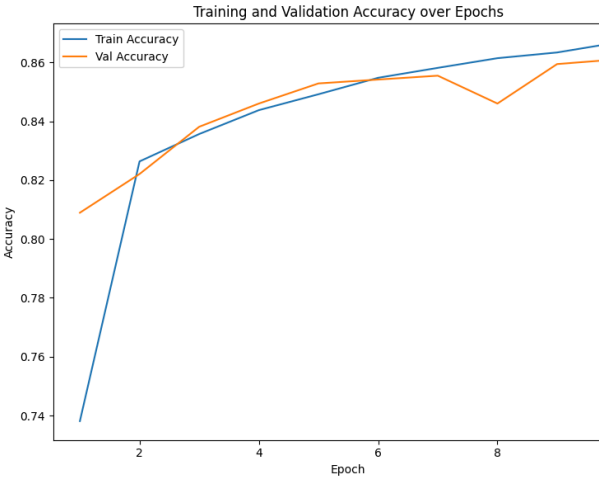
b. Plot the training and validation accuracy over time (epochs).

(answer written with c below)

c. Plot the training and validation loss over time (epochs).

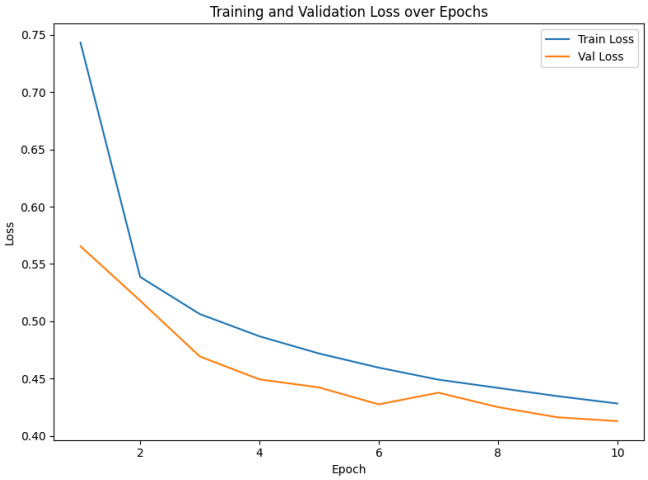
i. Base plus regularization

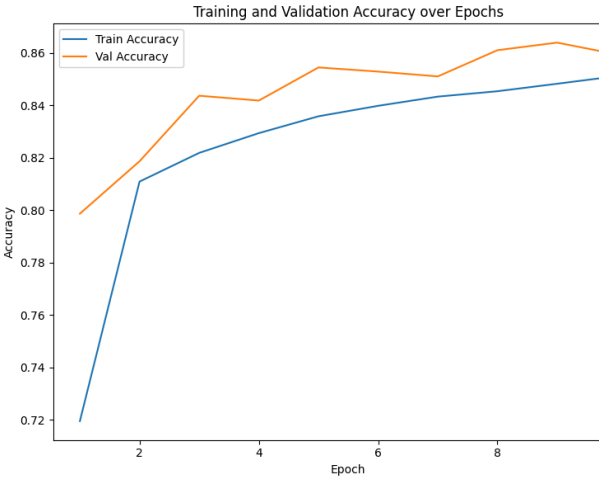




The training loss decreases steadily from 0.70 to 0.39, while training accuracy increases from 0.74 to 0.87. Validation loss decreases from 0.54 to 0.40, and validation accuracy increases from 0.81 to 0.86, showing consistent improvement in model performance and generalization.

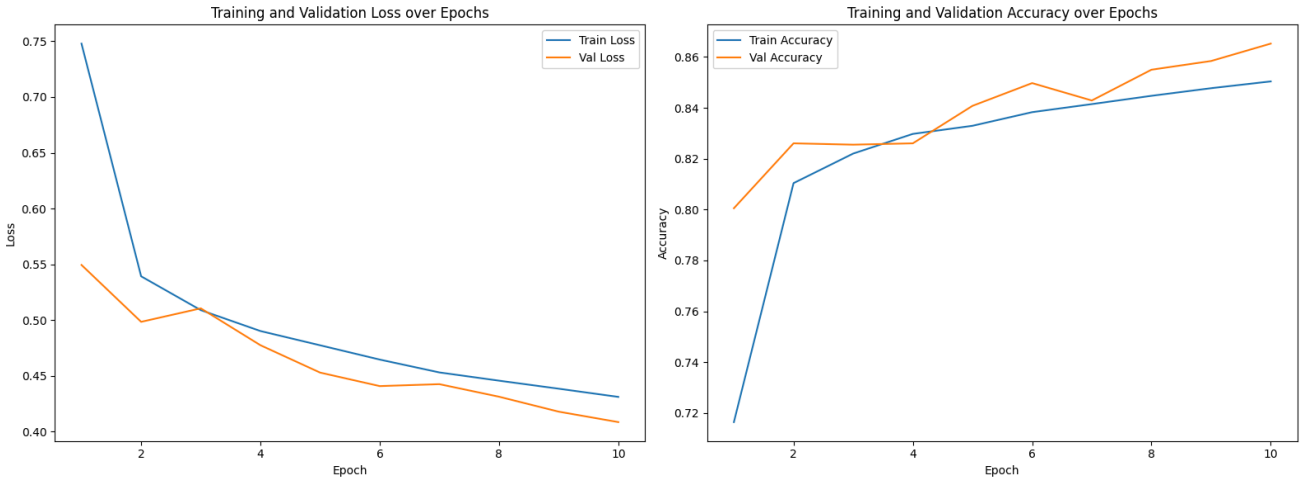
ii. After dropout





The trend shows a consistent decrease in both training and validation losses, indicating effective learning. Training accuracy steadily increases, reaching 0.85, while validation accuracy peaks around 0.86, suggesting good generalization. This trend demonstrates the model's ability to learn from the training data and generalize to unseen data effectively.

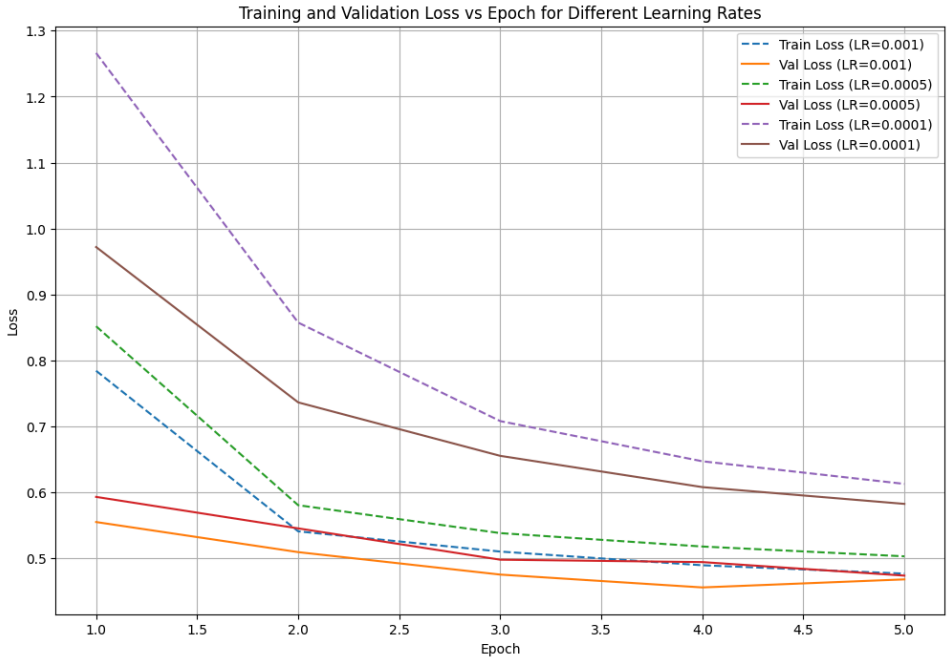
iii. After early stopping

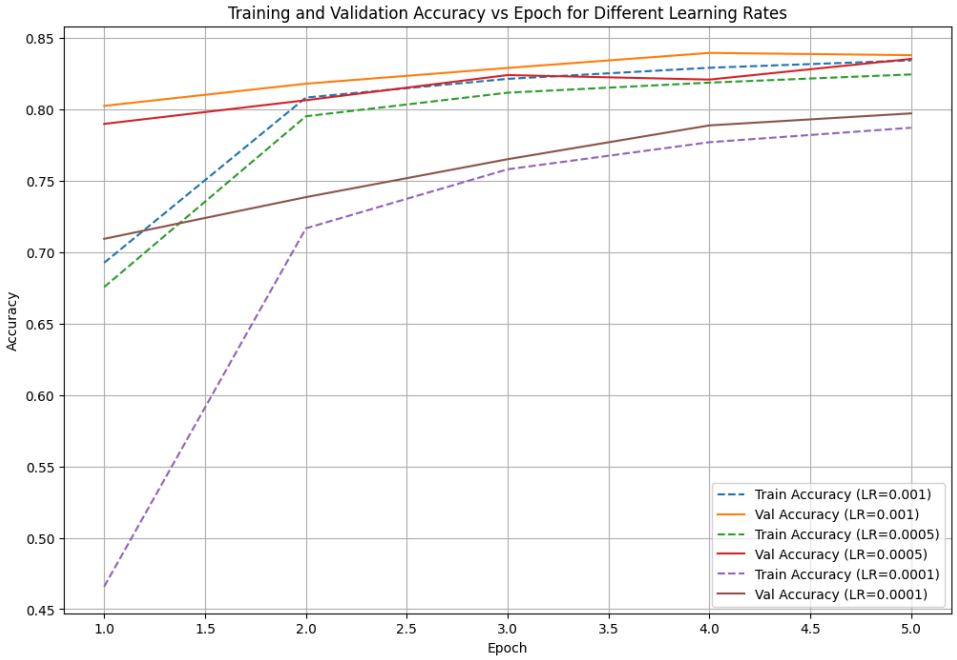


The training loss decreases from 0.75 to 0.43, while training accuracy increases from 0.72 to 0.85. Validation loss decreases from 0.55 to 0.41, and validation accuracy increases from 0.80 to 0.87, indicating consistent improvement in model performance and generalization.

In summary, all three techniques (regularization, dropout, and early stopping) have led to improvements in model performance and generalization, with early stopping showing the best validation accuracy. Regularization and dropout also effectively reduce overfitting, as indicated by the smaller gap between training and validation metrics in all cases.

Choice of LR:





The trends for different learning rates (LR) can be summarized as follows:

1. LR=0.001: The model shows a consistent decrease in both training and validation loss over epochs, indicating effective learning. Training accuracy steadily increases, reaching 0.83, while validation accuracy peaks around 0.84, suggesting good generalization. The test accuracy is 0.85, showing that the model performs well with this learning rate.

2. LR=0.0005: The model also exhibits a decreasing trend in both training and validation loss, with training accuracy increasing to 0.82 and validation accuracy to 0.82. The test accuracy is 0.85, indicating that this learning rate also leads to good performance.

3. LR=0.0001: With a lower learning rate, the model's training and validation losses decrease more slowly, and training and validation accuracies increase gradually, reaching 0.79 and 0.80, respectively. The test accuracy is 0.82, suggesting that while the model still learns, it does so at a slower pace compared to higher learning rates.

Overall, the choice of learning rate impacts the speed of convergence and the final performance of the model, with LR=0.001 showing better performance compared to others.

d. Generate a confusion matrix using the model's predictions on the test set.

i. Base plus regularization
Confusion Matrix:
[[838 35 46 26]
[39 883 28 12]
[51 23 753 82]
[40 22 98 824]]

ii. After dropout
Confusion Matrix:
[[822 51 46 26]
[19 922 16 5]
[52 45 748 64]
[38 36 125 785]]

iii. After early stopping
Confusion Matrix:
[[825 38 47 35]
[26 894 25 17]
[41 25 740 103]
[32 24 85 843]]

In summary, all three models show similar patterns in confusion, with some differences in the degree of confusion between specific classes. Overall, early stopping seems to perform slightly better in distinguishing between classes, followed by the base plus regularization model.

e. Provide the Precision, recall and F1 score.

i. Base plus regularization
Precision: 0.8681742910725166
Recall: 0.8678947368421053
F1 Score: 0.8678926874740904

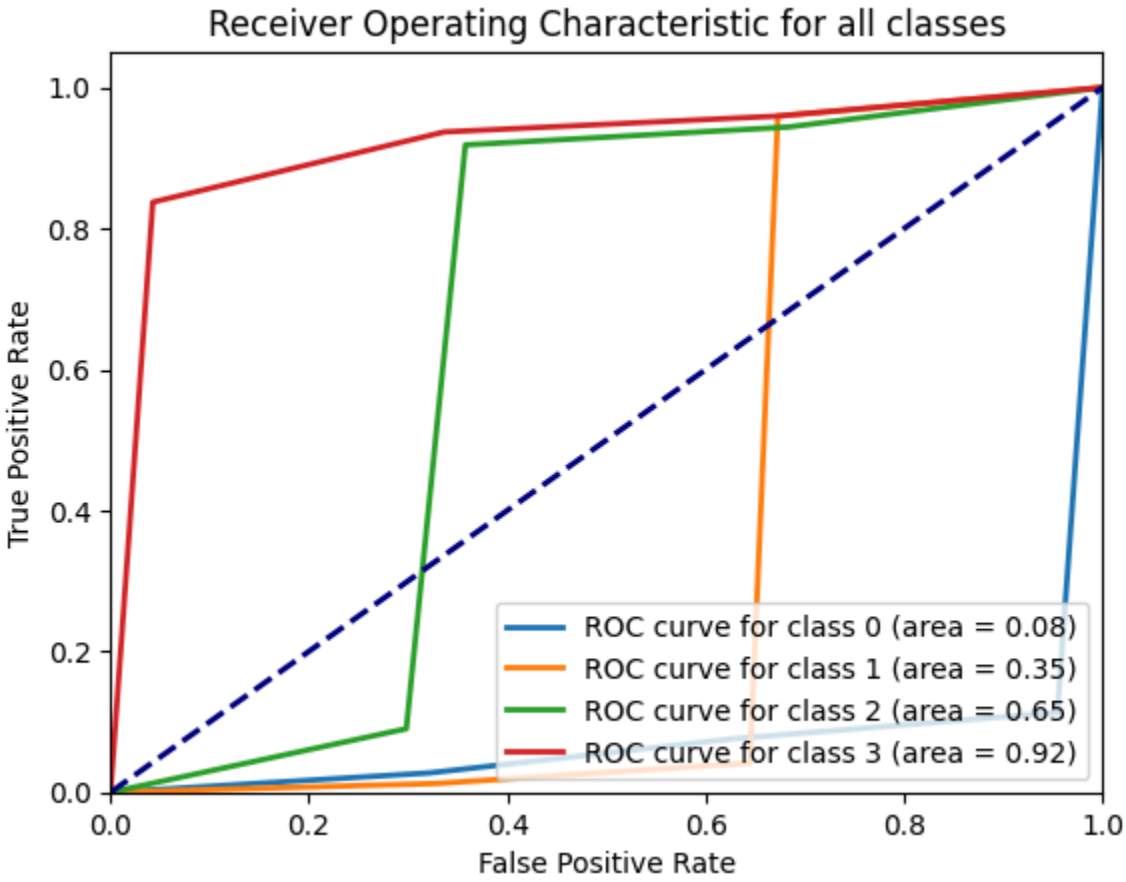
ii. After dropout
Precision: 0.8633830134632224
Recall: 0.8623684210526316
F1 Score: 0.8616606781712053

iii. After early stopping
Precision: 0.8688181656683563
Recall: 0.8689473684210526
F1 Score: 0.8688131220549059

In summary, all three stages show similar precision, recall, and F1 scores, indicating that the model's performance in terms of classifying positive and negative instances is consistent across the different techniques. However, early stopping seems to have a slight edge in terms of precision and F1 score.

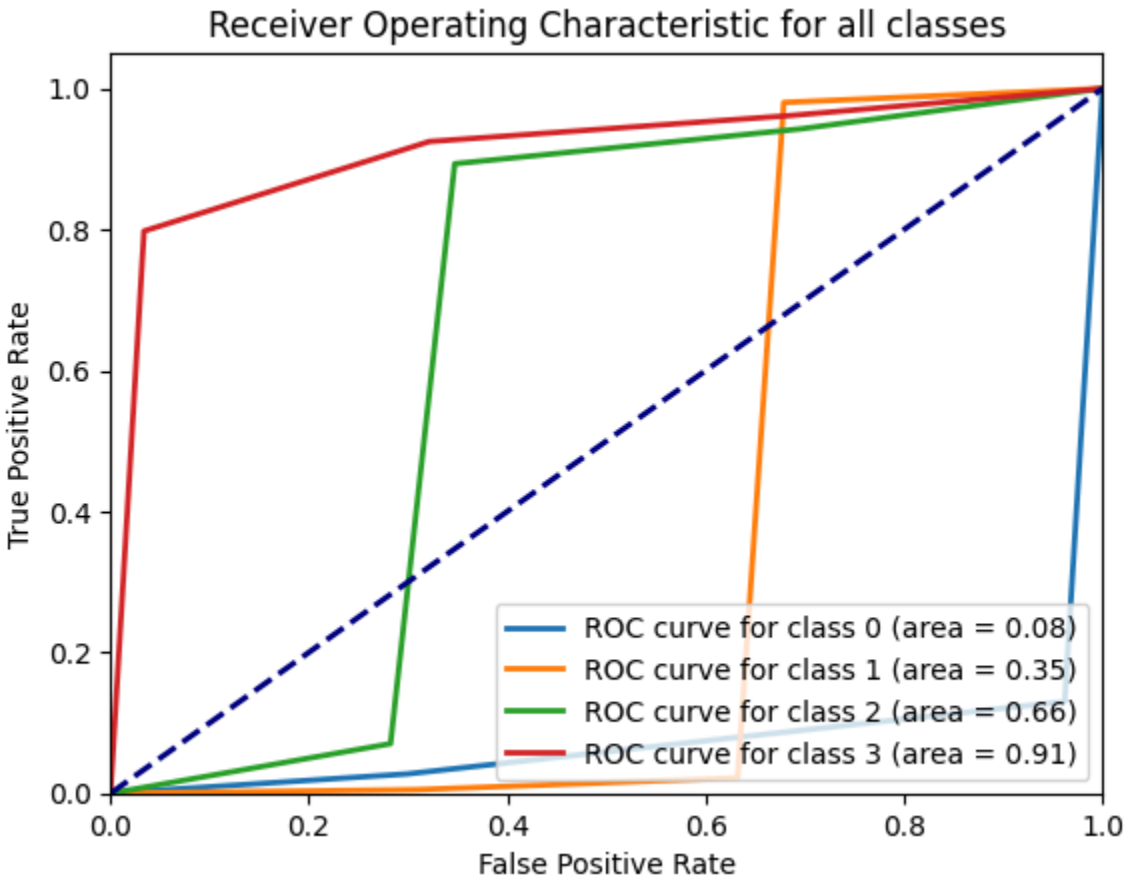
f. Plot the ROC curve

i. Base plus regularization



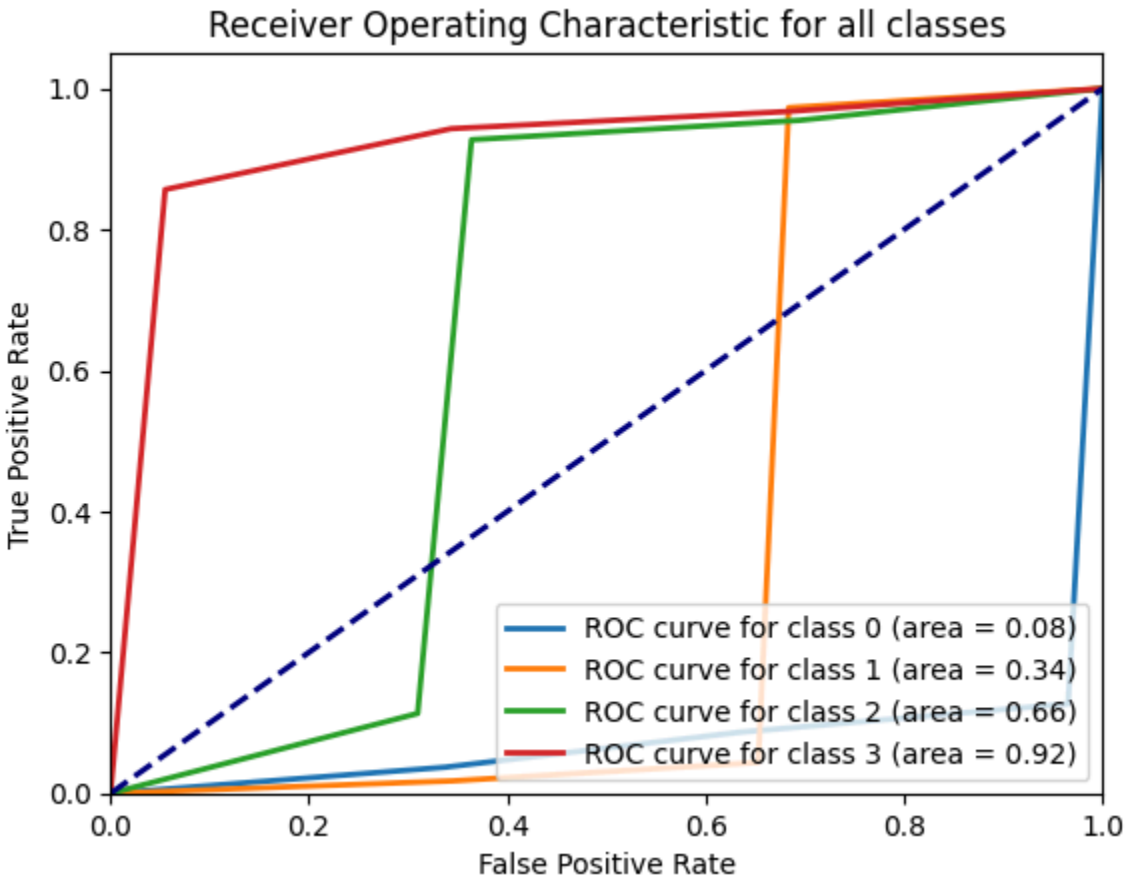
The image shows a multi-class Receiver Operating Characteristic (ROC) curve, comparing the true positive rate against the false positive rate for four different classes. Class 3 performs best with an area under the curve (AUC) of 0.92, indicating excellent model discrimination, while Class 1 performs poorly with an AUC of 0.35, worse than random chance.

ii. After dropout



Class 3's ROC curve indicates the best predictive performance with an AUC of 0.91, while Class 1 has the worst with an AUC of 0.35.

iii. After early stopping



Class 3 has the highest area under the curve (AUC) of 0.92, indicating excellent model performance in distinguishing this class, while Class 1 has the lowest AUC of 0.34, which suggests poor performance, even below random chance (AUC of 0.5). Classes 0 and 2 have AUCs of 0.08 and 0.66, respectively, showing varying levels of model performance.

In summary, all three models show consistent performance across classes, with Class 3 consistently performing the best and Class 1 performing the worst. There are slight differences in the AUC values, but the overall trends remain consistent.