Assignment 2 - Autoencoder and Transformer Architectures

> **Instructor: Alina Vereshchaka**

Team Contribution

Team Member	Assignment Part	Contribution (%)
Charvi Kusuma	I, II, III, IV, Bonus	50, 50, 50, 50, 50
Tarun Reddi	I, II, III, IV, Bonus	50, 50, 50, 50, 50

Part I: Theoretical Part – Autoencoders for Anomaly Detection in Manufacturing [10 points]

In manufacturing industries, ensuring the quality of products is important. Detecting anomalies or defects in the production process can be challenging, especially in large-scale operations. Traditional methods often rely on manual inspection or rule-based systems, which can be time consuming and prone to human error.

Scenario: A manufacturing company "ElectroGuard Innovations" produces electronic components, and they want to improve their quality control process by implementing an automated anomaly detection system. The company collects sensor data from the production line, which includes various parameters such as temperature, pressure, and voltage readings. Anomalies in these readings could indicate potential defects in the components.

Objective: To develop an automated anomaly detection system using an autoencoder neural network to identify defects in the manufacturing process.

Autoencoder Architecture:

- Input layer: 1000-dimensional vectors
- Hidden layer 1: Fully connected layer with 512 units and ReLU activation
- Bottleneck layer: Fully connected layer with 32 units
- Hidden layer 2: Fully connected layer with 512 units and ReLU activation
- Output layer: Fully connected layer with 1000 units and sigmoid activation
- Autoencoder is trained using MSE loss

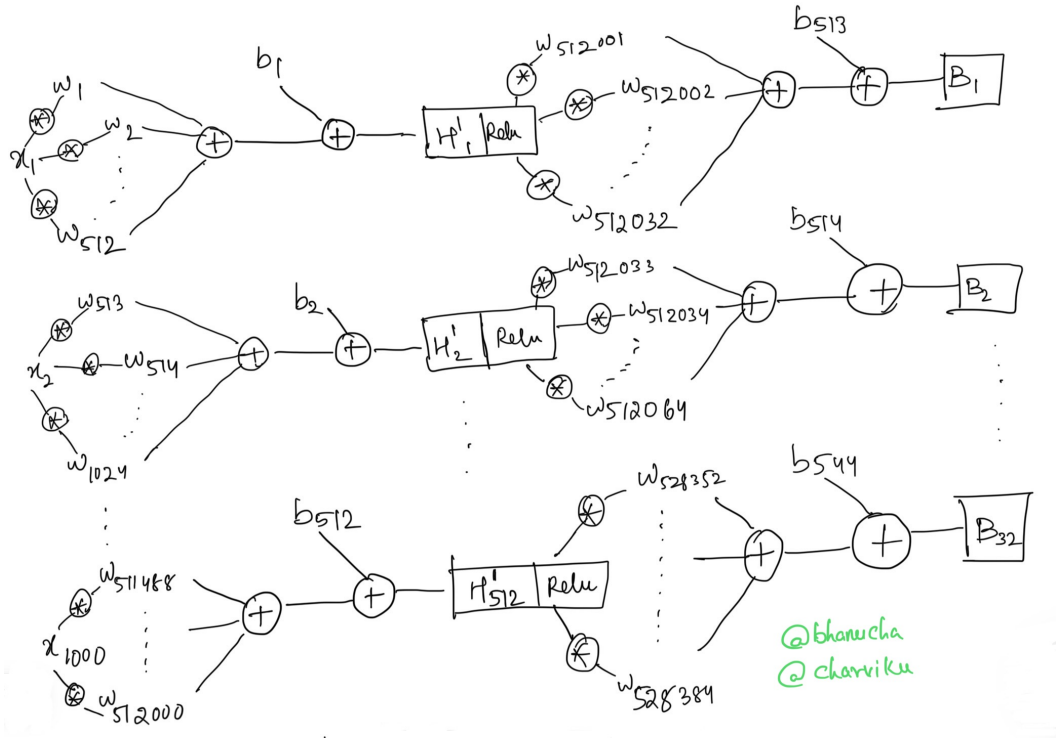
TASKS:

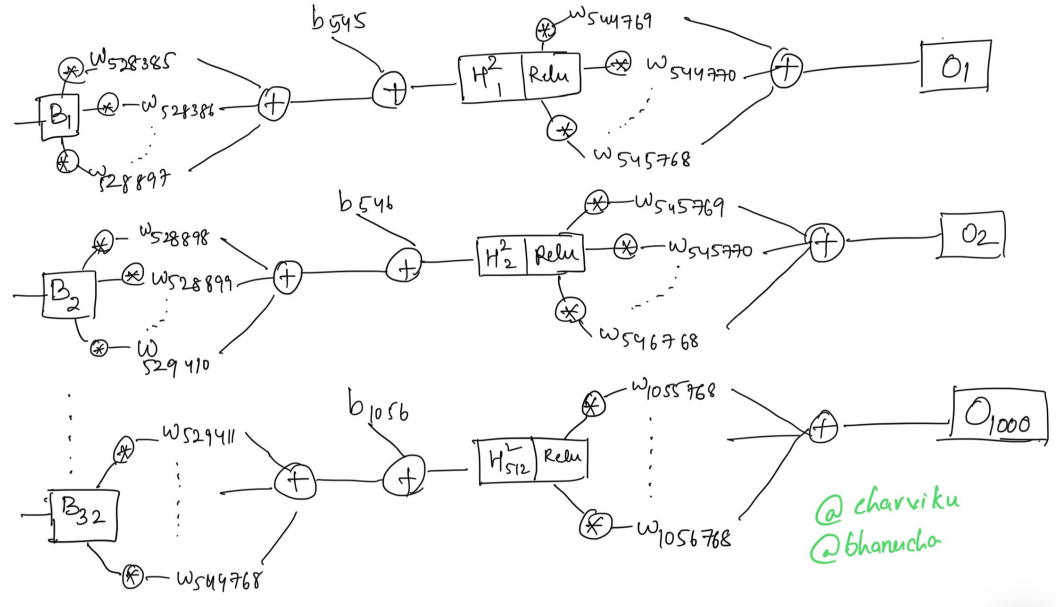
1. Calculate the total number of parameters in the autoencoder, including weights and biases.

- **Input to Hidden Layer 1**: $1000 \times 512 + 512 = 512512$
- **Hidden Layer 1 to Bottleneck**: $512 \times 32 + 32 = 16416$
- **Bottleneck to Hidden Layer 2**: $32 \times 512 + 512 = 16896$
- **Hidden Layer 2 to Output Layer**: $512 \times 1000 = 512000$

  Total = 1,057,824

2. Generate a computational graph for the autoencoder.





3. Discuss potential challenges and limitations (at least 4) of using autoencoders for anomaly detection in manufacturing.

- Autoencoders will mimic their input, and if training data includes errors, the model will also treat these errors as normal, making it hard to spot real problems. Getting clean data without any errors is hard in complicated factory settings.

- In factories, problems can be minor yet complex which need deeper knowledge of how things are made to spot them. Autoencoders might not catch these tiny differences without extra context.
- How well autoencoders work depends on how accurately the input data describes the real situation. In factories where data from sensors can be messy or complex it's tough to find a way to clearly show what's normal and what's not.
- Factory conditions change over time, like different materials or wear and tear on machines. An autoencoder trained under one set of conditions may fail to recognize new, different conditions, leading to incorrect alarms or missed detections.

4. Propose potential improvements or extensions (at least 4) to the system to enhance its effectiveness in detecting anomalies.

- Mix in what experts know about making things to help the system better understand which changes matter.
- Teach the system using a few examples of both normal and problem cases so it can learn the difference better.
- Use different approaches together to catch more types of problems since each method might see things a bit differently.
- Keep teaching the system with new information so it can understand and adapt to changes in how things are made over time.

Part II: Autoencoders for Anomaly Detection [30 points]

Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise?

- **Dataset Name:** Numenta Anomaly Benchmark (NAB), in that the realTweets/ subset focused on Apple data.
- **Nature of Data:** This dataset contains Twitter mentions of Apple, a large publicly traded company. The mentions are quantified over specific time intervals.
- **Data Type and Structure:**
  - The dataset comprises **time-series data**, tracking the volume of Twitter mentions over time.
  - It consists of two variables: timestamp and value.
    - timestamp indicates the date and time of the data entry, initially loaded as an object type and then converted to datetime64 for proper time series analysis.
    - value represents the number of mentions, recorded as an integer (int64).
- **Size and Scale:** The dataset contains **15,902 entries**, indicating a considerable volume of data for analysis.
- The value variable has a mean of approximately 85.55, with a standard deviation of 321.05, indicating a wide variance in the number of Twitter mentions.
- The minimum number of mentions recorded is 0, and the maximum is 13,479, highlighting potential peaks of high activity or popularity also as anomaly.
- Quartiles indicate that 25% of the data has 29 mentions or fewer, 50% (median) has 47 mentions or fewer, and 75% has 76 mentions or fewer.

- There are no missing values in the dataset for either timestamp or value, indicating good data completeness.

Visualizations

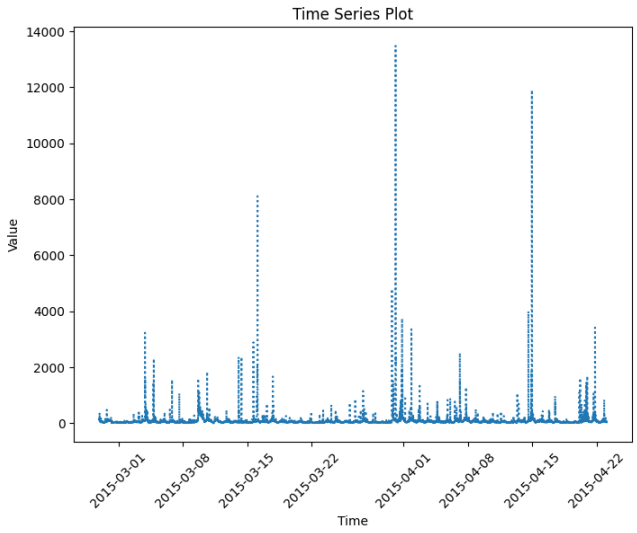
Time series plot displaying the number of Twitter mentions for Apple over time.

Frequent Low Values: The majority of data points are low, indicating typical mention counts.

Sporadic High Spikes: Occasional spikes suggest atypical events or high-interest occurrences.

No Clear Patterns: A cursory look does not reveal obvious daily or weekly trends.

Dense Data Representation: A dotted line style effectively demonstrates the density of data points.

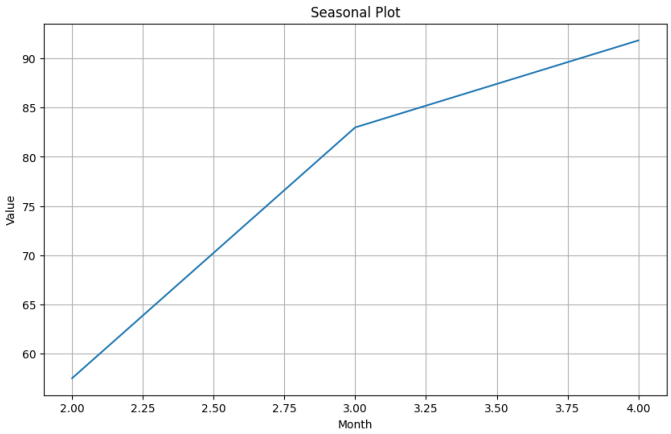


The seasonal plot indicates a gradual increase in Twitter mention values over the months:

Ascending Trend: There is a clear upward trajectory from February to April.

No Anomalies Visible: The plot does not display any abrupt changes or spikes.

Simple Monthly Pattern: The visualization suggests a possible seasonal trend with a steady month-over-month rise.

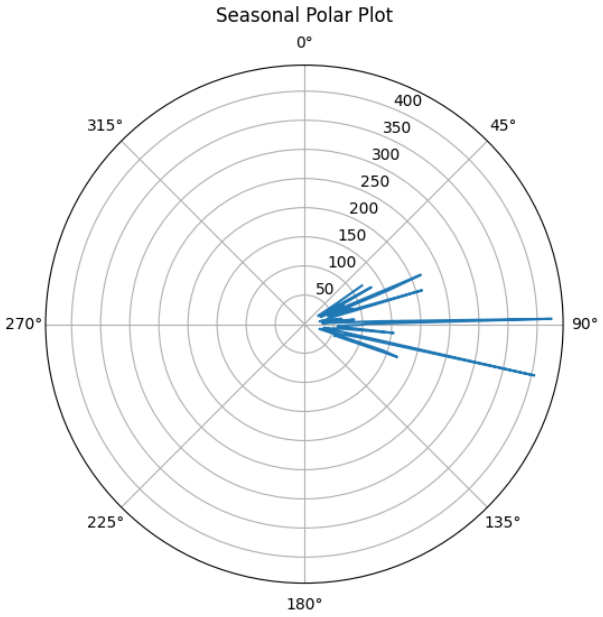


The seasonal polar plot presents data points plotted against angles in a circular format:

Radial Distribution: The data points are distributed across different angles, suggesting a time-based pattern (likely daily).

Concentration of Data: The plot shows most values at lower radii, with no significant outliers.

Uniformity: There is a lack of pronounced peaks or troughs, implying relatively stable values over the observed cycles.



Describe the details of your autoencoder models, including the layers, activation functions, and any specific configurations employed.

Autoencoder Model Descriptions:

- **Common Features Across All Models:**
  - All models are designed for input with a shape specified by input_shape = (30, 1), aimed to encoding and decoding data to identify patterns or anomalies.
  - Each model architecture includes an encoder part that reduces the dimensionality, and a decoder part that reconstructs the data.
  - All models are compiled with the Adam optimizer and mean squared error (MSE) as the loss function.
- **Base Model**
  - We follow a base structure with dense layers in both encoder and decoder parts.
  - The main differences we will create will be the activation functions used in the layers and applying LSTM and increasing number of layers.
- **Version 1 and 2 Model:**
  - We increased the dense layers and parameters compared to the base model, aimed at a more better representation and to check the result variations when increased.
  - Changed the activation functions with ReLU, Sigmoid, Tanh
- **Version 3 Model:**
  - Utilizes LSTM (Long Short-Term Memory) layers, suitable for time-series data, offering a distinct approach from dense layers.
  - Incorporated RepeatVector to match the encoder's output shape with the decoder's input requirement.

Comparative Table of Version Models:

Feature / Model	Base Model	Version 1	Version 2	Version 3
Layer Types	Dense	Dense	Dense	LSTM, TimeDistributed
Activation Functions	ReLU, Sigmoid	ReLU, Sigmoid	ReLU, Sigmoid, Tanh	ReLU
Total Parameters	588 (2.30 KB)	1,356 (5.30 KB)	1,356 (5.30 KB)	4,193 (16.38 KB)
Complexity	Low	Moderate	Moderate	High
Distinct Features	Simplified structure	Increased Parameters	Variation in activation functions	LSTM layers for sequence data

Discuss the results and provide relevant graphs:
a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
b. Plot the training and validation accuracy over time (epochs).
c. Plot the training and validation loss over time (epochs).
d. Generate a confusion matrix using the model's predictions on the test set.
e. Report any other evaluation metrics used to analyze the model's performance on the test set.

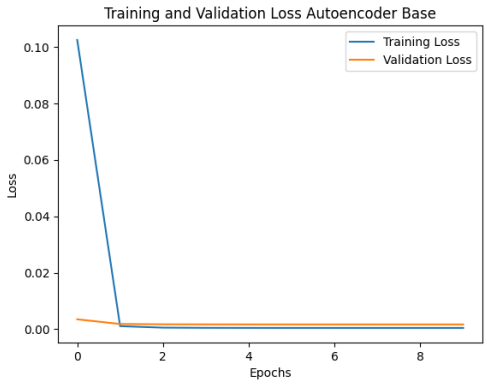
Base autoencoder model for detecting anomalies in time-series data:

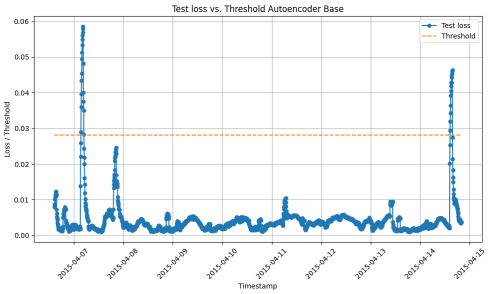
- Training Loss: Reduced significantly from 0.1225 to 0.00046 over 10 epochs.
- Validation Loss: Decreased and plateaued at approximately 0.0017.
- Testing Loss: Very low at 0.000117, indicating a good fit on the test data.
- Number of Anomalies detected: 32

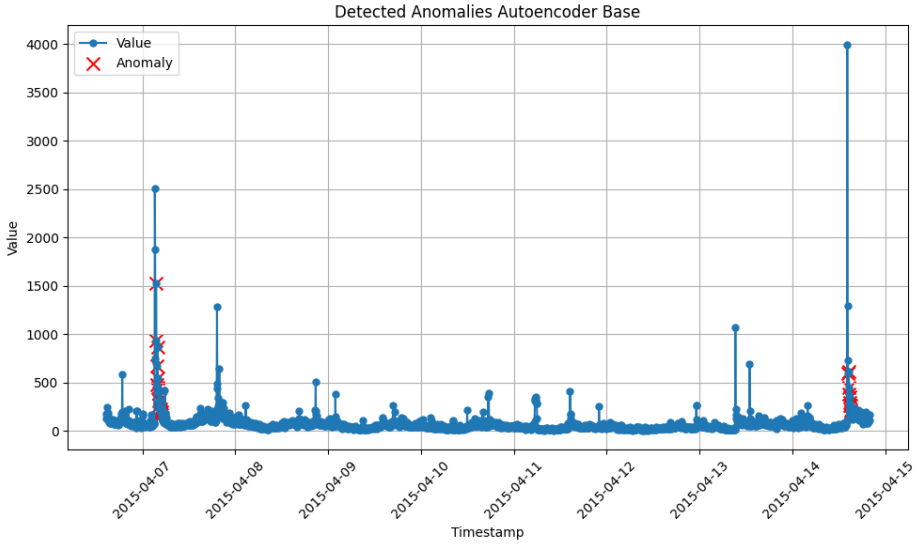
Plot of Training and Validation Accuracy:
Not Applicable

Plot of Training and Validation Loss Over Time (Epochs):

- The plot shows the loss decreasing sharply for the training data and more gradually for the validation data. It indicates that the model is learning and generalizing well since the validation loss follows the training loss downwards.







Other Evaluation Metrics and Analysis:

- Test Loss vs. Threshold Plot: The second on the right plot illustrates test loss over time with a threshold line. Points above the threshold are potential anomalies.
- Detected Anomalies Plot: The third plot shows the actual values over time, with anomalies marked in red. This visualization directly identifies which points are considered anomalies based on the model's loss exceeding the threshold.

Autoencoder Version 1 for anomaly detection

- Training Loss: Started at 0.0562 and decreased to 0.0004429 over 10 epochs.
- Validation Loss: Decreased slightly from 0.0018 to 0.0016788, remaining stable after the initial drop.
- Testing Loss: Reported as 0.00010596, indicating the model's good performance on unseen data.
- Number of Anomalies detected: 31

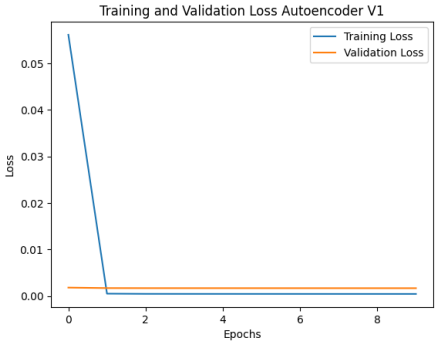
Plot of Training and Validation Accuracy:
Not Applicable

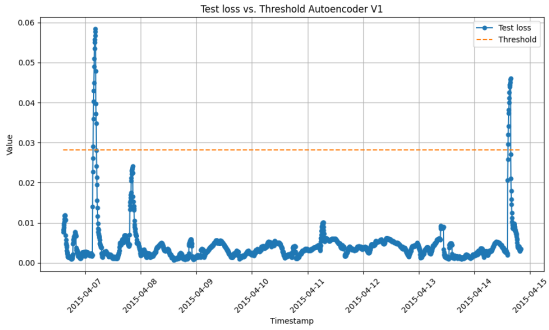
Plot of Training and Validation Loss Over Time (Epochs):

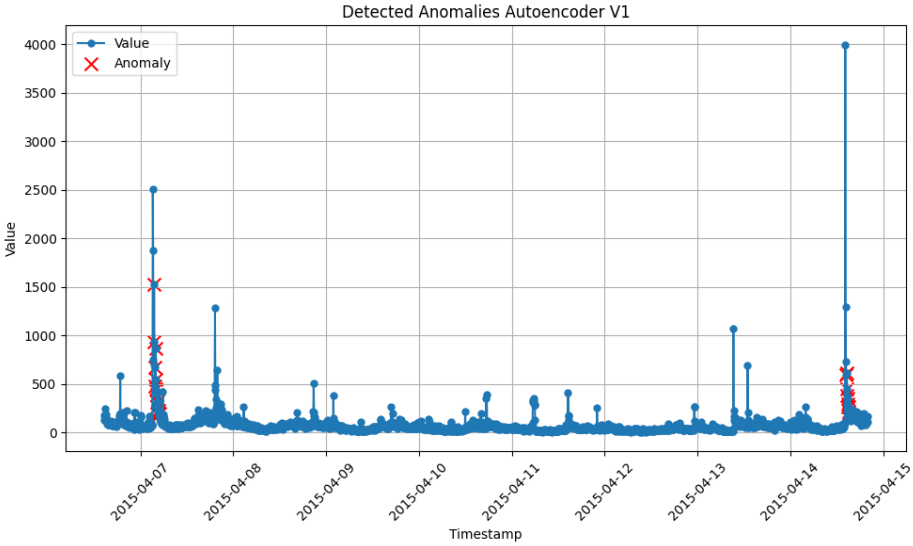
- The first image below on the left shows the training and validation loss over epochs for the autoencoder. Both losses decrease sharply and then plateau, indicating that the model is learning and then stabilizing as it converges to a minimum loss value.

Other evaluation metrics used to analyze the model's performance on the test set:

- Anomaly Detection: The third image shows us the detected anomalies based on the test loss compared to a threshold. Anomalies are marked with red 'X' symbols where the test loss exceeds the threshold.
- Threshold Analysis: The second image visualizes the test loss versus the threshold. The test loss spikes correspond with the anomalies detected in the third image.







Autoencoder Version 2 for anomaly detection

- Training Loss: Began at approximately 0.0027 and settled around 0.000475 after 10 epochs.
- Validation Loss: Hovered around 0.0017, indicating stability after the initial descent.
- Testing Loss: A low value of 0.00012878, showing good model performance on the test set.
- Number of Anomalies detected: 69

Plot of Training and Validation Accuracy Over Time (Epochs):

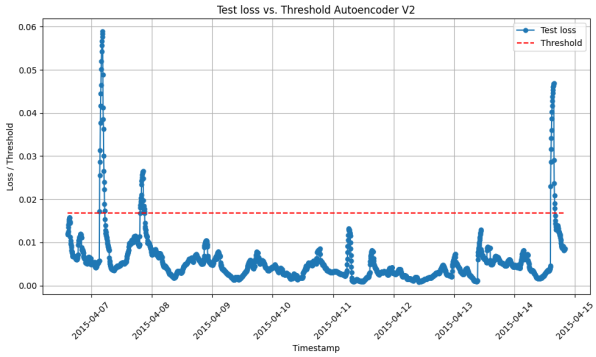
Not Applicable

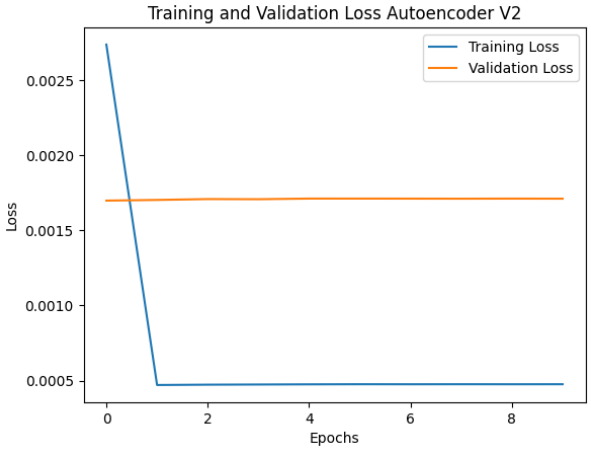
Plot of Training and Validation Loss Over Time (Epochs):

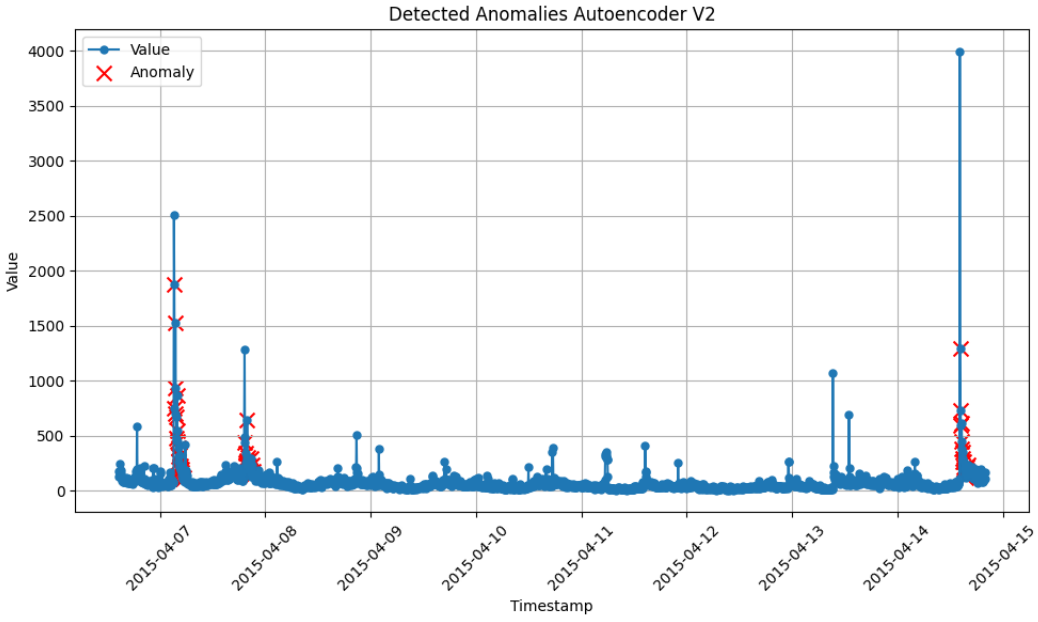
- The first image below on the left displays the training and validation losses over epochs, with both decreasing and then plateauing, suggesting that the model is learning and then stabilizing as it minimizes loss. However here in this version, validation loss is not at par with the training loss, signs of not able to generalize on unseen data that well.

Other evaluation metrics used to analyze the model's performance on the test set:

- Test Loss vs. Threshold Plot: The second image illustrates how the test loss compares with a set threshold. This threshold is used to determine whether a data point is anomalous.
- Detected Anomalies Plot: The third image shows the actual values and detected anomalies over time. The anomalies are the points where the test loss exceeds the threshold, highlighted with red 'X' marks.
- The anomalies are plotted on the original value scale after inverse transformation, providing a clear visual representation of where anomalies occur relative to normal values.







Autoencoder Version 3, which incorporates LSTM layers to process the time series data:

- Training Loss: Remained approximately constant around 0.000475 throughout training.
- Validation Loss: Remained steady at 0.0017, indicating no overfitting.

- Test Loss: Reported as 0.00012878, signifying that the model performs similarly on both seen (training) and unseen (test) data.
- Number of Anomalies detected: 69

Plot Training and Validation Accuracy:

Not Applicable

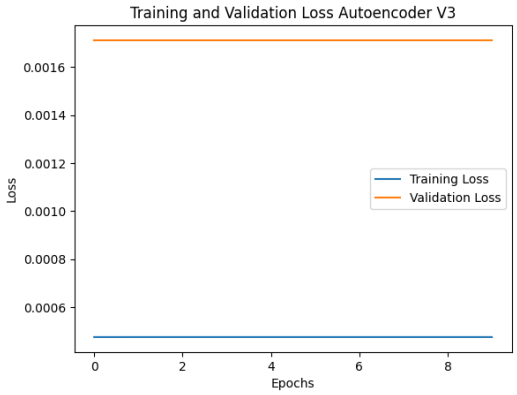
Plot Training and Validation Loss:

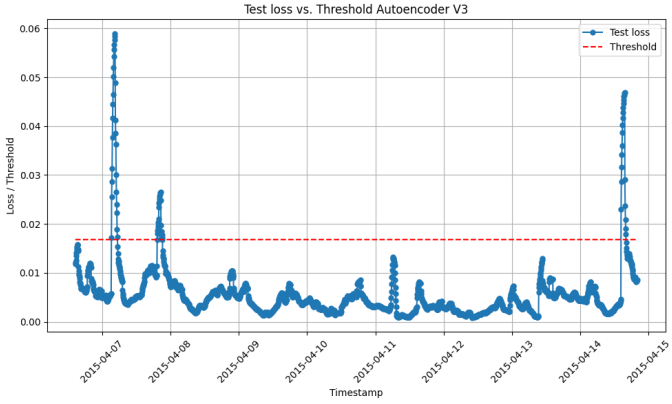
- The first image showcases that the training and validation loss remain almost constant after an initial descent in training loss. This could suggest that the model quickly reached a plateau where further training did not yield significant improvements.

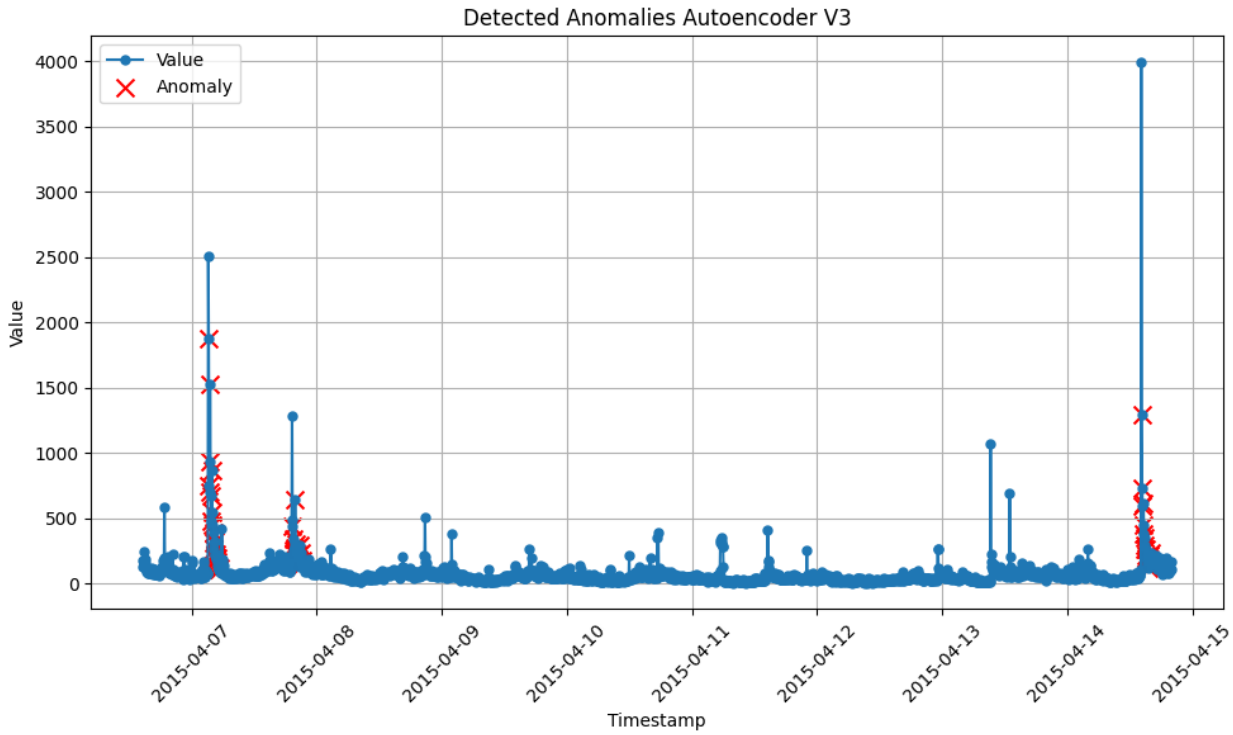
Other Evaluation Metrics and Analysis:

- Test Loss vs. Threshold Plot: In the second image, the test loss mostly stays below the threshold, with spikes crossing the threshold line indicating potential anomalies.
- Detected Anomalies Plot: The third image visualizes the values and anomalies detected over time. Anomalies are marked with red 'X' symbols at points where test loss exceeds the threshold.
- The anomalies are clearly identifiable, and the threshold appears to successfully separate the normal data points from the outliers.

The model's training and validation losses suggest that the LSTM autoencoder has learned a representation of the data but did not continue to improve after the initial epochs. This could be due to the simplicity of the task for the LSTM's capabilities or the need for further tuning of the model architecture and hyperparameters.







Comparison of All Models:

Validation Loss: This is a direct measure of how well the model is reconstructing the validation set data. A lower validation loss indicates better reconstruction performance.

The LSTM-based architecture (V3) seems to have the highest steady-state validation loss, implying that it might not be reconstructing the validation data as accurately as the other versions.

Autoencoder V1 and V2 have lower and similar validation losses compared to the base model, suggesting better reconstruction ability on the validation set.

- **Autoencoder Base**: The validation loss decreases significantly at the start and then plateaus.
- **Autoencoder V1**: There's a steep drop in validation loss initially, followed by a plateau.
- **Autoencoder V2**: The validation loss starts low and remains roughly constant.
- **Autoencoder V3 (LSTM-based)**: The validation loss remains constant across epochs after an initial descent.

Discuss the strengths and limitations of using autoencoders for anomaly detection.

- Autoencoders excel at learning compressed, dense representations of data, making them effective at identifying outliers or anomalies that deviate from the norm.
- They can learn to detect anomalies in an unsupervised manner, which means they do not require labeled data to identify abnormalities. This is beneficial in domains where anomalies are rare or labeling is costly.
- Autoencoders can be applied to a wide range of data types, including images, videos, and structured data, making them versatile tools for various anomaly detection tasks.

- The performance of autoencoders heavily depends on the choice of hyperparameters (like the number of layers, the size of the bottleneck, etc.), which can be difficult to tune optimally without extensive experimentation.
- While efficient in detecting global anomalies, they might struggle with more subtle, local anomalies that do not significantly alter the overall data structure.
- Autoencoders are designed to minimize reconstruction error, which may lead to cases where anomalies are reconstructed well enough that they're not flagged as anomalies, particularly if the anomaly is similar to normal data patterns.

Part III: Theoretical Part – Transformers [20 points]

In this part of our assignment, we explore the theoretical foundation behind Transformers.

TASKS:

1. **Break down the mathematical operations involved in self-attention. Explain how the input sequence x = ($x1$,$x2$,…,xN) is processed through these stages:**

- **Linear transformations. Describe how three linear transformations project the input sequence into:**
  - **Query (q): this vector plays the role of "asking questions" about the sequence elements.**
  - **Key (k): this vector helps determine which elements in the sequence are relevant for answering the queries.**
  - **Value (v): this vector contains the actual information from each element in the sequence. Briefly explain how these transformations are typically performed using weight matrices (denoted by Wq ,Wk,and Wv) and bias vectors (optional).**
- **Scaled dot-product attention. Explain how the model calculates attention scores using the query (qi) and key (kj) vectors. Include the formula for this step and discuss the role of the square root of the key vector dimension (dk) in the normalization process.**
- **Weighted sum. Describe how the calculated attention scores are used to weight the value vectors (vj). Essentially, this step focuses on the relevant elements in the sequence based on the attention scores. Explain the mathematical formula for this weighted sum.**
- **Optional output transformation: Explain the purpose of the optional final linear transformation (Wo) and bias term (bo) in generating the latent representation z. How does this step potentially impact the final representation?**

$$x = [x_1, x_2, x_3 \dots x_N]$$

i) Linear Transformations :

- Query (q) will have its own weight vector W_q

$$q_i = W_q x_i + b_q$$

$$q_1 = w_q x_1 + b_q$$
$$q_2 = w_q x_2 + b_q$$
$$\vdots$$
$$q_N = w_q x_N + b_q$$

- key (k) will also have its own weight vector W_k

$$k_i = w_k x_i + b_k$$

$$k_1 = w_k x_1 + b_k$$
$$k_2 = w_k x_2 + b_k$$
$$\vdots$$
$$k_N = w_k x_N + b_k$$

@ bhamucha
@ charvi ku

- Value (v) will have W_v as its weight vector.

$$v_i = w_i x_i + b_i$$

$$v_1 = w_i x_1 + b_i$$
$$v_2 = w_i x_2 + b_i$$
$$\vdots$$
$$v_N = w_i x_N + b_i$$

2) Scaled dot-product attention

- Calculation of attention scores for k_i and q_i.

$$\text{Score}(q_i, k_i) = \frac{e^{q_i \cdot k_i}}{\sum_i e^{q_i k_i}}$$

For scaling scores will be done with $\frac{q_i \cdot k_i}{\sqrt{d_k}}$

The scores are calculated for every key particular to one query.

We take the dot products of query and key in the exponents

This is basically Softmax accross all these attention scores

* These scores are scaled down by dividing $\sqrt{d_k}$ where d_k will be the dimension of the key vectors

* It is used to prevent from having extremely small gradients in softmax.

Finally, $\text{Attention}(q, k, v) = \text{softmax}\left(\frac{q \cdot k}{\sqrt{d_k}}\right) \cdot V$

@bhanucha
@chaavitku

3) Weighted Sum

* Basicaly multiplying scores with values v_i creating a weighted sum of value vectors.

$$z_i = \sum_j \text{Attension}(q_i, k_j) v_j$$

Attension(q_i, k_j) is the output after softmax, and j is the sequence number for the input sequence.

@bhanucha
@charviku

4) Output Transformations:

W_0 and b_0 will be used to transform the $z = (z_1, z_2, \cdots z_N)$ vector like:

$$O_i = W_0 z_i + b_0$$

$$O_1 = W_0 z_1 + b_0$$

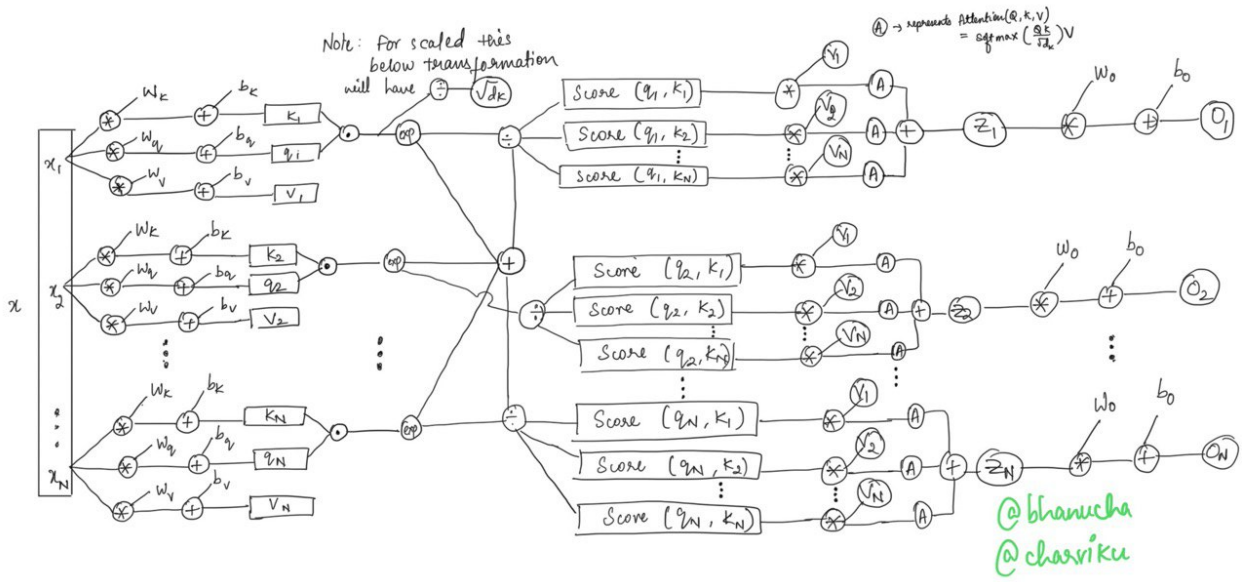
$$O_2 = W_0 z_2 + b_0$$

$$\vdots$$

$$O_N = W_0 z_N + b_0$$

@bhanucha
@charviku

2. **Draw the computational graph that depicts the flow of data through the self-attention mechanism. Include all the transformations mentioned in Question 1.**



Part IV: Build Transformer with PyTorch [40 points]

Dataset

The AG News dataset is a collection of news articles gathered from more than 2,000 news sources by ComeToMyHead in more than 1.4 million articles. We used this because it is commonly used for benchmarking text classification and natural language processing models. The dataset is typically divided into four categories: World, Sports, Business, and Science/Technology.

Describe the Transformer architecture you have defined

- A simple Transformers architecture with two main: a Block of the Transformer and a layer for token and position embedding.
- Transformer Block: In this layer implementation a simplified version of the Transformer's encoder block with a mutliple self-attention mechanism MultiHeadAttention that allows the model to focus on different parts of the input sequence simultaneously. The output of the attention mechanism is then passed through two layers of normalization with LayerNormalization, having a feed-forward network, consisting of two dense layers with a ReLU activation in between. This setup needs the model to process and transform the sequence data effectively.
- Token and Position Embedding: This layer converts input tokens into dense vectors of a specified embedding dimension. By first transforming token indices into embeddings through an embedding layer. Positional information is taken by generating positional embeddings and adding these to the token embeddings.

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

Base Model

- We got high training accuracy and low loss which indicates the model learned the training data well. But there is a significant difference on unseen data, where lower validation and testing accuracy compared to the training accuracy, suggested overfitting. So we did the following:

Dropout Model

- Slightly decrease in training accuracy with introduction of dropout layers which likely reduced overfitting by randomly dropping units. BUT increased validation and testing accuracy shows that dropout helped the model generalize better than the base model.

Dropout + Early Stopping Model

- Early stopping further reduced overfitting by halting training before the model could memorize the training data. THis significantly increased testing accuracy and reduced loss. This model outperformed the previous versions in generalization to unseen data.

Dropout + Early Stopping + Regularized Model

- Slightly reduced overfitting compared to previous model where regularization penalizes large weights, further addressing overfitting. High testing accuracy but slightly lower than the early stopping model alone which indicates that while regularization helps prevent overfitting, the model's ability to generalize to unseen data might have slightly decreased compared to the Dropout + Early Stopping model. This could be due to the strength of regularization possibly being too high. However all of this is better than Base model.

The Dropout + Early Stopping model demonstrated the best balance between high accuracy and generalization to unseen data, suggesting that while regularization is beneficial, finding the optimal regularization strength is crucial to maximize performance.

Discuss the results and provide the relevant graphs

- **Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.**

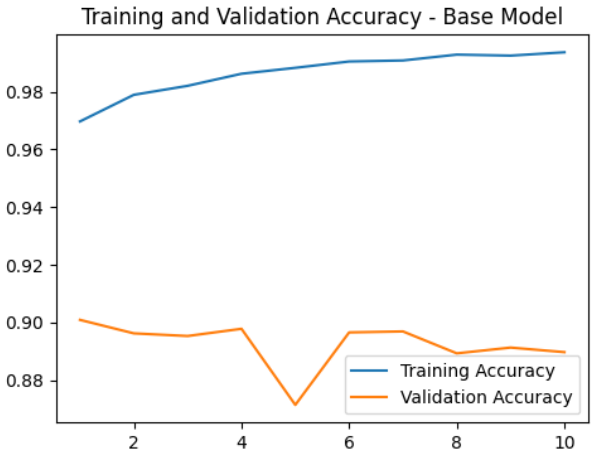
Model	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Testing Accuracy	Testing Loss
Base Model	99.36%	0.017	88.97%	0.733	89.20%	0.709
Dropout Model	98.79%	0.033	89.84%	0.661	90.11%	0.668
Dropout + Early Stopping Model	96.90%	0.089	90.29%	**0.356**	**91.76%**	**0.263**
Dropout + Early Stopping + Regularized Model	**97.42%**	**0.089**	**90.39%**	0.458	91.13%	0.285

- Dropout and early stopping significantly mitigated overfitting, evident from the closer gap between training and validation/testing metrics. The regularization slightly decreased the testing accuracy, posibbly due to regularization metric being higher than it has to be.
- The Dropout + Early Stopping model showcased the best performance in terms of generalizing to unseen data, achieving the highest testing accuracy and lowest testing loss. This improvement underscores the effectiveness of early stopping in preventing overlearning and enhancing the model's ability to generalize.
- Regularization Effects: The addition of regularization after early stopping and dropout did not significantly improve the model's ability to generalize based on testing accuracy and loss. While it helped to reduce overfitting (as indicated by reduced training accuracy), the slight decrease in testing performance suggests that the balance between bias and variance could be optimized further.

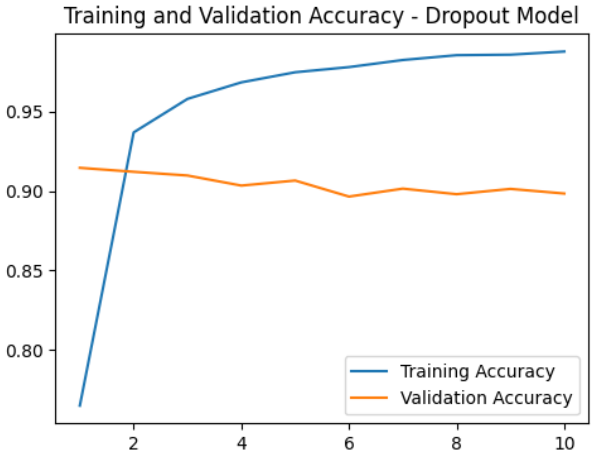
- **Plot the training and validation accuracy over time (epochs).**
- **Plot the training and validation loss over time (epochs).**

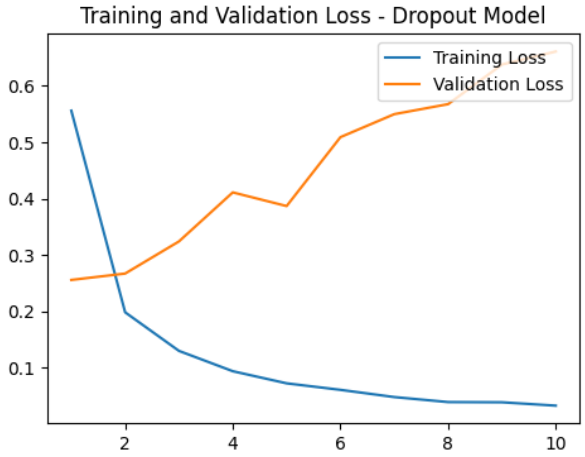
Base Model:

We can identify the overfitting from the graphs where there is huge gap between the validation and training metrics.

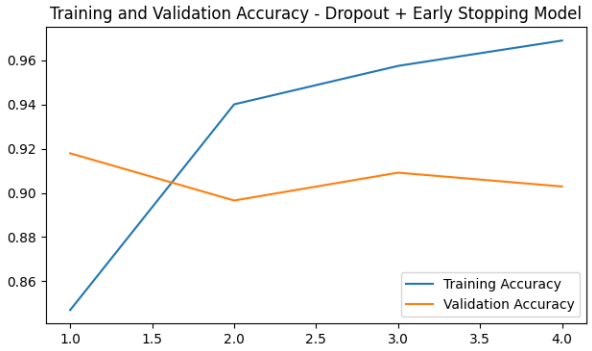


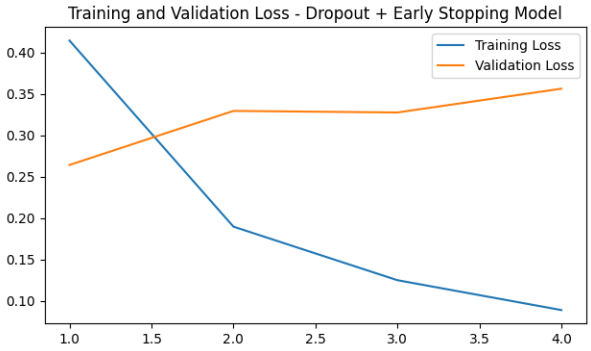
Dropout Model: training accuracy surpasses validation accuracy significantly, suggesting potential overfitting, while training loss decreases sharply compared to a more gradual reduction in validation loss, indicating good learning but with room for improved generalization.



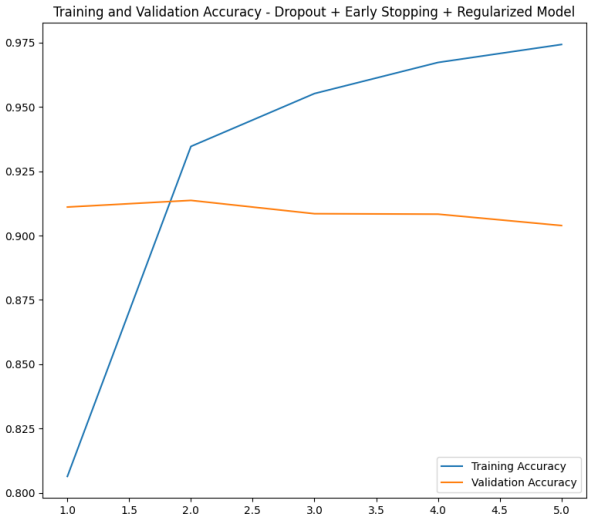


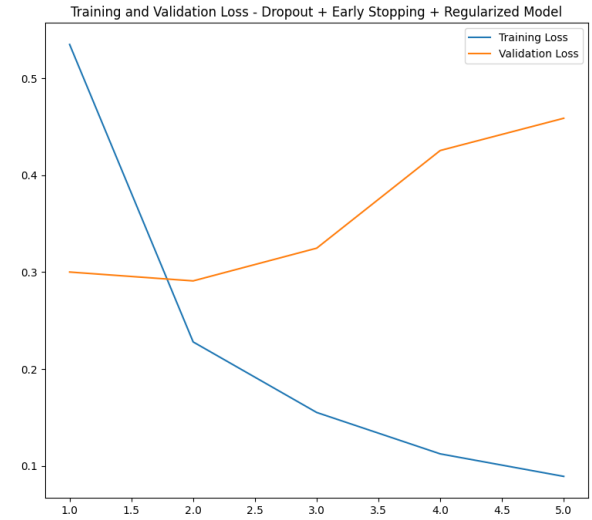
Dropout + Early Stopping Model: Performed well, among other models, ability to generalize on unseen data as well as trained samples.



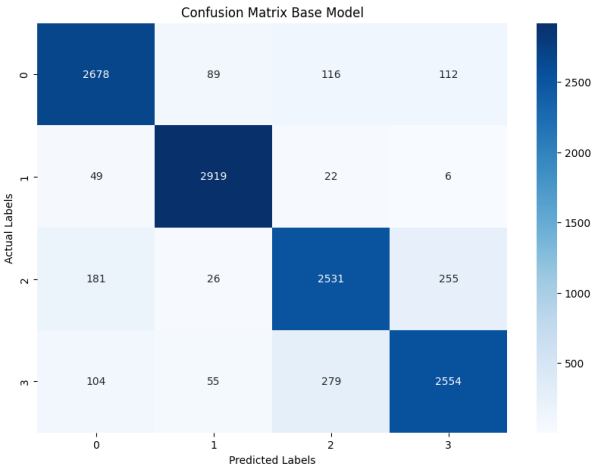


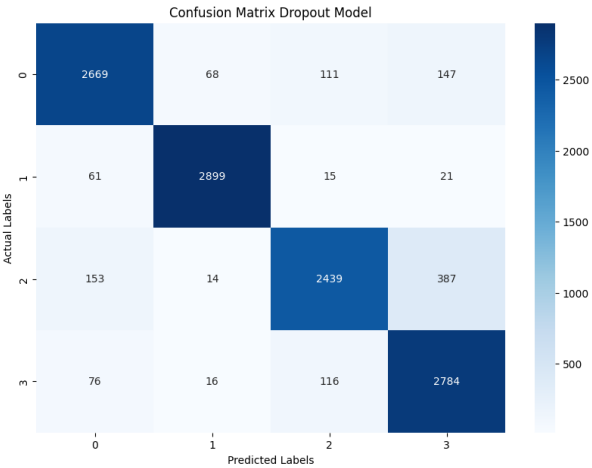
Dropout + Early Stopping + Regularized Model: Early stopping reached at 5 epochs, there is little difference within validation and training metrics.

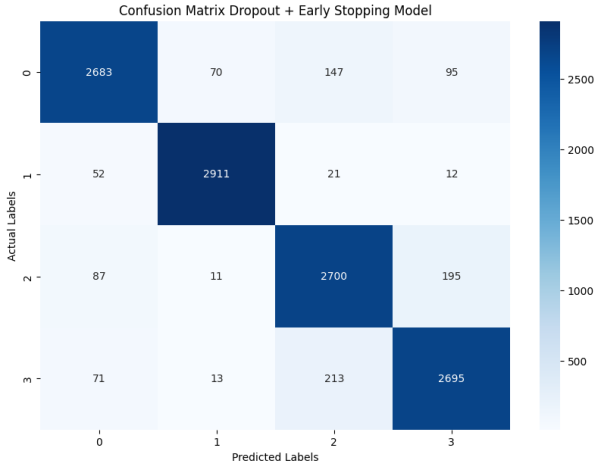


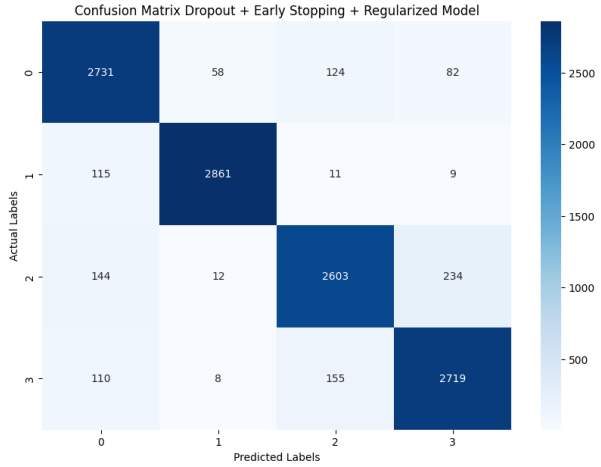


- **Generate a confusion matrix using the model's predictions on the test set.**









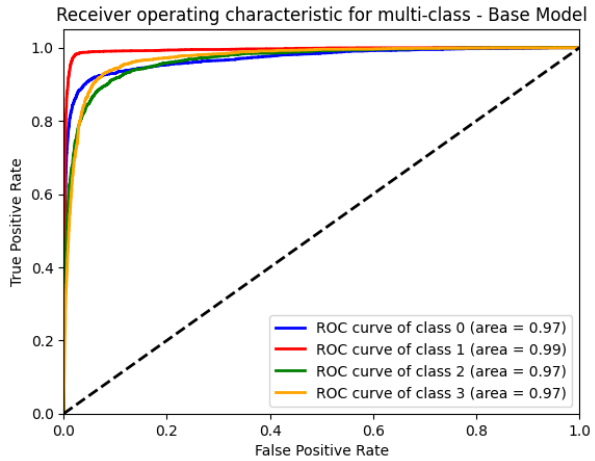
As you can see almost every prediction is in the diagonal indicating higher accuracy and predictions for all models.

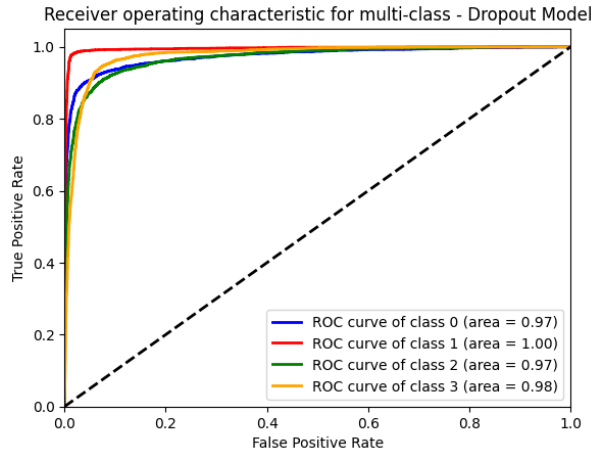
- **Provide the Precision, recall and F1 score.**

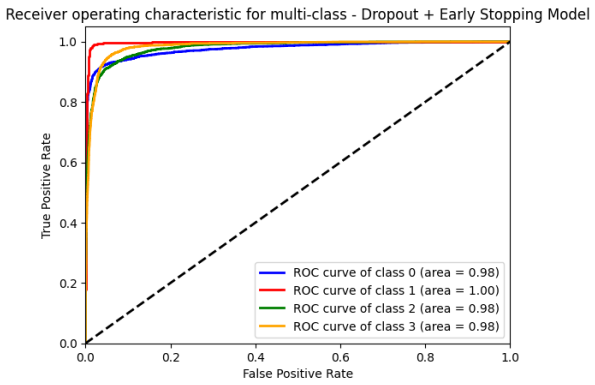
Model	Precision	Recall	F1 Score
Base Model	89.13%	89.20%	89.15%
Dropout Model	90.32%	90.11%	90.08%
Dropout + Early Stopping Model	**91.79%**	**91.76%**	**91.76%**
Dropout + Early Stopping + Regularized Model	91.19%	91.13%	91.14%

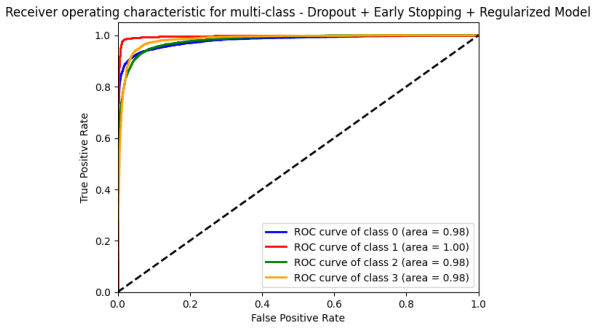
Precision, Recall, and F1 Score: These metrics improved with each iteration, with the Dropout + Early Stopping model again standing out. These improvements suggest not only better generalization but also a balanced ability to predict across different classes accurately.

- **Plot the ROC curve**









Model performance for multi-class classification in the all the models is great as seen from the AUC Area Under the Curve for each class is very high, close to 1, which indicates a strong ability to distinguish between the positive class and the negative class for each category. Class 1, with an AUC of 1.00, is perfectly distinguished by the model, while the other classes also show good discrimination with AUCs of 0.98. The curves hug the top left corner, reflecting high true positive rates and low false positive rates, which is indicative of a well-performing classifier.

Bonus points [max 10 points]

Vision Transformer (ViT) for Image Classification [7 points]
Implement a vision transformer paper. to solve a problem defined in Assignment 1, Part I. Compare the results with VGG and ResNet.
EfficientNet [3 points]

VGG Model:

- **Training Accuracy**: 93.44%
- **Validation Accuracy**: 92.78%
- **Test Accuracy**: 91.83%
- **Precision**: 90.63%
- **Recall**: 90.22%
- **F1 Score**: 90.14%

ResNet Model:

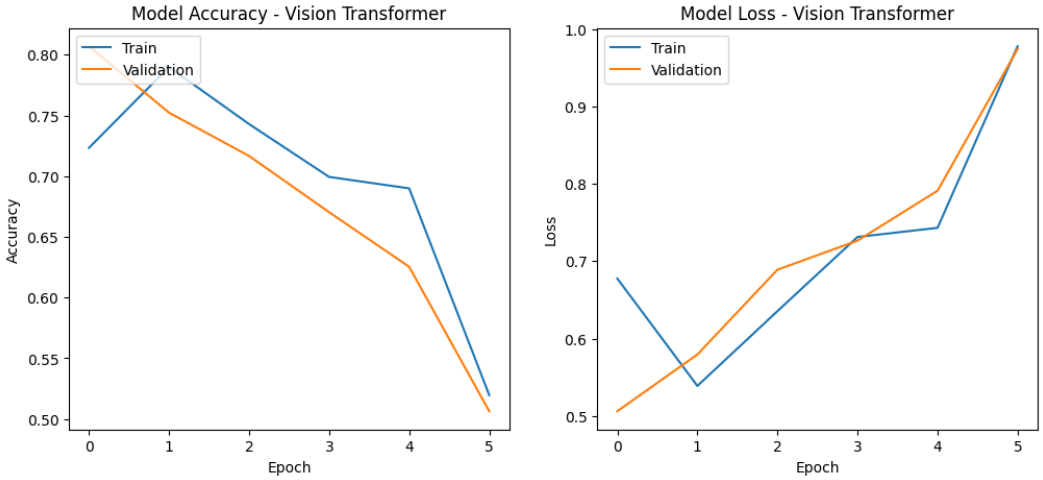
- **Training Accuracy**: 97.14%
- **Validation Accuracy**: 90.01%
- **Test Accuracy**: 90.14%
- **Precision**: 90.57%
- **Recall**: 90.15%

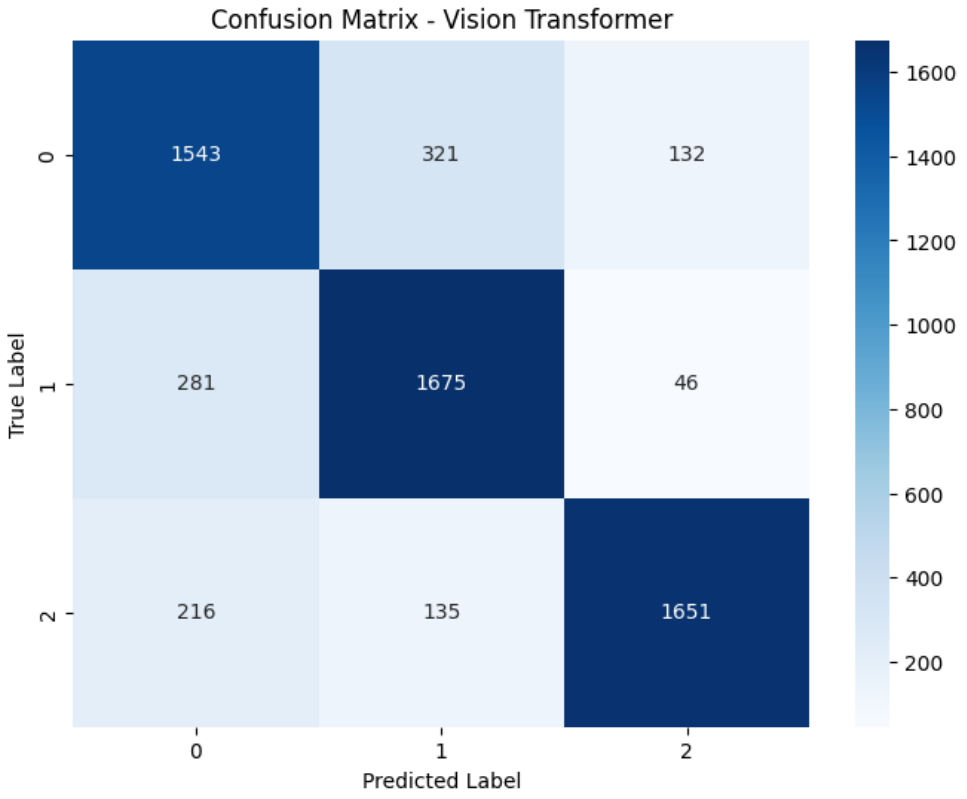
- **F1 Score**: 90.15%

Vision Transformer (ViT) Model (Based on images and provided values):

- **Training Accuracy**: 78%
- **Validation Accuracy**: 80% before it starts declining, indicating overfitting or a need for additional tuning.
- **Training Loss**: 0.67
- **Test Accuracy**: 81.15%
- **Precision**: 0.8151
- **Recall**: 0.8115
- **F1 Score**: 0.8124

- VGG had a well-balanced performance across training, validation, and test datasets. The precision, recall, and F1 scores are quite close to each other, suggesting that the VGG model generalizes well. Resnet showed a higher training accuracy, which indicated potential overfitting as this does not show high test accuracy. The ResNet model had very similar precision, recall, and F1 score to VGG.
- Vision Transformer (ViT) gave a lower test accuracy compared to the VGG and ResNet models. The precision, recall, and F1 score are also lower, suggesting that the ViT model's ability to generalize had not been as good as the other two models. However, this could also be due to a smaller number of epochs or other parameter settings and the need for further hyperparameter tuning.





Use EfficientNet to solve a problem defined in Assignment 1, Part I. Compare the results with VGG and ResNet. You can use pre-defined EfficientNet models.

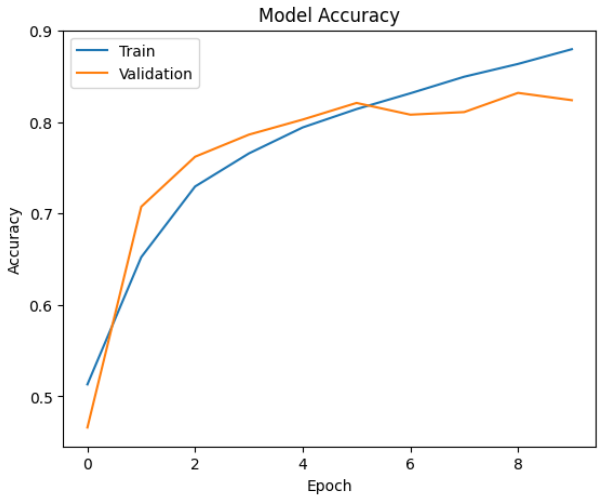
EfficientNet Model:

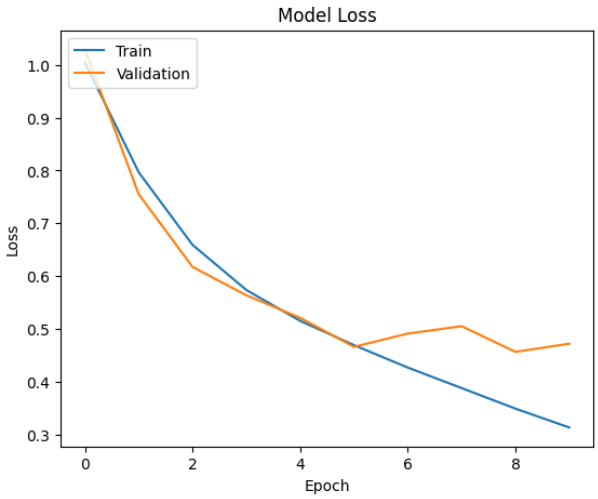
- **Training Accuracy**: Increases to approximately 87.9% by epoch 10.
- **Validation Accuracy**: Peaks at approximately 83% by epoch 10.
- **Test Accuracy**: 83.08%
- **Precision**: 0.8332
- **Recall**: 0.8308
- **F1 Score**: 0.8315

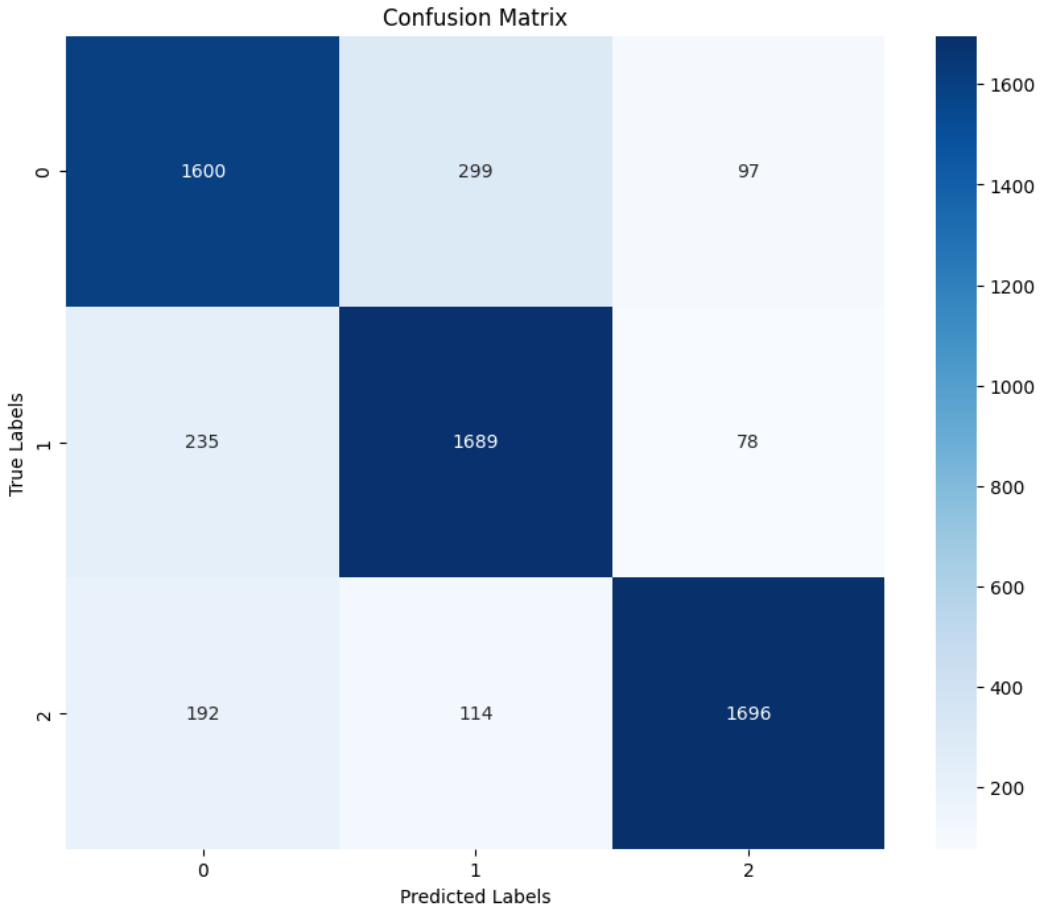
- VGG Model showed higher test accuracy (91.83%) and overall better precision, recall, and F1 scores compared to the EfficientNet model. Even the ResNet Model also displays superior performance in test accuracy (90.14%) and closely related precision, recall, and F1 scores when compared to EfficientNet.
- The VGG and ResNet models seem to be more robust in terms of generalization as reflected by higher validation and test accuracies compared to the EfficientNet model.
- The EfficientNet model, while showing a good learning trend, does not reach the performance level of the VGG or ResNet models by the 10th epoch based on the provided metrics.
- The confusion matrix for the EfficientNet model indicates some misclassification between the classes, but less confusion between class 2 and the others, similar to the VGG model.

- The accuracy and loss graphs for EfficientNet show healthy convergence, with the training and validation accuracy curves coming closer together and the validation loss curve flattening out, suggesting that longer training might yield improved results.

While the EfficientNet model has not outperformed the VGG and ResNet models based on the provided metrics, it is worth noting that EfficientNet architectures tend to perform better with more extensive training and especially when leveraging transfer learning with pre-trained weights. For the particular dataset and problem defined in the assignment, VGG and ResNet currently offer better accuracy and generalization capabilities. However, with additional epochs, tuning, or even potentially leveraging EfficientNet's pre-trained weights if applicable, wemight observe a significant improvement in the EfficientNet model's performance.







REFERENCES

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

https://www.timescale.com/blog/what-is-a-time-series-plot-and-how-can-you-create-one/

https://plotly.com/python-api-reference/generated/plotly.express.line_polar.html

https://machinelearningmastery.com/time-series-data-visualization-with-python/

https://towardsdatascience.com/5-types-of-plots-that-will-help-you-with-time-series-analysis-b63747818705

https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f

https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233#:~:text=Utilizing a Keras LSTM model to forecast stock trends&text=At the same time%2C these,LSTM)%20for%20times%20series%20forecasting.

https://medium.com/@mrconnor/predicting-stock-prices-using-lstm-neural-networks-d5877224ee0

https://towardsdatascience.com/time-series-of-price-anomaly-detection-with-lstm-11a12ba4f6d9

https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a