# Lab - 2

Akanksha Tyagi and Anjali H. Ojha

Department of Applied Data Science, San Jose State University

DATA 255: Deep Learning Applications

Dr. Simon Shim

May 6, 2024

## Part 1: Implement SR-GAN

Implement SR-GAN from paper Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (Ledig et al., 2017) using PyTorch.

The paper tackles a common issue in image super-resolution: maintaining fine texture details when enlarging images significantly. Past methods aimed at reducing mean squared error often produced images lacking fine details and visual quality at higher resolutions. The authors introduce SRGAN, a Generative Adversarial Network (GAN) model with a novel loss function that combines adversarial and content losses. The adversarial loss helps the model generate images more akin to real photos, while the content loss prioritizes perceptual likeness over pixel accuracy. SRGAN excels at creating realistic images with 4x upscaling, outperforming prior methods in terms of visual appeal based on extensive evaluations.

### Figure 1

*SRGAN Architecture From paper*

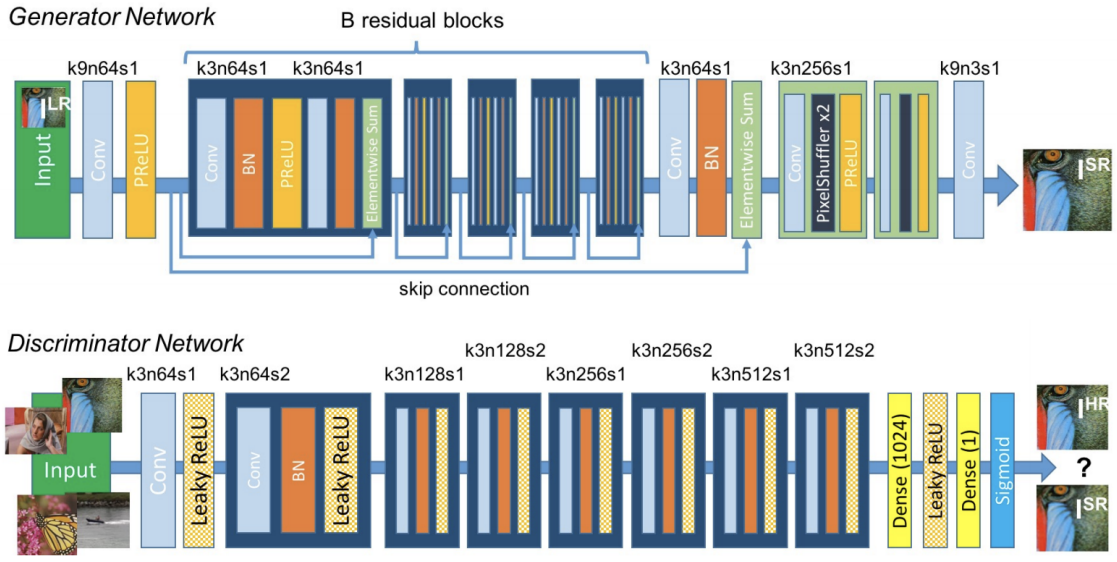


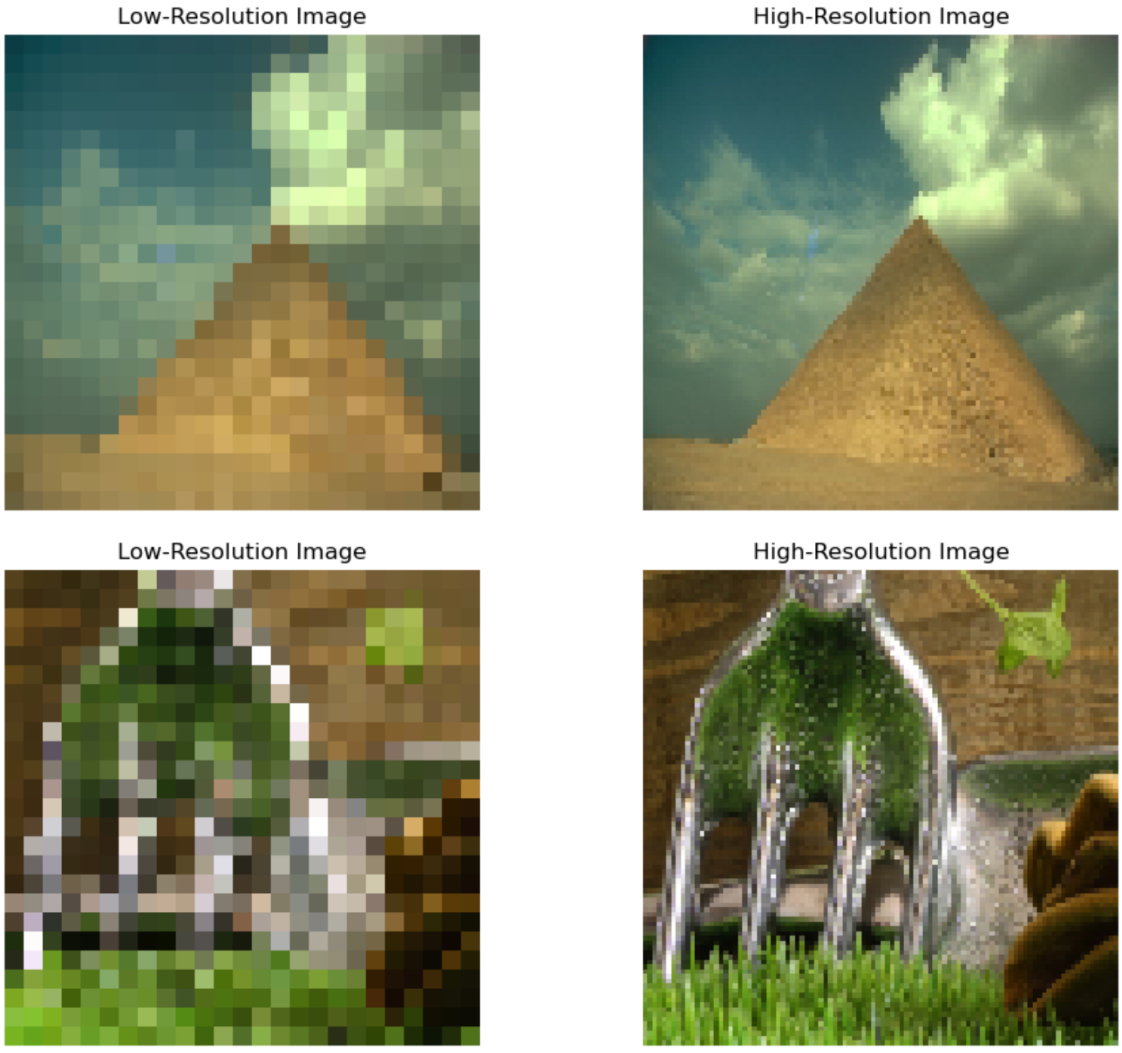

Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

***Training Data***

The provided dataset is low-resolution (LR) images with size (25 x 25 pixels )and high-resolution (HR) images with (100 x 100 pixels). There are 509 low-resolution and high-resolution images provided and there are 6 Test images with sizes of 512 x 512 pixels. Figure 2 shows the sample training data. The dataset provided in the pair is low-resolution on the left side and high-resolution on the right side.

**Figure 2**

*Sample training examples Low-Res image is 25*25 and High-Res image is 100*100 pixels*



### *Data Preprocessing*

The data transformation includes resizing, converting to tensors, and normalization. The dataset is populated by reading LR and HR image files from specified directories. The class provides methods to retrieve individual items and determine the dataset's length. Additionally, it includes print statements to display information about the dataset during initialization.

### *Models*

This paper explores the GAN for image enhancement, as the generator network can generate images but how to tune weights to make the generation efficient was a challenging task. The paper used perpetual loss to handle that. Here are the samples for the model summary of the Generator and Discriminator network. Figure 3 shows the first few layers of the generator network, and figure 4 shows the full discriminator network summary including different layers and trainable parameters.

**Figure 3**

*Generator Model Starting Some Layer*

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv2d-1 | [-1, 64, 24, 24] | 15,616 |
| Identity-2 | [-1, 64, 24, 24] | 0 |
| PReLU-3 | [-1, 64, 24, 24] | 64 |
| ConvBlock-4 | [-1, 64, 24, 24] | 0 |
| Conv2d-5 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-6 | [-1, 64, 24, 24] | 128 |
| PReLU-7 | [-1, 64, 24, 24] | 64 |
| ConvBlock-8 | [-1, 64, 24, 24] | 0 |
| Conv2d-9 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-10 | [-1, 64, 24, 24] | 128 |
| ConvBlock-11 | [-1, 64, 24, 24] | 0 |
| ResidualBlock-12 | [-1, 64, 24, 24] | 0 |
| Conv2d-13 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-14 | [-1, 64, 24, 24] | 128 |
| PReLU-15 | [-1, 64, 24, 24] | 64 |
| ConvBlock-16 | [-1, 64, 24, 24] | 0 |
| Conv2d-17 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-18 | [-1, 64, 24, 24] | 128 |
| ConvBlock-19 | [-1, 64, 24, 24] | 0 |
| ResidualBlock-20 | [-1, 64, 24, 24] | 0 |
| Conv2d-21 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-22 | [-1, 64, 24, 24] | 128 |
| PReLU-23 | [-1, 64, 24, 24] | 64 |
| ConvBlock-24 | [-1, 64, 24, 24] | 0 |
| Conv2d-25 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-26 | [-1, 64, 24, 24] | 128 |
| ConvBlock-27 | [-1, 64, 24, 24] | 0 |
| ResidualBlock-28 | [-1, 64, 24, 24] | 0 |
| Conv2d-29 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-30 | [-1, 64, 24, 24] | 128 |
| PReLU-31 | [-1, 64, 24, 24] | 64 |
| ConvBlock-32 | [-1, 64, 24, 24] | 0 |
| Conv2d-33 | [-1, 64, 24, 24] | 36,928 |
| BatchNorm2d-34 | [-1, 64, 24, 24] | 128 |

**Figure 4**

*Discriminator Model*

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 96, 96]           1,792
          Identity-2           [-1, 64, 96, 96]               0
         LeakyReLU-3           [-1, 64, 96, 96]               0
         ConvBlock-4           [-1, 64, 96, 96]               0
            Conv2d-5           [-1, 64, 48, 48]          36,928
       BatchNorm2d-6           [-1, 64, 48, 48]             128
         LeakyReLU-7           [-1, 64, 48, 48]               0
         ConvBlock-8           [-1, 64, 48, 48]               0
            Conv2d-9          [-1, 128, 48, 48]          73,856
      BatchNorm2d-10          [-1, 128, 48, 48]             256
        LeakyReLU-11          [-1, 128, 48, 48]               0
        ConvBlock-12          [-1, 128, 48, 48]               0
           Conv2d-13          [-1, 128, 24, 24]         147,584
      BatchNorm2d-14          [-1, 128, 24, 24]             256
        LeakyReLU-15          [-1, 128, 24, 24]               0
        ConvBlock-16          [-1, 128, 24, 24]               0
           Conv2d-17          [-1, 256, 24, 24]         295,168
      BatchNorm2d-18          [-1, 256, 24, 24]             512
        LeakyReLU-19          [-1, 256, 24, 24]               0
        ConvBlock-20          [-1, 256, 24, 24]               0
           Conv2d-21          [-1, 256, 12, 12]         590,080
      BatchNorm2d-22          [-1, 256, 12, 12]             512
        LeakyReLU-23          [-1, 256, 12, 12]               0
        ConvBlock-24          [-1, 256, 12, 12]               0
           Conv2d-25          [-1, 512, 12, 12]       1,180,160
      BatchNorm2d-26          [-1, 512, 12, 12]           1,024
        LeakyReLU-27          [-1, 512, 12, 12]               0
        ConvBlock-28          [-1, 512, 12, 12]               0
           Conv2d-29            [-1, 512, 6, 6]       2,359,808
      BatchNorm2d-30            [-1, 512, 6, 6]           1,024
        LeakyReLU-31            [-1, 512, 6, 6]               0
        ConvBlock-32            [-1, 512, 6, 6]               0
           Conv2d-33              [-1, 1, 6, 6]           4,609
================================================================
Total params: 4,693,697
Trainable params: 4,693,697
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.11
Forward/backward pass size (MB): 42.19
Params size (MB): 17.91
Estimated Total Size (MB): 60.20
----------------------------------------------------------------
```

### *Training Metrics*

With the given dataset, we trained the model 100 iterations and tracked different training metrics like Generator Loss, Discriminator Loss, TV Loss, and Perceptual Loss.

**Figure 5**

*Sample Generated Images and other metrics while training*

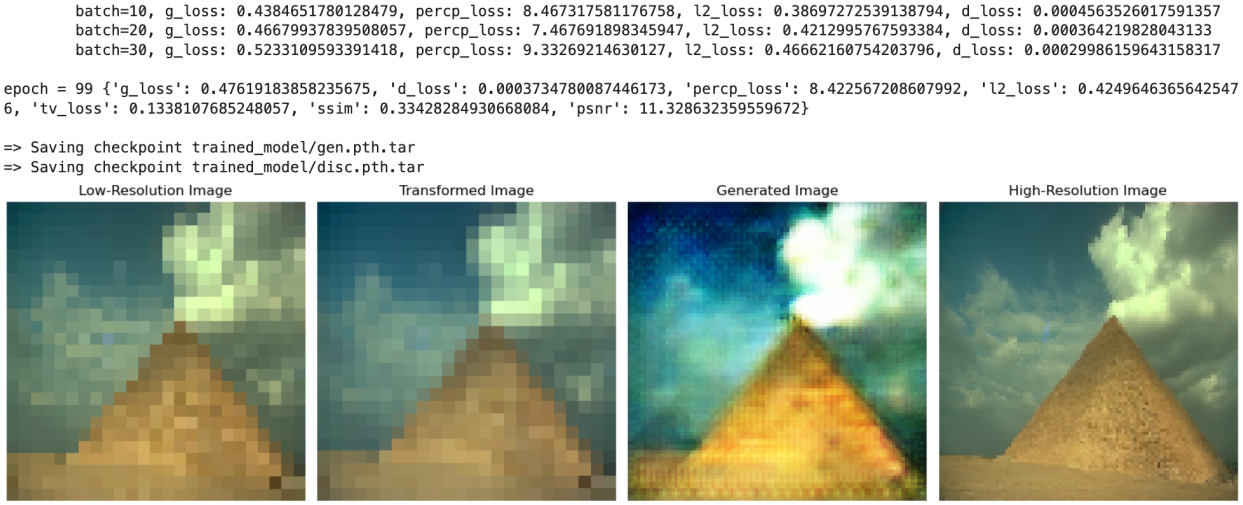
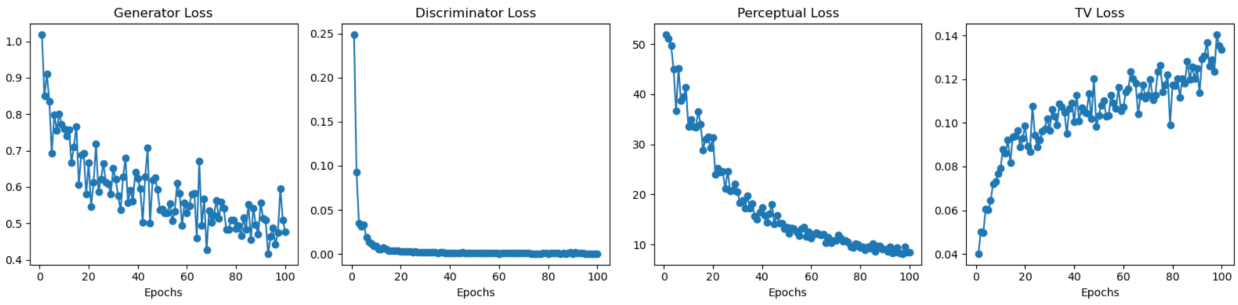



**Figure 6**

*Training Losses while training*

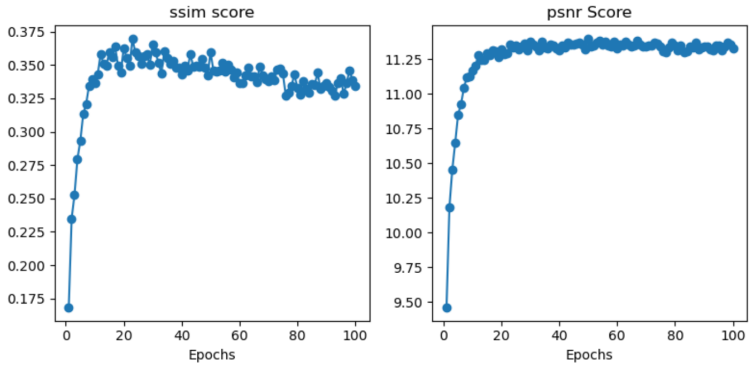



Other than that we are also computing the Structural Similarity Index (SSIM), and Peak Signal-to-Noise Ratio (PSNR) scores for the training data. SSIM measures the similarity between two images in terms of structure, luminance, and contrast, rather than

just pixel-wise differences. PSNR measures the ratio between the maximum possible power of a signal (in this case, the original image) and the power of the noise (error) that affects the quality of the reconstructed or compressed image. PSNR is the mean square error between the generated image and the actual image.

**Figure 7**

*SSIM and PSNR Scores on validation data for different training epochs.*

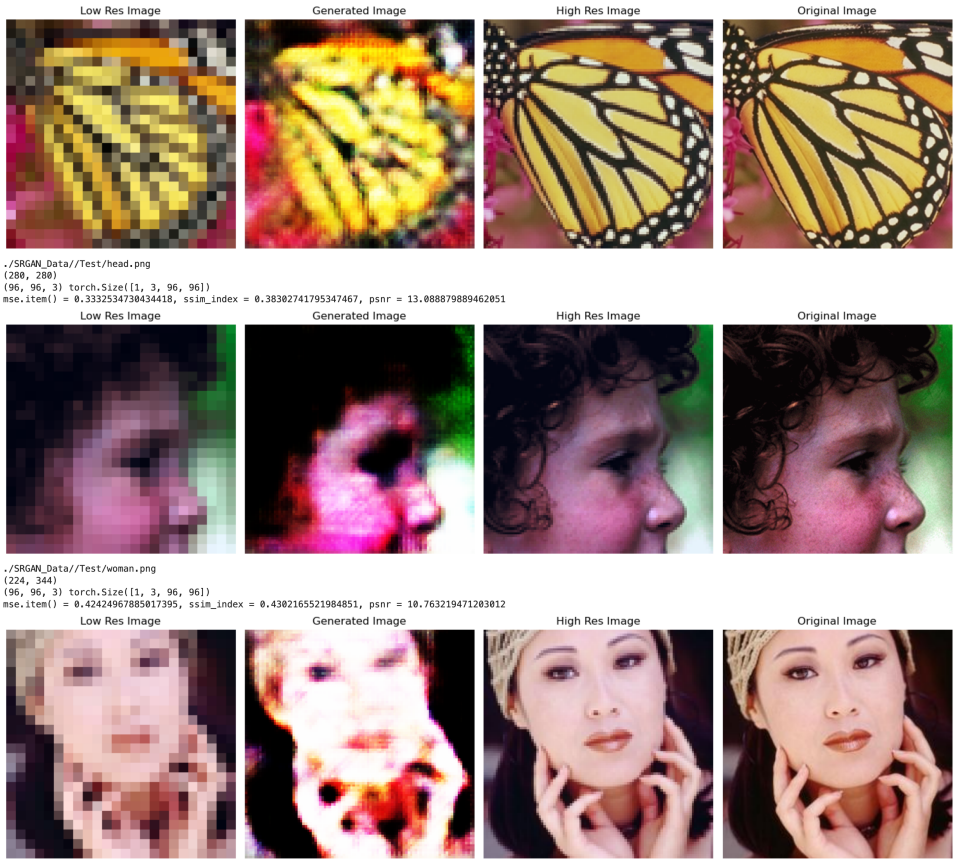



***Evaluation Metrics***

Figure 8 shows the generated images for the given test images. To get the output for the test images, we first down-sample the images to 24 x 24 pixels using the BICUBIC method and then generate the 96 x 96 pixel output images.

**Figure 8**

*Test Images Enhancement Comparison for Model Qualitative Analysis. Images on the left side are Low-Res images, High-Res generated images, actual High-Res images, and actual images.*




### *GAN Latent Space*

We explore the GAN latent space by first generating low-resolution versions of a high-resolution image using different downsampling methods such as bilinear, nearest, bicubic, etc. These low-resolution images are then processed through GAN to produce enhanced high-resolution images. The purpose is to visually compare the quality and characteristics of the enhanced images based on the downsampling method used. The code utilizes libraries like PyTorch and Torchvision for image manipulation and visualization, showing the impact of different interpolation techniques on the final image output.

**Figure 9**

*Different down-sampled Images on Left and Right side is the High-Res generated image (Part-1)*

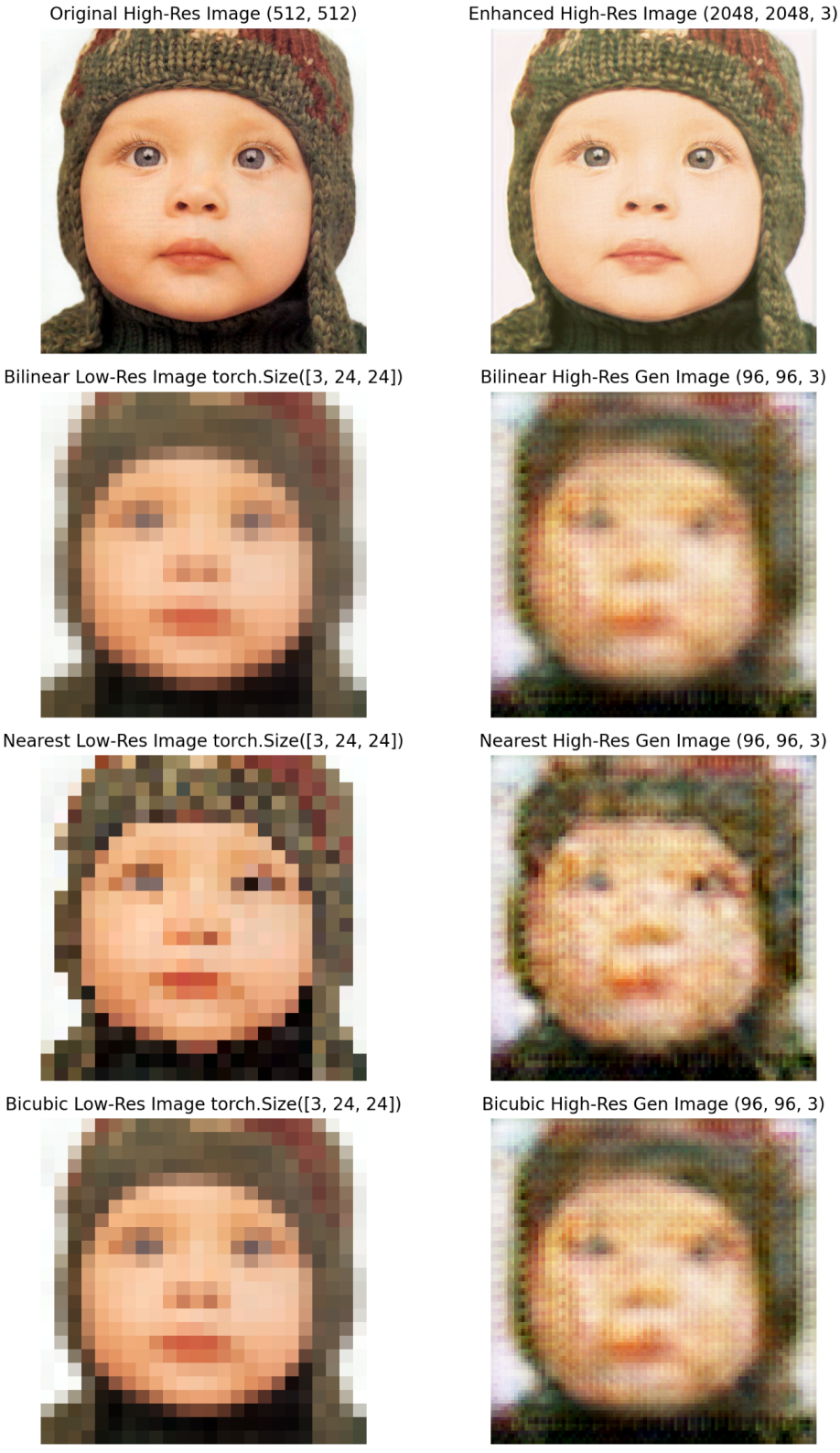
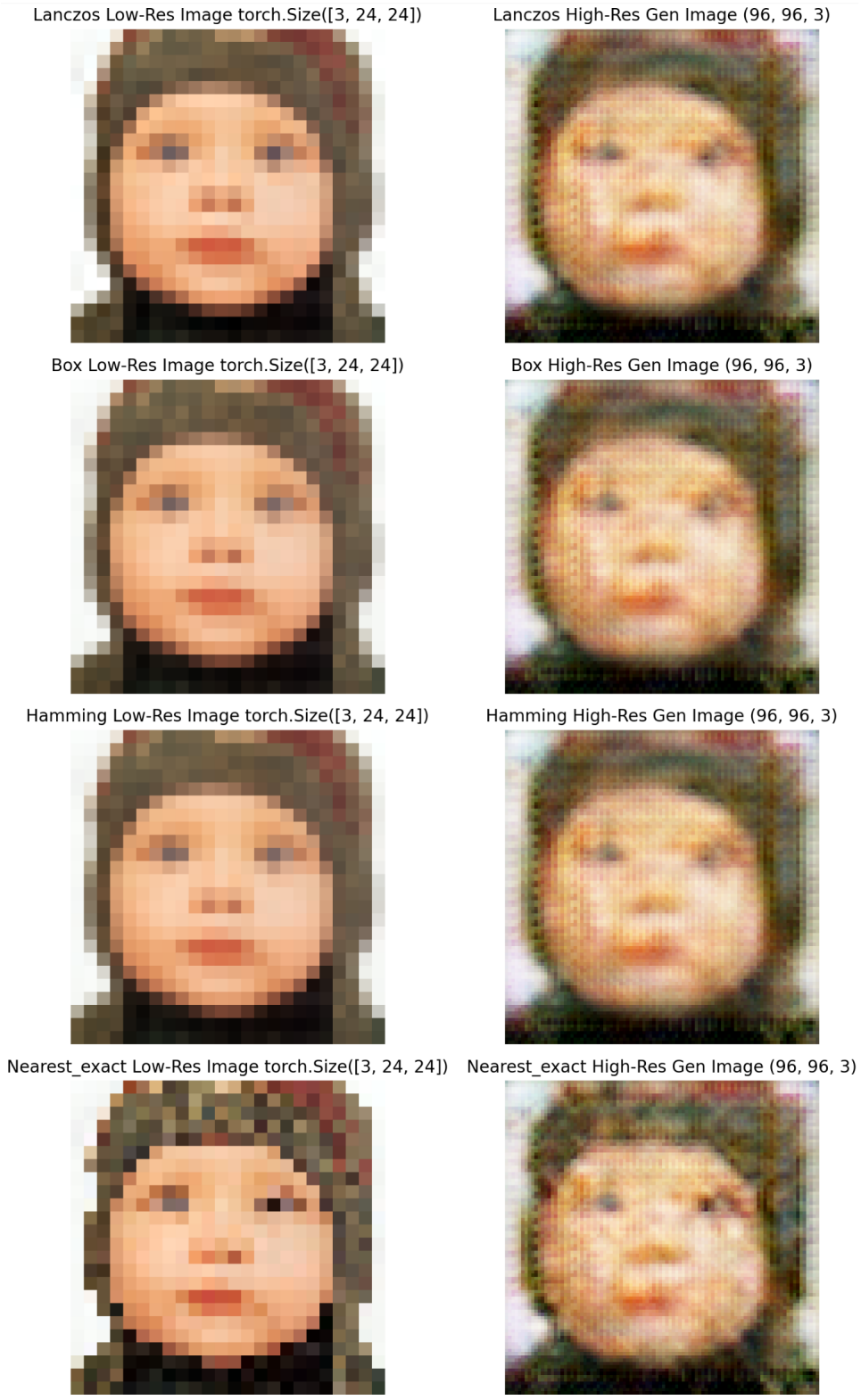


**Figure 10**

*Different down-sampled Images on Left and Right side is the High-Res generated image (Part-2)*



**Challenges and Observations**

There are the following points to observation -

- As the training data was very less, so there was not enough data for the model to learn. We can clearly say that the training converges but the image quality was not improving after a certain iteration.
- It is able to capture the basic structure right and keep the overall color tone to the actual image, but it overexposes the image colors making them more vibrant in each iteration.
- The model also was not able to learn the textures in the images but learned the bright and darker spots.
- The training part was very slow, as I was using my laptop for training it took a lot of time.

## Part 2: NLP - Jigsaw Toxic Comment Classification

### Step 1. Load the Dataset

The dataset comprises various categories of comments, each labeled based on its content. The summary reveals the frequency of occurrences for each label within the dataset. Among these categories, **Neutral** comments dominate the dataset with a substantial count of 143,346 occurrences, indicating a prevalent presence of non-toxic and non-offensive comments. Conversely, **Threat** comments are the least common, appearing only 478 times. The other categories, such as **Toxic**, **Severe Toxic**, **Obscene**, **Insult**, and **Identity Hate**, fall within moderate to high ranges of occurrence counts, suggesting a diverse range of comment types within the dataset, spanning from mildly offensive to extremely toxic or hateful content.

**Figure 11**

*Label Distribution for the Given Dataset*

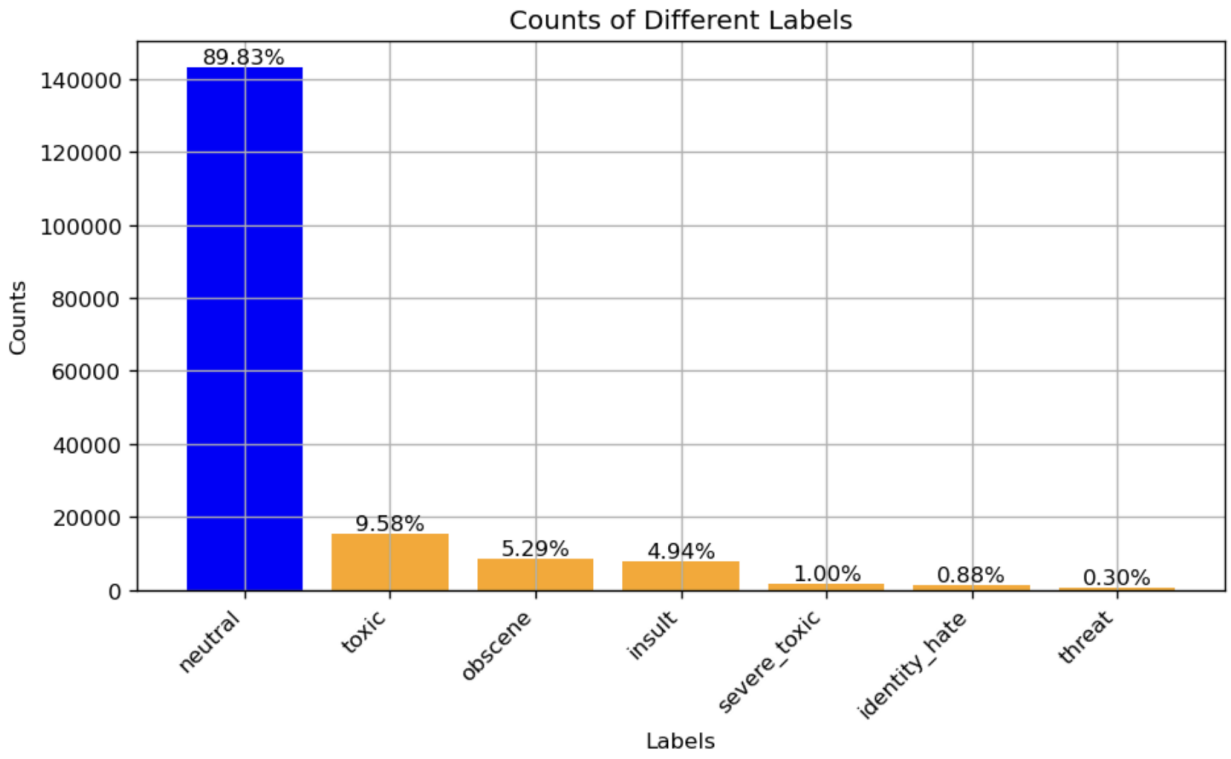


Table 1 shows the dataset description for the problem. We use the Toxic Comment Classification Challenge dataset from Kaggle (**jigsaw-toxic-comment-classification-challenge**).

**Table 1**

*Data Set Description*

| Data | Size | Valid Data |
|---|---|---|
| Training Data | 159,571 | 159,571 |
| Test Data | 153,164 | 63,978 |

### Step 2. Preprocess the Data

Preprocess the data according to your needs. This preprocessing typically involves resizing the images to the desired dimensions, applying transformations like normalization, and organizing the data into a format suitable for training. For this part of the process we use Spacy (Vasiliev, 2020), and NLTK (Loper & Bird, 2002) tools for some of the text processing tasks.

#### *Remove Contraction*

It is crucial to process contraction expansion in text data preprocessing. Contractions like "can't" or "won't" are expanded to their full forms ("cannot" and "will not"), ensuring linguistic consistency and standardization. The function initially replaces special characters with apostrophes to normalize the text. It then utilizes the `contractions` library to fix contractions where possible, ensuring that abbreviated forms are transformed into their complete equivalents. This method guarantees that contractions are appropriately expanded, contributing to a more uniform and understandable text corpus.

**Figure 12**

*Contraction Removal Example, here we can see that "can't" is replaced with "cannot" and "I'am" is replaced with "I am"*

| comment_text | wo_contractions |
|---|---|
| Hellllllllo! 😊 How's it going? I can't wait for tomorrow's party! 🎉 | Hellllllllo! 😊 How's it going? I cannot wait for tomorrow's party! 🎉 |
| I'm feeling awesome today! Can't believe it's already Friday! 🚀 | I am feeling awesome today! Can't believe it is already Friday! 🚀 |
| they'd gone forever and I hate this | they would gone forever and I hate this |
| I've got a lot of work to do this weekend. 😅 | I have got a lot of work to do this weekend. 😅 |
| Hey man, I can't not trying to edit war. | Hey man, I cannot not trying to edit war. |
| D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | D'aww! He matches this background colour I am seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) |

### *Clean Repeat Patterns*

This pre-processing focuses on text normalization by lowercasing the text and removing repeated patterns. It employs regular expressions to identify and replace repeated characters, enhancing text clarity and reducing redundancy. Furthermore, the function addresses specific text patterns like profanity and offensive language, replacing them with suitable alternatives for a more sanitized text representation. These preprocessing steps significantly contribute to improved text quality and analysis accuracy.

**Figure 13**

*Repeat Removal Example*

| comment_text | wo_contractions | wo_repeat |
|---|---|---|
| Hellllllllo! 😊 How's it going? I can't wait for tomorrow's party! 🎉 | Hellllllllo! 😊 How's it going? I cannot wait for tomorrow's party! 🎉 | helo! 😊 how's it going? i cannot wait for tomorrow's party! 🎉 |
| I'm feeling awesome today! Can't believe it's already Friday! 🚀 | I am feeling awesome today! Can't believe it is already Friday! 🚀 | i am feeling awesome today! can't believe it is already friday! 🚀 |
| they'd gone forever and I hate this | they would gone forever and I hate this | they would gone forever and i hate this |
| I've got a lot of work to do this weekend. 😅 | I have got a lot of work to do this weekend. 😅 | i have got a lot of work to do this weekend. 😅 |
| Hey man, I can't not trying to edit war. | Hey man, I cannot not trying to edit war. | hey man, i cannot not trying to edit war. |
| D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | D'aww! He matches this background colour I am seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | d'aww! he matches this background colour i am seemingly stuck with. thanks. (talk) :, january , (utc) |

### *Remove Special Characters and Emojis*

The method targets the removal of emojis and emoticons from text data. Emojis, while expressive, can introduce noise in text analysis and processing tasks. This function employs a regular expression pattern to identify and eliminate emojis across various Unicode ranges, ensuring that the text remains emoji-free. This streamlined text format enhances the accuracy and relevance of NLP tasks, facilitating more effective semantic analysis and machine learning model training.

**Figure 14**

*Special Character Removal Example*

| | comment_text | wo_contractions | wo_special |
|---|---|---|---|
| 0 | Hellllllllo! 😊 How's it going? I can't wait for tomorrow's party! 🎉 | Hellllllllo! 😊 How's it going? I cannot wait for tomorrow's party! 🎉 | Hellllllllo! How's it going? I cannot wait for tomorrow's party! |
| 1 | I'm feeling awesome today! Can't believe it's already Friday! 🚀 | I am feeling awesome today! Can't believe it is already Friday! 🚀 | I am feeling awesome today! Can't believe it is already Friday! |
| 2 | they'd gone forever and I hate this | they would gone forever and I hate this | they would gone forever and I hate this |
| 3 | I've got a lot of work to do this weekend. 😅 | I have got a lot of work to do this weekend. 😅 | I have got a lot of work to do this weekend. |
| 4 | Hey man, I can't not trying to edit war. | Hey man, I cannot not trying to edit war. | Hey man, I cannot not trying to edit war. |
| 5 | D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | D'aww! He matches this background colour I am seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | D'aww! He matches this background colour I am seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) |

### *Remove Stop Words*

The function targets the removal of stop words from text data. Stop words are common words that contribute minimal semantic value and are often excluded from text analysis. This function utilizes a predefined list of stop words, augmented with custom additions, to filter out stop words from the text. By

**Figure 15**

*Stop Words Removal Example*

| comment_text | wo_contractions | wo_repeat | wo_stopwords |
|---|---|---|---|
| Hellllllllo! 😊 How's it going? I can't wait for tomorrow's party! 🎉 | Hellllllllo! 😊 How's it going? I cannot wait for tomorrow's party! 🎉 | helo! 😊 how's it going? i cannot wait for tomorrow's party! 🎉 | helo! 😊 how's going? wait tomorrow's party! 🎉 |
| I'm feeling awesome today! Can't believe it's already Friday! 🚀 | I am feeling awesome today! Can't believe it is already Friday! 🚀 | i am feeling awesome today! can't believe it is already friday! 🚀 | feeling awesome today! can't believe friday! 🚀 |
| they'd gone forever and I hate this | they would gone forever and I hate this | they would gone forever and i hate this | gone forever hate |
| I've got a lot of work to do this weekend. 😅 | I have got a lot of work to do this weekend. 😅 | i have got a lot of work to do this weekend. 😅 | got lot weekend. 😅 |
| Hey man, I can't not trying to edit war. | Hey man, I cannot not trying to edit war. | hey man, i cannot not trying to edit war. | hey man, trying edit war. |
| D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | D'aww! He matches this background colour I am seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | d'aww! he matches this background colour i am seemingly stuck with. thanks. (talk) :, january , (utc) | d'aww! matches background colour seemingly stuck with. thanks. (talk) :, january , (utc) |

### *Lemmatization*

Lemmatization is the process of reducing words to their base or root form, known as the lemma, by removing inflections or variations. This is often done in natural language processing (NLP) to standardize words and reduce them to a common base form.

**Figure 16**

*Lemmatization Example*

| | comment_text | wo_contractions | wo_repeat | wo_stopwords | w_lemma |
|---|---|---|---|---|---|
| 0 | Hellllllllo! 😊 How's it going? I can't wait for tomorrow's party! 🎉 | Hellllllllo! 😊 How's it going? I cannot wait for tomorrow's party! 🎉 | helo how s it going i cannot wait for tomorrow s party | helo s going wait tomorrow s party | helo s go wait tomorrow s party |
| 1 | I'm feeling awesome today! Can't believe it's already Friday! 🚀 | I am feeling awesome today! Can't believe it is already Friday! 🚀 | i am feeling awesome today can t believe it is already friday | feeling awesome today t believe friday | feel awesome today t believe friday |
| 2 | they'd gone forever and I hate this | they would gone forever and I hate this | they would gone forever and i hate this | gone forever hate | go forever hate |
| 3 | I've got a lot of work to do this weekend. 😅 | I have got a lot of work to do this weekend. 😅 | i have got a lot of work to do this weekend | got lot weekend | get lot weekend |
| 4 | Hey man, I can't not trying to edit war. | Hey man, I cannot not trying to edit war. | hey man i cannot not trying to edit war | hey man trying edit war | hey man try edit war |
| 5 | D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | D'aww! He matches this background colour I am seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | d aww he matches this background colour i am seemingly stuck with thanks talk january utc | d aww matches background colour seemingly stuck thanks talk january utc | d aww match background colour seemingly stick thank talk january utc |

Finally, after all the preprocessing we get clean texts which we can build on. Here is the example of that -

**Figure 17**

*Cleaned Data*

| comment_text | cleaned |
|---|---|
| Hellllllllo! 😊 How's it going? I can't wait for tomorrow's party! 🎉 | helo s go wait tomorrow s party |
| I'm feeling awesome today! Can't believe it's already Friday! 🚀 | feel awesome today t believe friday |
| they'd gone forever and I hate this | go forever hate |
| I've got a lot of work to do this weekend. 😅 | get lot weekend |
| Hey man, I can't not trying to edit war. | hey man try edit war |
| D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC) | d aww match background colour seemingly stick thank talk january utc |

Once all data is cleaned we trained a tokenizer to tokenize it, later that tokenized data will be input for the model training.

**Step 3. Utilize a Model Implementing a Natural Language Processing Strategy**

The LSTMModel2 architecture comprises several key components designed for text classification tasks. In the first layer we created an embedding layer using the **wiki-news-300d-1M.vec** (Mikolov et al., 2017), which transforms input tokens into dense vectors of fixed size, aiding in capturing semantic information. Following this, an LSTM layer processes these embeddings, leveraging its ability to retain long-range dependencies and sequential patterns within the input data. The batch_first parameter being set to True indicates that the input data is organized with batch size as the first dimension, aligning with common practices in deep learning frameworks.

After the LSTM layer, the model includes an AdaptiveMaxPool1d layer for global max pooling across the time dimension, condensing the output sequence length to 1 while preserving essential features. Dropout layers are strategically placed throughout the architecture, introducing regularization by randomly dropping a fraction of input units during training to mitigate overfitting. Subsequent fully connected Dense layers with ReLU activation introduce non-linearity, enabling the model to learn complex relationships within the data. The final Dense layer with a Sigmoid activation function produces output probabilities for each class, making it suitable for binary or multi-label classification tasks

where independent class probabilities are desired.

**Figure 18**

*LSTM Model Architecture*

```
LSTMModel2(
  (embedding_layer): Embedding(149352, 300)
  (lstm): LSTM(300, 160, batch_first=True)
  (global_max_pooling): AdaptiveMaxPool1d(output_size=1)
  (dropout1): Dropout(p=0.1, inplace=False)
  (dense1): Linear(in_features=160, out_features=80, bias=True)
  (relu): ReLU()
  (dropout2): Dropout(p=0.1, inplace=False)
  (dense2): Linear(in_features=80, out_features=40, bias=True)
  (dropout3): Dropout(p=0.1, inplace=False)
  (dense3): Linear(in_features=40, out_features=6, bias=True)
  (sigmoid): Sigmoid()
)
```

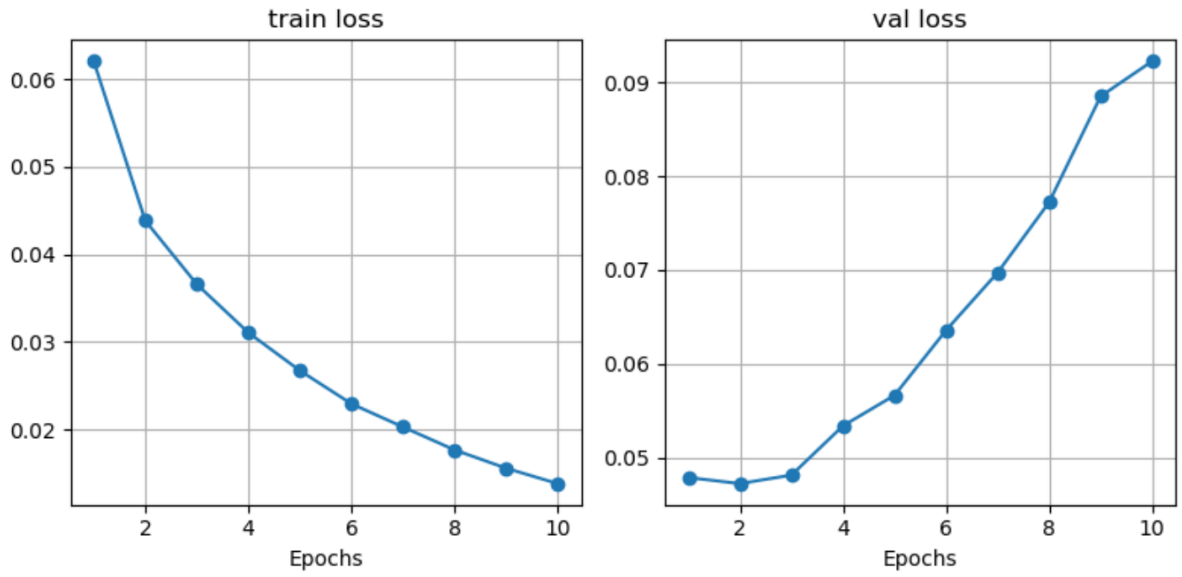
As it is a natural language processing problem, we used the LSTM model for this problem. In text data, each word and its ordering matters it contains contextual information so we can not just focus on a particular word itself.

**Step 4. Train Your Model**

We trained our model for 10 iterations and from the plots in the figure 19 we can see that the training loss is decreasing for each iteration but the validation data loss keeps increasing. And for 2 epochs training we are getting the lowest validation loss.

**Figure 19**

*Training Loss for the LSTM Model*




**Step 5. Display Results on the Test Dataset**

We evaluate the model's performance on the test data which have 63978 good examples. As it is a multilabel classification problem, we compute different metrics to evaluate the performance. We compute Precision, Recall, F1-Score, Accuracy, and ROC-AUC on the test data.

**Table 2**

*Classification Metrics*

| Class | Precision | Recall | F1-score | Accuracy | ROC AUC |
|---|---|---|---|---|---|
| toxic | 0.505 | 0.818 | 0.625 | 0.906 | 0.947 |
| severe_toxic | 0.306 | 0.460 | 0.367 | 0.991 | 0.981 |
| obscene | 0.607 | 0.721 | 0.659 | 0.957 | 0.969 |
| threat | 0.342 | 0.327 | 0.334 | 0.996 | 0.971 |
| insult | 0.590 | 0.591 | 0.591 | 0.956 | 0.959 |
| identity_hate | 0.572 | 0.433 | 0.493 | 0.990 | 0.968 |

**Figure 20**

*ROC AUC and Precision-Recall Curve on the Test Data*

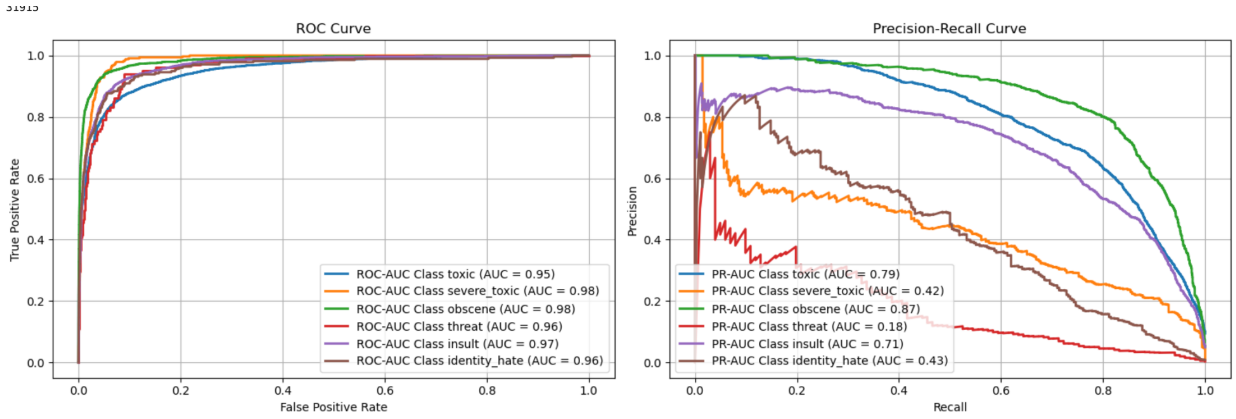



Figure 21 shows the cleaned text by the data pre-processing and the actual and predicted labels.

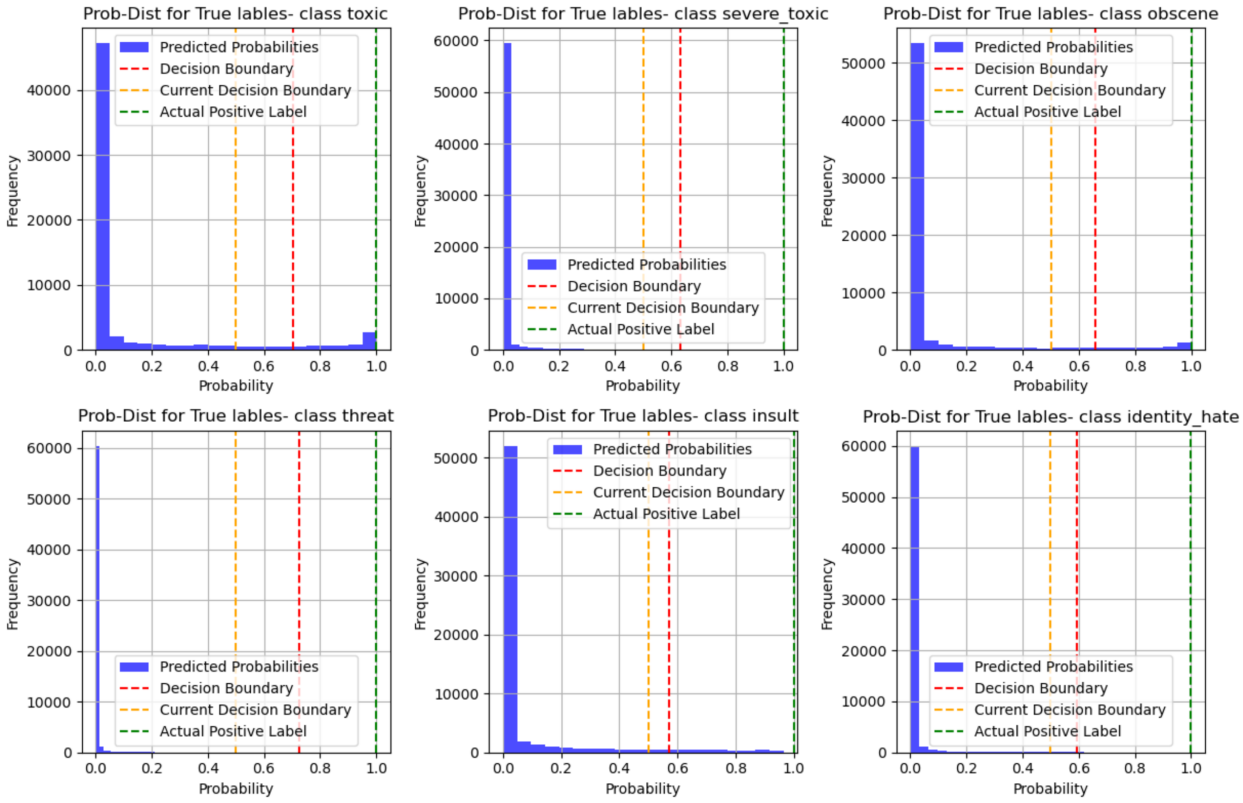
**Figure 21**

*Sample Test Data And Predicted Classes*

| comment_text | cleaned_text | actual_labels | predicted_label |
|---|---|---|---|
| DJ Robinson is gay as hell! he sucks his dick so much!!!!! | dj robinson gay hell suck dick | toxic,obscene,insult,identity_hate | toxic,obscene,insult |
| :Fuck off, you anti-semitic cunt. \| | fuck anti semitic cunt | toxic,obscene,insult | toxic,obscene,insult |
| ::No, he is an arrogant, self serving, immature idiot. Get it right. | arrogant self serve immature idiot right | toxic,obscene,insult | toxic,insult |
| ::::Well, it sucks to have a university to be nicknameless. And it's the first time in NCAA history that it has happened. / | suck university nicknameless ncaa history happen | toxic,obscene | toxic,obscene |
| == Argh == \n\n Some random idiot deleted the whole Japan article <. | argh random idiot delete japan article | toxic,obscene,insult | toxic,obscene,insult |

**Figure 22**

*Sample Test Data And Predicted Classes*




The best accuracy we got on the test data is **97.528%**, to achieve this we trained the model on the full dataset including all 159K instances, and evaluated its performance

on the test data. To further improve the model's performance we match the predicted class probabilities distribution to the trained data label distribution and then decide the threshold. Figure 22 shows how we decided the boundary.

**Challenges**

We faced the following challenges:

- As I was training an LSTM model it was very time-consuming and took some time to train.
- Handling the uncleaned data was also very challenging, as I tried various approaches but each step took a lot of back and forth.
- Experimented with two LSTM networks one simpler and the other one was a bit more deeper and wider, but did not observe any noticeable difference.
- The dataset was very highly skewed, and its a multilabel classification problem. As the baseline was more than 80% without doing anything just tag everything without any tag. Improving upon this was challenging, and we tried various learning rates for the optimizer and different regularization parameters.

## Part-3 PDF RAG LLM with Langchain

***Create a Retrieval-Augmented Generation (RAG) LLM.*** The first step is to create a Retrieval-Augmented Generation (RAG) Language Model (LLM) capable of consuming PDF documents. This model should allow users to input questions based on PDF documents and generate relevant answers. Your team can leverage any LLM model available, such as ChatGPT, Llama2, and Mistral, along with PDF data collected from various sources.

***Use PDF Dataset.*** Utilize the PDF dataset provided in the canvas or collect PDF data from relevant sources for training and testing the RAG model.

***Implement RAG Model with Langchain and FAISS.*** Implement the RAG model using Langchain (llama index) for document indexing and FAISS (or alternative databases like ChromaDB) for efficient similarity search. This implementation is crucial and carries 15 points. Thoroughly test the RAG model, establish quality metrics specific to RAG, fine-tune the model based on these metrics, and document the metric results in your report.

***Class Competition and Grading Criteria.*** This project will be conducted as a class competition, and grading will be based on the ROGUE score. Ensure that your RAG model achieves a high ROGUE score, indicating accurate and relevant answers generated in response to user prompts.

### Solution Approach

To develop the Retrieval-Augmented Generation (RAG) application, we utilize LangChain, the LLAMA 3 (AI@Meta, 2024) language model, and the FAISS (Douze et al., 2024; Johnson et al., 2019) vector database. The application leverages LangChain's integration framework, LLAMA 3's state-of-the-art language processing abilities, and FAISS's efficient vector search to enhance the application's ability to provide precise, contextually relevant information retrieval.

### *LLAMA 3*

The application uses LLAMA 3 as the LLM model to power the chatbot. LLAMA 3 was recently introduced by Meta. It uses the vocabulary of 128k tokens which substantially improves the model performance. We experiment with both the base LLAMA 3 model that has 8 billion parameters and the large model that has 70 billion parameters. We also use the LLAMA 3 model to generate embeddings. We use the Ollama library to create the LLAMA 3 models.

### *FAISS*

Facebook AI Similarity Search (Faiss) is a library designed for the efficient search and clustering of dense vectors. It supports algorithms capable of searching through any size of vector sets. We use FAISS to create a vector store for the RAG application using Langchain. The embedding model backed by LLAMA 3 is used to extract text embeddings from the PDF. These embeddings are inserted into a vector database using FAISS. The vector database allows for efficient search in the PDF which provides high-quality context for the questions in the RAG application.

### *Evaluation*

We use ROGUE (Lin, 2005) score to evaluate the RAG application. The ROUGE score assesses how effectively the generated answers match the reference answers. It measures how many key points and significant words from the reference text are included in the generated text. To evaluate the RAG application, the reference answers are generated by using ChatGPT 4. We provide the answers by ChatGPT 4 and those by our application in the accompanying jupyter notebook (`problem3.ipynb`). Table 3 shows the evaluation metrics. To start with, we use the following prompt in the model.

```
template = """
Answer the question based on the context below. If you can't
answer the question, reply "I don't know".

Context: {context}

Question: {question}
"""
```

| Metric | Score |
|---|---|
| rouge1 | 0.4464 |
| rouge2 | 0.2593 |
| rougeL | 0.3422 |
| rougeLsum | 0.3828 |

**Table 3**

*ROGUE score achieved by the model evaluated against ChatGPT 4.*

**Fine tuning.** We finetune the prompt to improve the ROGUE score. We observe that ChatGPT answers are quite detailed. Thus, we change the prompt by asking the model to reply in detail. We also observe that the ChatGPT answers all parts of the queries. So, we explicitly ask the model to ensure that all the parts of the question are answered. The final prompt is shown below.

```
template = """
Answer the question based on the context below.
Provide a detailed, clear, and informative response.
Use simple language for clarity and ensure all parts of the query are addressed.
```

```
Context: {context}


Question: {question}
"""
```

The evaluation metrics improve as shown in table 4.

| Metric | Score |
|---|---|
| rouge1 | 0.5012 |
| rouge2 | 0.2952 |
| rougeL | 0.3665 |
| rougeLsum | 0.4335 |

**Table 4**

*ROGUE score achieved by the model evaluated against ChatGPT 4 after finetuning the prompt.*

### *Challenges faced*

During the development of the app, we faced the following challenges:

- The LLAMA 3 model takes a long to run since it is a very big model with 70 billion parameters.
- The quality of the answer depends on the quality of the embedding model since it provides the context to the LLM model.
- Since we had only one pdf, the model couldn't be fine tuned. To fine tune the model, we need a lot of data.

### *RAG Application*

We develop a streamlit application for the pdf based question answering. The app uses llama 3 8B model so that the answer generation is quicker. Figure 23 shows the

landing page of the PDF based QA application. The user can upload a pdf and ask questions related to it. Once the user uploads the pdf, a vector embeddings store is created using the embeddings model as show in figure 24. After processing the pdf, the application becomes ready to answer questions. Figure 25 shows an example of a question answered by the chat bot. The application can be run using the following command :

```
streamlit run rag_app.py
```

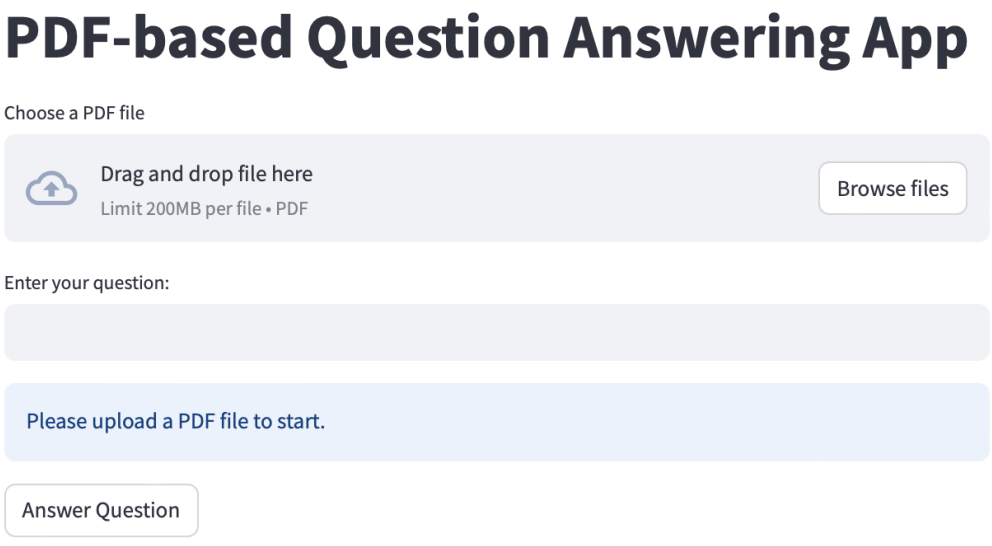



**Figure 23**

*Starting page of the RAG application.*

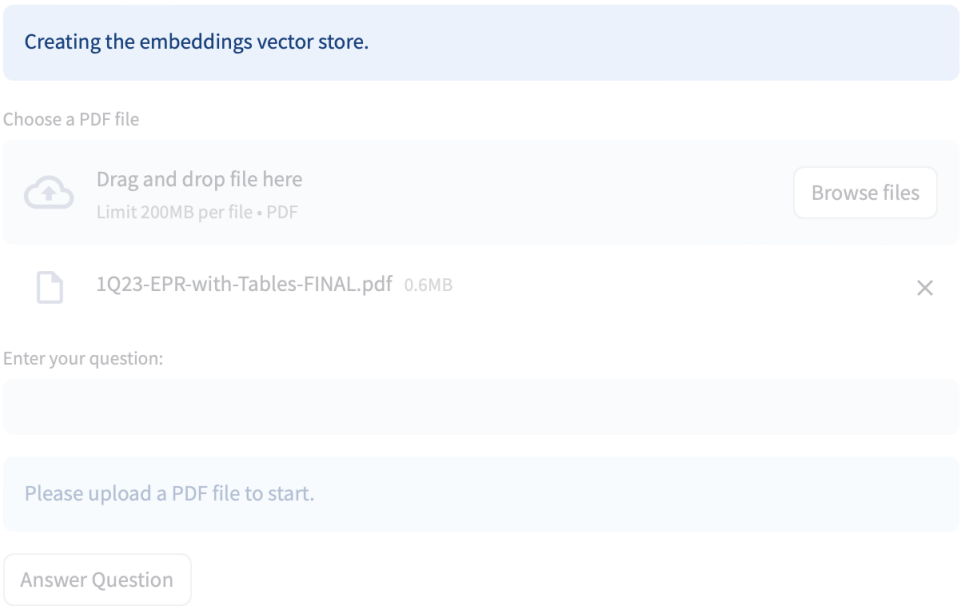



**Figure 24**

*PDF uploading page of the app. When the pdf is uploaded, the vector embeddings store is created using the embeddings model.*

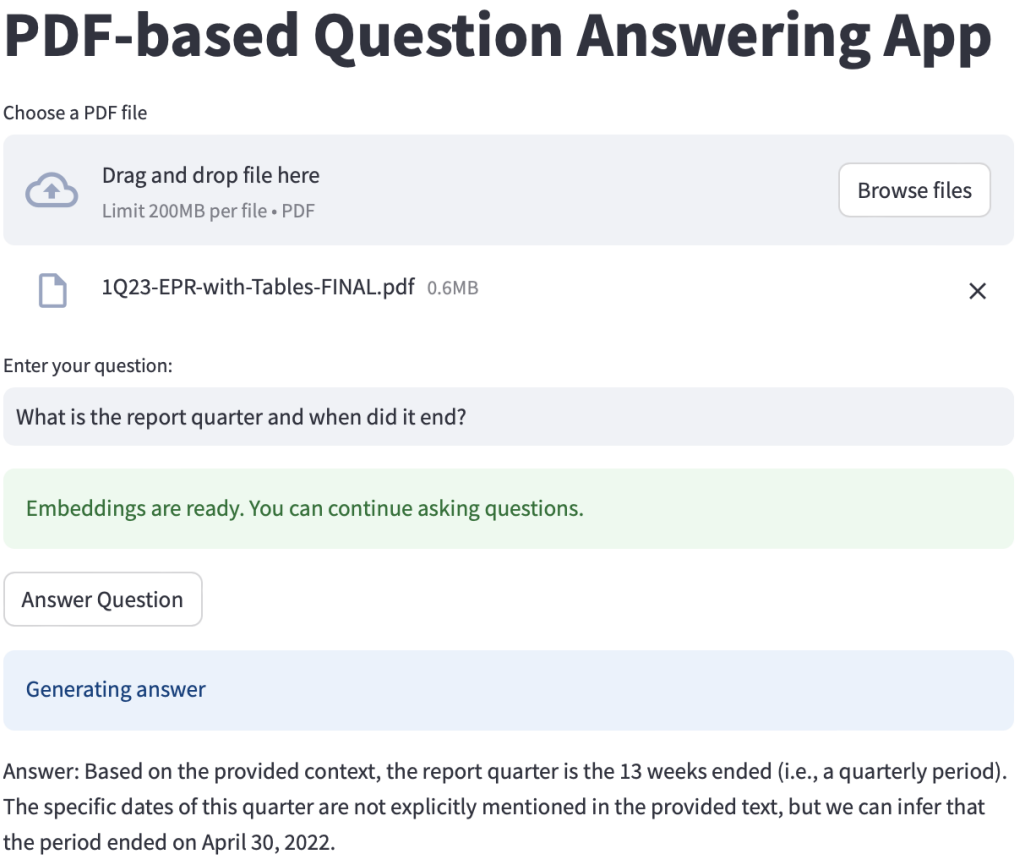



**Figure 25**

*Answer page of the app. The LLM is asked a question about the pdf. It reads the context and replies.*

## References

AI@Meta. (2024). Llama 3 model card. https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., & Jégou, H. (2024). The faiss library.

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, *7*(3), 535–547.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4681–4690.

Lin, C. (2005). Recall-oriented understudy for gisting evaluation (rouge). *Retrieved August*, *20*, 2005.

Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*.

Vasiliev, Y. (2020). *Natural language processing with python and spacy: A practical introduction*. No Starch Press.