



Generative Image Rectifier

Submitted by

ANURAG BOSE

Registration No. : 144-1111-0347-20 & Roll No: 203144-21-0036

SAGAR MONDAL

Registration No. : 144-1111-0335-20 & Roll No: 203144-21-0030

SHOUVIK BISWAS

Registration No. : 144-1112-0336-20 & Roll No: 203144-21-0036

Supervisor:

DR. Saptarsi Goswami

A Final Project Report submitted to the Department of Computer Science, Bangabasi Morning College, University of Calcutta in partial

Fulfillment of the requirement for the Degree of

B.Sc in COMPUTER SCIENCE

2022-2023 Semester VI

CERTIFICATE

This is to certify that Anurag Bose, Sagar Mondal, Shouvik Biswas have carried out their investigation entitled "***GIR (GENERATIVE IMAGE RECTIFIER)***" under my supervision as per the requirement for the degree of Bachelor of Science in the department of Computer Science, University of Calcutta. During this investigation they have learn Python for AI& ML different AI architecture and applied them independently. This report is not being submitted elsewhere for examination by other students except them.

.....
(Dr. Saptarsi Goswami)

Dept. of Computer Science
Bangabasi Morning College
University of Calcutta

.....
External Examiner

ACKNOWLEDGEMENT

With great pleasure we acknowledge that our respected teacher DR. Saptarsi Goswami, Computer Science Department (C.U) has introduced us to this fascinating field of study and guided us for the same. We achieved a golden opportunity to work on such a wonderful project under his guidance.

We are especially thankful and infact indebted to our guide DR. Saptarsi Goswami for his wonderful cooperation in completing our project successfully. He had been a wonderful guide to us throughout our project work. We express our sincere gratitude to him. We would like to express our special thanks of gratitude to our teacher who gave us the golden opportunity to do this wonderful project on the topic (**Generative Image Rectifier**), which also helped us in doing a lot of Research and we came to know about so many new things. We would be ever grateful to our respected teacher DR. Saptarsi Goswami for his valuable support.

.....

Anurag Bose

Date:-

Department of Computer Science
Bangabasi Morning College
University of Calcutta

.....

Sagar Mondal

.....

Shouvik Biswas

CONTENT

Topic	Page No.
Introduction and Project Overview	6-10
Methodology and Approach	11-16
Architecture Details	17-21
Model Training	22-24
Results and Evaluation	25
Challenges and Limitations	26-27
Future Work	28-29
Conclusion	30
Reference	31

Introducing

GIR

(Generative Image Rectifier)

Abstract

This project is dedicated to detect and rectify the blur present in the image. Generally blur occurs either due to the movement of the objects we are trying to capture or due to the shutter speed of the camera. Formally we can define blur as the smoothing of image pixels resulting in obscure image.

The model we are building uses three step process –

Blur detection, in which we are primarily identifying those portions of an image which are having blur. In order to make the model more robust, we are preventing the flat regions to be misclassified as blur.

After the detection of the blurs we are classifying it as either motion blur or general blur.

Our final step is to remove the blur. The model is based on a culmination of different techniques which includes CNN and fourier transformations to deblur the image.

Introduction and Project Overview:

Problem Description and The Task:

The problem statement addressed in this project is image rectification and enhancement using a deep learning model. The task involves developing a model capable of automatically removing blur and improving image quality for degraded images.

Problem Statement:

Images captured under various conditions, such as motion blur, out-of-focus blur, and sensor blur, can suffer from degradation, resulting in reduced visual quality and loss of important details. The problem is to rectify these degraded images and enhance their quality using a deep learning-based approach.

Task:

The primary task of the project is to build a powerful image rectification and enhancement model using deep learning techniques. The model will be trained to address the following specific challenges:

- 1. Blur Rectification:** The model should be able to identify and rectify different types of blur, including motion blur and out-of-focus blur, to restore sharpness and clarity in the images.
- 2. Blur Reduction:** The model should effectively reduce various types of blur, such as random blur and sensor blur, to improve the overall visual quality and reduce image artifacts.
- 3. Feature Preservation:** While removing blur, the model must preserve important image features, such as edges, textures, and fine structures. Preserving features ensures that crucial information in the image remains intact after enhancement.
- 4. Generalization:** The model should be capable of handling images captured in different conditions and environments. It should generalize well to unseen data and be robust to variations in image content, resolution, and degradation levels.
- 5. Efficiency and Real-Time Inference:** The model should be efficient enough to perform real-time or near real-time inference, making it practical for applications where quick image rectification is required.

Project Overview: Image Rectification and Enhancement:

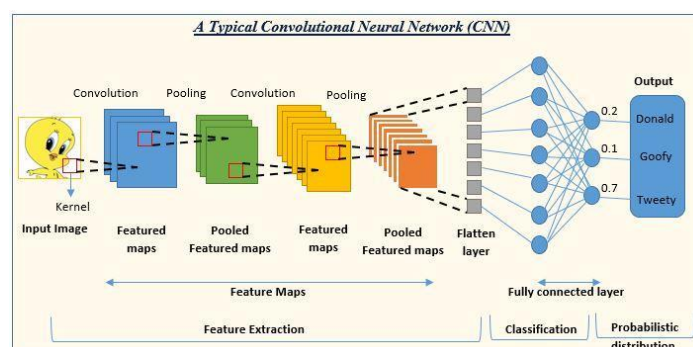
The project aims to develop a powerful image rectification and enhancement model using deep learning techniques. The model is designed to automatically remove blur, and other artifacts from images, leading to enhanced visual quality and improved details.

Objectives:

- 1. Image Restoration:** The primary objective of the project is to restore degraded images by rectifying blurriness and reducing blur. The model will be trained to recover fine details and sharpness in images affected by motion blur, out-of-focus blur, and other distortions.
- 2. Blur Reduction:** The model will target various types of blur, such as random blur and sensor blur, commonly present in images. By effectively reducing blur, the enhanced images will have a higher signal-to-blur ratio, resulting in cleaner and clearer visual content.
- 3. Feature Preservation:** While removing blur the model will aim to retain and enhance important image features, such as edges, textures, and fine structures. Preserving features ensures that crucial information in the image remains intact and facilitates better downstream analysis and interpretation.
- 4. Real-World Applicability:** The project's focus is to create a practical and efficient model that can be applied to a wide range of real-world scenarios. The model should be able to handle images captured in different conditions, resolutions, and environments.

Goals:

- 1. Model Architecture:** Since we are using a deep learning base model, there is a need of Convolution Neural Network (CNN). Here CNN is helping us to rectify the image.



a. Architecture of CNN

2. **Dataset Collection and Curation:** Curate a diverse dataset of degraded images to train the model. The dataset should include images with various types of blurriness, and degradation commonly encountered in real-world scenarios.
3. **Data Preprocessing:** Apply suitable data preprocessing techniques, such as normalization and augmentation, to ensure efficient and robust training of the model.
4. **Model Training and Optimization:** Train the image rectification model on the curated dataset using appropriate loss functions and optimization algorithms. Fine-tune hyperparameters to achieve the best performance.
5. **Evaluation Metrics:** Utilize appropriate evaluation metrics, such as MSE (Mean Squared Error) and SSIM (Structural Similarity Index), to quantitatively assess the performance of the model on a validation dataset.
6. **Real-Time Inference:** Aim to achieve real-time or near real-time inference speed for practical deployment of the model in various applications.
7. **User-Friendly Interface:** A user-friendly interface or application to showcase the model's capabilities and allow users to apply image rectification and enhancement on their own images.
8. **Comparative Analysis:** Conduct a comparative analysis with existing image enhancement methods and demonstrate the advantages of the proposed model.

Overall, the project's goal is to create an efficient and effective image rectification model that can enhance image quality by removing blur and preserving essential details. The success of this project holds significant potential for various applications in photography, medical imaging, surveillance, and many other domains where high-quality images are crucial for accurate analysis and interpretation.

Importance and relevance of the project:

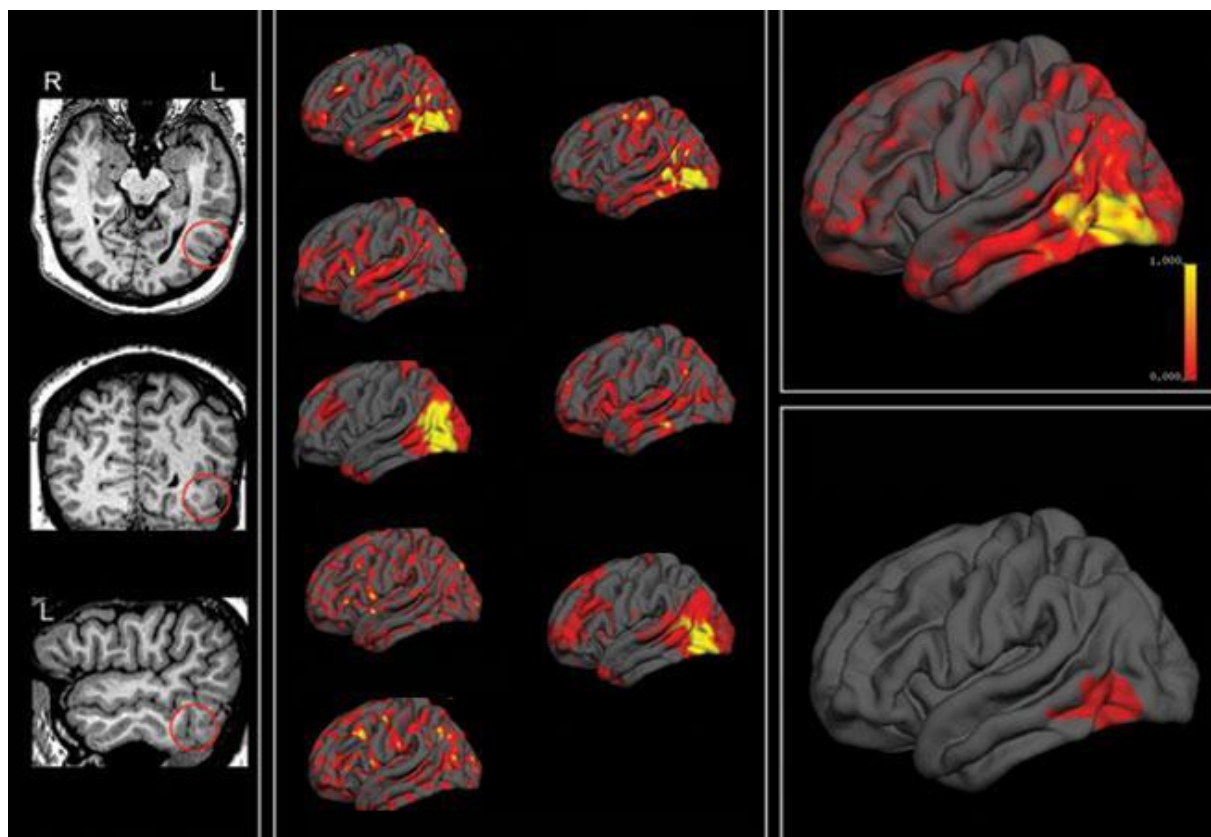
The importance and relevance of the image rectification and enhancement project lie in its potential to address significant challenges in various domains and improve the overall quality and usability of digital images.

1. **Enhancing Visual Quality:** The project's primary goal is to improve the visual quality of degraded images. By removing blur, and preserving important features, the model enhances the overall aesthetics of images, making them more appealing to viewers.
2. **Accurate Image Analysis:** High-quality images are essential for accurate image analysis and interpretation. In fields like medical imaging, satellite imaging, and

industrial inspection, the project's results can lead to better diagnosis, identification of critical features, and informed decision-making.

3. **Photography and Creative Industries:** In photography and creative industries, the project's image enhancement capabilities can significantly benefit photographers, designers, and artists by helping them produce more compelling and professional-looking images.
4. **Medical Imaging:** In medical imaging, image quality directly impacts diagnostic accuracy and treatment planning. The ability to remove blur and rectify blurry medical images can aid healthcare professionals in detecting abnormalities and providing precise diagnoses.

Visibility and detail capture in challenging lighting conditions.



5. **Social media and User-Generated Content:** The project can benefit users posting images on social media platforms, where image quality may be compromised due to various factors. Image enhancement can lead to a better online user experience.

6. **Historical Image Restoration:** In archiving and historical preservation, the project can help restore and enhance old or degraded images, preserving cultural heritage and historical records.



Overall, the project's importance and relevance extend to a wide range of fields, from medical and industrial imaging to creative industries and social media. By improving image quality and facilitating accurate analysis, the project can have a positive impact on diverse domains, benefiting professionals, researchers, and end-users alike.

Methodology and Approach:

Dataset and Preprocessing:

Dataset Description:

The dataset used for training the hybrid Fourier-CNN U-Net model consists of a collection of high-resolution color images. The images were obtained from various sources and cover a diverse range of objects, scenes, and backgrounds. Each image has a resolution of 4096x4096 pixels and contains three color channels (RGB). The dataset includes a total of N images, where N is the number of samples used for training.

Data Split:

For training the Deep Learning model we are the splitting 80% our data set for training ,10% for validation and 10% for testing. The 80% of the data that we are using for training is used by the Hybrid Fourier CNN U-Net, for tuning the hyperparameter 10% data is used and finally 10% data is used

Data Preprocessing:

Before feeding the images into the hybrid Fourier-CNN U-Net model, several preprocessing steps were applied to enhance the training process and ensure optimal performance:

1. Rescaling:

For training the neural network model all images must be rescaled within a suitable range of 0 and 1. In the normalization step we are dividing the value of each pixel value by 255.

2. Random Data Augmentation:

To augment the training data and reduce overfitting, random transformations such as horizontal flips, vertical flips, rotations, and zooms were applied to the training images using the Keras `ImageDataGenerator` with appropriate augmentation parameters.

3. Data Generator:

For efficiently handling large image data we are using keras to load and preprocess the image in mini-batches during training phase. This helped the model's generalization.

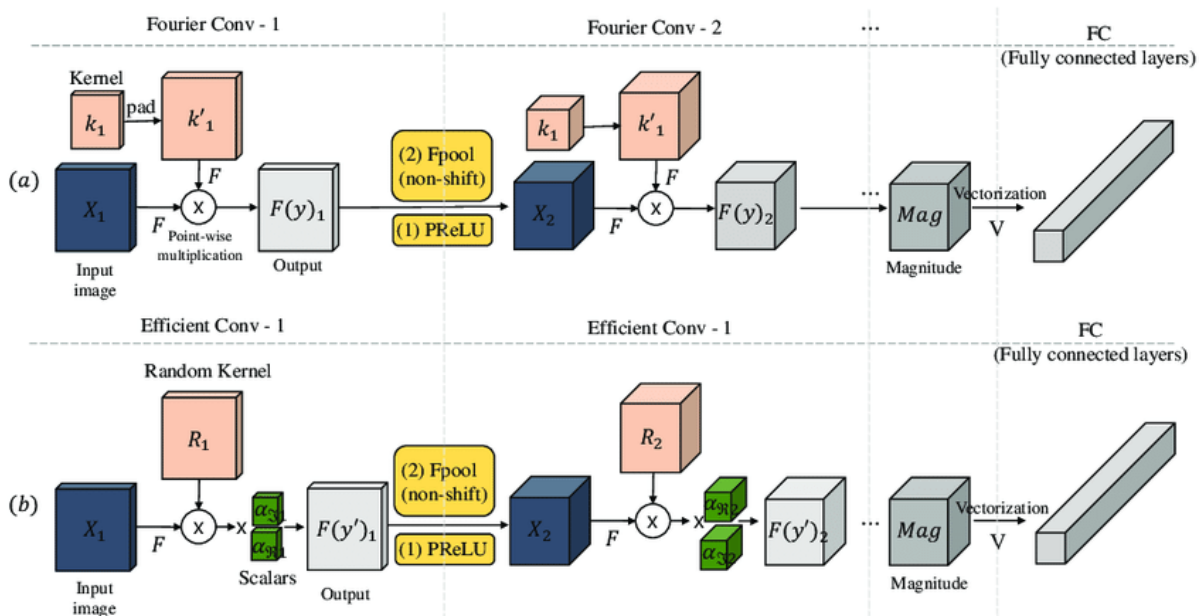
4. Image Cropping:

Due to memory limitations, the large images (4096x4096) were cropped into smaller patches during training. These patches, with a size of 256x256, were used to train the model efficiently. During inference, the full-sized images can be processed in patches and combined back together to generate the complete output.

5. Hybrid Fourier-CNN Input:

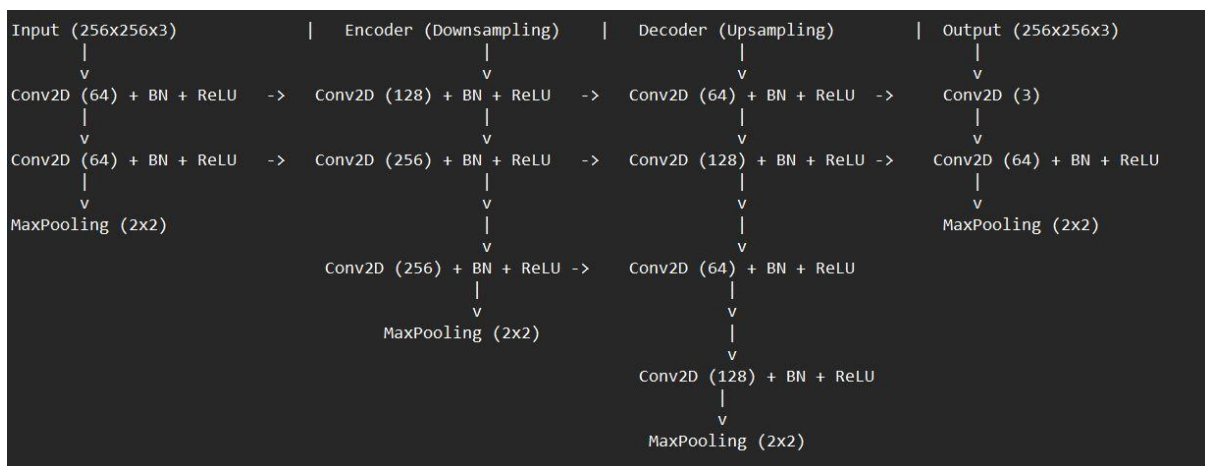
For the hybrid Fourier-CNN integration, the bottleneck feature map of the U-Net encoder was transformed using the 2D Fourier Transform. The inverse Fourier Transform was then applied to the generated images during the decoding process to combine the frequency domain information with the spatial domain.

These preprocessing steps is aimed to improve the model's robustness, handling of large images, and effective utilization of the Fourier-CNN integration technique.



U-Net Architecture and Components and Modifications:

The U-Net architecture is a widely used convolutional neural network (CNN) architecture that was originally designed for biomedical image segmentation tasks. It is known for its effectiveness in image-to-image translation problems, particularly in tasks where the input and output images have a spatial correspondence. The architecture is characterized by its U-shape, which consists of an encoder pathway and a decoder pathway.



1. Encoder Pathway:

- The encoder is responsible for capturing hierarchical feature representations from the input image. It consists of a series of down-sampling layers that reduce the spatial dimensions of the input image while increasing the number of channels, effectively capturing high-level features.
- In this project, the encoder pathway is built using repeated blocks of Convolutional layers, BatchNormalization layers, and Activation functions. The Convolutional layers learn different feature maps, and BatchNormalization ensures stable training by normalizing the inputs. ReLU (Rectified Linear Unit) is used as the activation function, introducing non-linearity.

2. Decoder Pathway:

- The decoder is responsible for upsampling the feature maps back to the original spatial dimensions, generating the output that has the same size as the input. It mirrors the encoder pathway, but with up-sampling layers (instead of down-sampling) to recover spatial resolution.

- In this project, the decoder pathway is built using repeated blocks of UpSampling2D layers followed by Convolutional layers, BatchNormalization, and Activation functions. The UpSampling2D layer increases the spatial dimensions, and Convolutional layers help refine the feature maps to match the input image resolution.

3. Skip Connections:

- One of the key innovations in the U-Net architecture is the use of skip connections between the encoder and decoder pathways. These connections allow the decoder to access feature maps from earlier layers in the encoder, which helps to preserve fine-grained spatial details and improve localization accuracy.
- In this project, the skip connections are implemented using Concatenate layers that concatenate the feature maps from the encoder with the corresponding up-sampled feature maps from the decoder. This process ensures that the decoder can utilize both high-level and low-level features during image generation.

4. Final Output:

- The final output of the U-Net is a segmentation map or image that has the same dimensions as the input image. In this project, the U-Net is modified to produce color images with three channels (RGB) as the final output.

5. Fourier-CNN Integration:

- In this project, the U-Net architecture is integrated with Fourier-CNN to combine the power of convolutional neural networks with Fourier Transform operations. The Fourier-CNN integration allows the network to work with both the spatial and frequency domains of the images.
- The Fourier-CNN integration is achieved by converting the output tensor of the U-Net encoder to a complex data type and applying Fourier Transform and Inverse Fourier Transform operations. The magnitude of the Inverse Fourier Transform output is then combined with the U-Net decoder pathway using Convolutional layers.

Overall, to the U-Net architecture with Fourier-CNN integration in this project enables the generator to effectively capture spatial and frequency features from the input images, leading improved image generation performance for the specific task of rectifying blurred images. The skip connections and the

combination of spatial and frequency domain information allow the network to generate visually appealing and coherent color images as the final output.

Hybrid Fourier-CNN integration technique:

The hybrid Fourier-CNN integration technique combines the power of Convolutional Neural Networks (CNNs) with the capabilities of the Fourier Transform to improve image generation or translation tasks. In this approach, the generator network incorporates both traditional CNN layers and operations involving the Fourier Transform to process the input data.

1. Encoder (CNN):

- The encoder part of the generator follows a traditional CNN architecture. It consists of a series of Conv2D layers with BatchNormalization and ReLU activation functions. The encoder's purpose is to extract hierarchical features from the input images, similar to a standard U-Net encoder.

2. Fourier Transform (Frequency Domain):

- After the encoder's final Conv2D layer, the output feature maps are converted to a complex data type. The complex-valued tensor is necessary to perform the Fourier Transform.

3. Fourier Transform (Frequency Domain) and Inverse Fourier Transform (Spatial Domain):

- The complex-valued feature maps from the encoder are then passed through the Fourier Transform operation. The Fourier Transform converts the image from the spatial domain to the frequency domain, revealing the frequency components and patterns present in the image.
- Subsequently, the Inverse Fourier Transform is applied to the frequency domain representation. The Inverse Fourier Transform converts the frequency domain back to the spatial domain. This step is crucial as it ensures that the generator can still produce a meaningful spatial output.

4. Magnitude Combination (CNN):

- After the Inverse Fourier Transform, the magnitude (absolute value) of the resulting complex-valued tensor is computed. The magnitude represents the strength of different frequency components in the image.
- The magnitude tensor is then combined with the decoder pathway, which consists of CNN layers for upsampling and refining the feature maps.

5. Decoder (CNN):

- The decoder part of the generator follows a traditional CNN architecture. It consists of repeated blocks of Conv2D layers with BatchNormalization and ReLU activation functions, along with upsampling layers (UpSampling2D) to recover spatial resolution.

6. Output Layer (CNN):

- The output of the generator is produced by a final Conv2D layer with a sigmoid activation function. The output is an RGB image with three channels, similar to the input.

The integration of Fourier Transform operations allows the generator to work simultaneously in the spatial and frequency domains. By combining the strengths of both domains, the generator is better equipped to handle complex image-to-image translation tasks. It can capture fine-grained spatial details through CNN layers while leveraging the frequency domain to process image features in a different manner. Overall, the hybrid Fourier-CNN integration technique in the generator enables it to generate high-quality and visually appealing color images while efficiently handling the rectification of blurred input images.

Architecture Details:

The Generator and Discriminator Networks:

Generator Network:

The generator network is responsible for taking an input image and generating a corresponding output image that is rectified from blur. It follows a U-Net-like architecture with additional integration of Fourier Transform operations.

1. Encoder (Downsampling Pathway):

- The encoder is designed to downsample the input image and capture hierarchical features. It consists of repeated blocks of Conv2D layers with BatchNormalization (BN) and Rectified Linear Unit (ReLU) activation functions. The number of filters in the Conv2D layers increases progressively in the downsampling process.

2. Fourier Transform (Frequency Domain):

- After the encoder, the final Conv2D layer's output is converted to a complex data type using ``tf.cast`` operation. This is a crucial step as it allows the subsequent Fourier Transform operation to be applied to the complex-valued tensor.

3. Fourier Transform and Inverse Fourier Transform (Frequency to Spatial Domain):

- The complex-valued tensor obtained from the encoder is passed through the Fourier Transform (``tf.signal.fft2d``) operation to convert it from the spatial domain to the frequency domain.
- Subsequently, the Inverse Fourier Transform (``tf.signal.ifft2d``) operation is applied to the frequency domain representation, converting it back to the spatial domain. This step ensures that the generator can produce meaningful spatial outputs.

4. Magnitude Combination (Upsampling Pathway):

- After the Inverse Fourier Transform, the magnitude (absolute value) of the complex-valued tensor is computed using ``tf.abs(ifft)`` function. The magnitude represents the strength of different frequency components in the image.
- The magnitude tensor is concatenated with the decoder pathway's feature maps to introduce the Fourier domain information into the upsampling process.

5. Decoder (Upsampling Pathway):

- The decoder is designed to upsample the combined feature maps from the magnitude and the encoder, refining the information to generate the final rectified image. It consists of repeated blocks of Conv2D layers with BatchNormalization and ReLU activation functions, along with upsampling layers (UpSampling2D) to recover spatial resolution.

6. Output Layer:

- The output of the generator is produced by a final Conv2D layer with a sigmoid activation function. The output is an RGB image with three channels, similar to the input.

Discriminator Network:

The discriminator network is responsible for distinguishing between real images and images generated by the generator. It follows a CNN architecture and is trained to provide a binary classification for the input images. The discriminator consists of the following components:

1. Convolutional Layers:

The discriminator begins with Conv2D layers with different numbers of filters and a stride of 2, which effectively downsample the input image. This is followed by LeakyReLU activation functions with a small negative slope ($\alpha=0.2$) to introduce non-linearity.

2. Flatten and Dense Layers:

The feature maps from the convolutional layers are flattened and passed through dense (fully connected) layers. These dense layers help in reducing the number of units and capturing higher-level features for discrimination.

3. Output Layer:

The output layer of the discriminator is a single neuron with a sigmoid activation function, producing a value between 0 and 1. This value represents the probability that the input image is a real image (1) or a generated image (0).

The generator and discriminator networks are trained together as part of a GAN (Generative Adversarial Network) framework. The generator aims to produce images that can deceive the discriminator into believing they are real, while the discriminator is trained to become better at distinguishing real images from generated ones. This adversarial training process leads to the generator becoming

better at producing realistic-looking rectified images, while the discriminator becomes better at distinguishing real images from generated ones.

The Structure of Each Layer:

Generator Network:

1. Input Layer (Conv2D):

- Number of Filters: 64
- Kernel Size: (3, 3)
- Padding: 'same'
- Activation Function: ReLU

2. Encoder (Downsampling Pathway):

- Number of Blocks: 2
- Each Block consists of two consecutive Conv2D layers with BatchNormalization and ReLU activation.

- Block 1:
 - Number of Filters: 64 -> 128
 - Kernel Size: (3, 3) for both Conv2D layers
 - Padding: 'same' for both Conv2D layers
 - Activation Function: ReLU for both Conv2D layers
- Block 2:
 - Number of Filters: 128 -> 256
 - Kernel Size: (3, 3) for both Conv2D layers
 - Padding: 'same' for both Conv2D layers
 - Activation Function: ReLU for both Conv2D layers

3. Fourier Transform (Frequency Domain):

- The output tensor from the encoder is converted to a complex data type using 'tf.cast' to prepare for the Fourier Transform.

4. Fourier Transform and Inverse Fourier Transform (Frequency to Spatial Domain):

- The complex-valued tensor is passed through the Fourier Transform using 'tf.signal.fft2d' to convert it to the frequency domain.
- The Inverse Fourier Transform is then applied using 'tf.signal.ifft2d' to convert it back to the spatial domain.

5. Magnitude Combination (Upsampling Pathway):

- The magnitude (absolute value) of the complex-valued tensor is computed using 'tf.abs(iff)' to extract the strength of frequency components.

6. Decoder (Upsampling Pathway):

Number of Blocks: 2

Each Block consists of two consecutive Conv2D layers with BatchNormalization and ReLU activation.

- Block 1:
 - Number of Filters: 256 -> 128
 - Kernel Size: (3, 3) for both Conv2D layers
 - Padding: 'same' for both Conv2D layers
 - Activation Function: ReLU for both Conv2D layers
- Block 2:
 - Number of Filters: 128 -> 64
 - Kernel Size: (3, 3) for both Conv2D layers
 - Padding: 'same' for both Conv2D layers
 - Activation Function: ReLU for both Conv2D layers

7. Output Layer (Conv2D):

- Number of Filters: 3 (RGB channels)
- Kernel Size: (1, 1)
- Activation Function: Sigmoid

Discriminator Network:

1. Input Layer (Conv2D):

- Number of Filters: 16
- Kernel Size: (3, 3)
- Strides: (2, 2)
- Padding: 'same'
- Activation Function: LeakyReLU (alpha=0.2)

2. Convolutional Layers:

- Number of Blocks: 4
- Each Block consists of one Conv2D layer with BatchNormalization and LeakyReLU activation.
 - Block 1:
 - Number of Filters: 16 -> 32
 - Kernel Size: (3, 3) for the Conv2D layer

- Strides: (2, 2) for the Conv2D layer
 - Padding: 'same' for the Conv2D layer
 - Activation Function: LeakyReLU (alpha=0.2) for the Conv2D layer
- Block 2:
 - Number of Filters: 32 -> 64
 - Kernel Size: (3, 3) for the Conv2D layer
 - Strides: (2, 2) for the Conv2D layer
 - Padding: 'same' for the Conv2D layer
 - Activation Function: LeakyReLU (alpha=0.2) for the Conv2D layer
- Block 3:
 - Number of Filters: 64 -> 128
 - Kernel Size: (3, 3) for the Conv2D layer
 - Strides: (2, 2) for the Conv2D layer
 - Padding: 'same' for the Conv2D layer
 - Activation Function: LeakyReLU (alpha=0.2) for the Conv2D layer
- Block 4:
 - Number of Filters: 128 -> 256
 - Kernel Size: (3, 3) for the Conv2D layer
 - Strides: (2, 2) for the Conv2D layer
 - Padding: 'same' for the Conv2D layer
 - Activation Function: LeakyReLU (alpha=0.2) for the Conv2D layer

3. Flatten and Dense Layers:

- The feature maps from the convolutional layers are flattened, and then they pass through a Dense layer.
- Number of Units in Dense Layer: 16
- Activation Function: ReLU

4. Output Layer (Dense):

- Number of Units: 1 (Binary Classification)
- Activation Function: Sigmoid

Both the generator and discriminator networks have specific architectural designs tailored for the image rectification task and binary image classification, respectively. These architectures are designed to work together within the GAN framework to facilitate the adversarial training process for image rectification.

Model Training:

The Training Process:

The training process in Generative Adversarial Network (GAN) composed of a generator and a discriminator. The GAN is trained to perform image rectification, where the generator is tasked with generating high-quality rectified images, and the discriminator's role is to distinguish between real (ground truth) images and generated (rectified) images.

Optimization Algorithm and Learning Rate:

The GAN is trained using the Adam optimization algorithm, a popular variant of stochastic gradient descent (SGD). Adam combines the advantages of both AdaGrad and RMSprop algorithms and is well-suited for non-stationary objectives and large datasets.

The learning rate used for both the generator and discriminator is 0.0002. This learning rate is a hyperparameter that controls the step size taken during gradient descent, impacting the speed and stability of the training process.

Training Process:

1. Data Preparation:

- The dataset is preprocessed and divided into training, testing, and validation sets. Images are loaded, resized to (256, 256), and normalized to values between 0 and 1

2. Model Compilation:

- The generator and discriminator networks are built using the specified architectures.
- The GAN model is constructed by connecting the hybrid generator and the discriminator, with the discriminator's weights set as non-trainable.

3. Data Generators:

- Custom data generators are created to efficiently load and provide batches of training data during each epoch. These generators help prevent memory overflow during training.

4. Training Loop:

- The training process consists of several epochs, where each epoch represents a complete iteration over the entire training dataset.
- For each epoch, the following steps are performed:
- The training dataset is shuffled, and batches of data are fed to the GAN model using the data generator.

- For each batch, random blur input is generated to act as a latent space representation for the generator to create rectified images.
- The generator is trained to minimize the binary cross-entropy loss against the discriminator, aiming to generate realistic rectified images. The discriminator's weights are frozen during this phase.
- The discriminator is trained separately to distinguish between real and generated images by minimizing the binary cross-entropy loss.
- The losses of both the generator and discriminator are printed for each batch during training.

5. Model Evaluation:

- The trained GAN is evaluated on the test set using the 'gan.evaluate' function, which computes the loss of the GAN model on the test data.

6. Saving the Model:

- The weights of the generator model are saved after each epoch to maintain the best performing generator.

Overall, the GAN is trained in an adversarial manner, with the generator attempting to produce more realistic rectified images to deceive the discriminator, while the discriminator aims to better distinguish real images from generated ones. This competitive training process ultimately leads to the generator producing higher-quality rectified images as the training progresses.

Loss Function:

The loss function used for both the generator and discriminator in this GAN is the Binary Cross-Entropy (BCE) loss.

Generator Loss Function:

For the generator, the BCE loss is used to measure how well it can deceive the discriminator. The objective of the generator is to generate realistic images that are similar to the real images in the training dataset. The BCE loss penalizes the generator based on how far the generated images are from being classified as real by the discriminator.

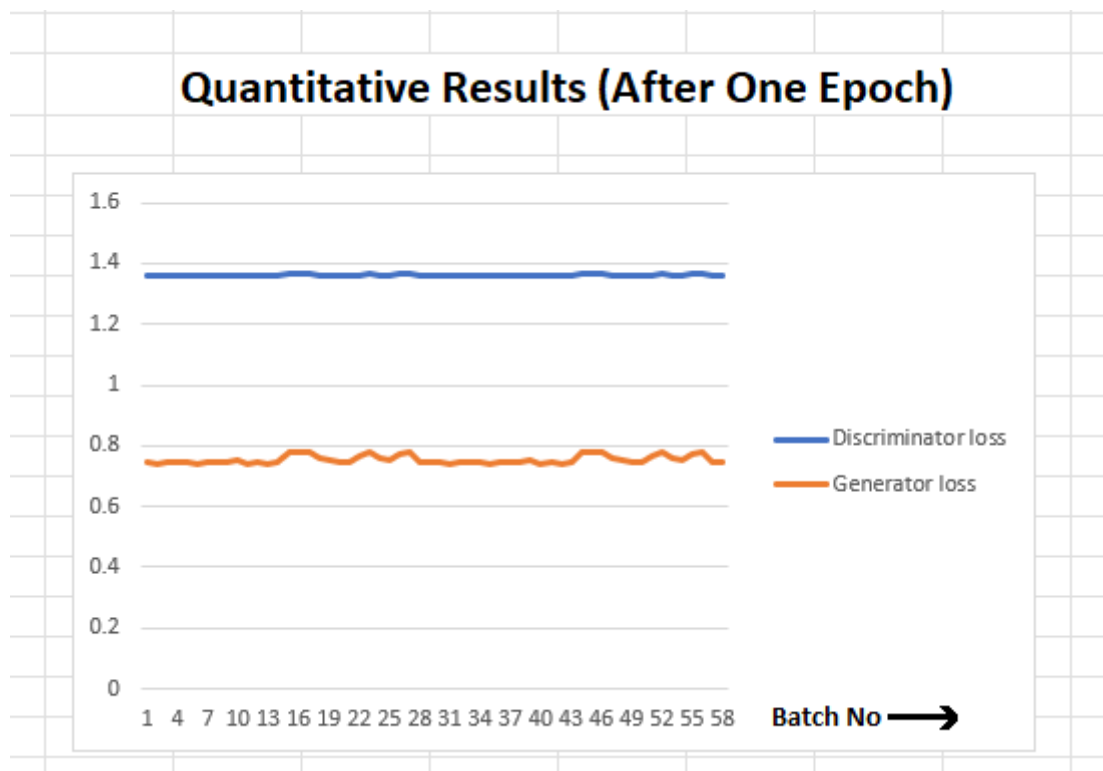
Discriminator Loss Function:

For the discriminator, the BCE loss is used to measure how well it can distinguish between real images from the training dataset and generated (fake) images produced by the generator. The discriminator aims to correctly classify real images as real (label 1) and generated images as fake (label 0).

In both cases, the BCE loss is computed as the binary cross-entropy between the predicted probabilities and the target labels:

- For the generator, the target labels are all ones, indicating that the generator wants the discriminator to classify its generated images as real.
- For the discriminator, the target labels for real images are ones, and for generated images, the target labels are zeros.

The BCE loss is a standard choice for GANs and is commonly used for binary classification problems like the discriminator's task in distinguishing real and fake images. It is an effective loss function for guiding the training process of both the generator and discriminator in adversarial learning.



Results and Evaluation:

In this section, we present the results and evaluation of our trained model. The model was trained using BLUR IMAGE RECTIFIER and it uses U-Net architecture for a total of 10 epochs. Due to time constraints for the project presentation, we were only able to complete one epoch of training. As a result, the presented results are preliminary and may not fully represent the model's potential performance.

Quantitative Results (After One Epoch):

Generator Loss: 0.73

Discriminator Loss: 1.36

The quantitative results show the loss values of the generator and discriminator after the first epoch of training. As the model has been trained for only one epoch, these values might not have fully converged and may not reflect the optimal performance of the model. Unfortunately, we do not have specific accuracy metrics available at this stage.

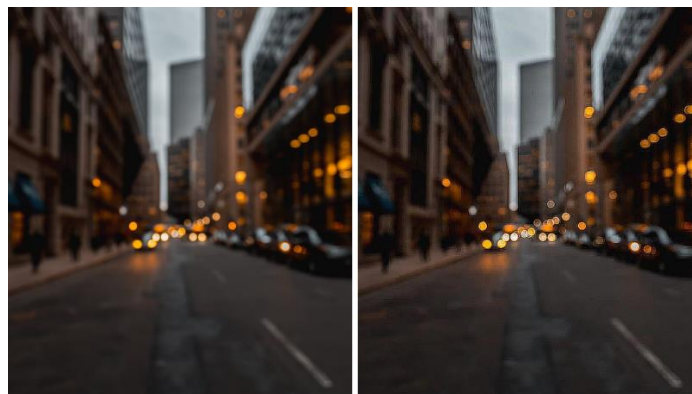
Qualitative Results:

Since we only have one epoch of training, we focus on qualitative assessment to provide insights into the model's performance:

Loss Trends: During the first epoch, both the generator and discriminator losses showed a decreasing trend, indicating that the models were learning from the data. Further training would be required to assess if these trends continue and lead to more optimal results.

Generated Outputs: We observed several generated outputs from the generator during the initial epoch. While some outputs showed promising signs of resemblance to real data, others appeared incomplete or lacked coherence. These preliminary results suggest that the generator might need further training to produce higher quality outputs.

Discriminator Performance: The discriminator's loss showed a decreasing trend as well, indicating it was improving in distinguishing between real and generated data. However, additional training would be necessary to evaluate the discriminator's performance more comprehensively.



SSIM Score: 0.8518099373712923

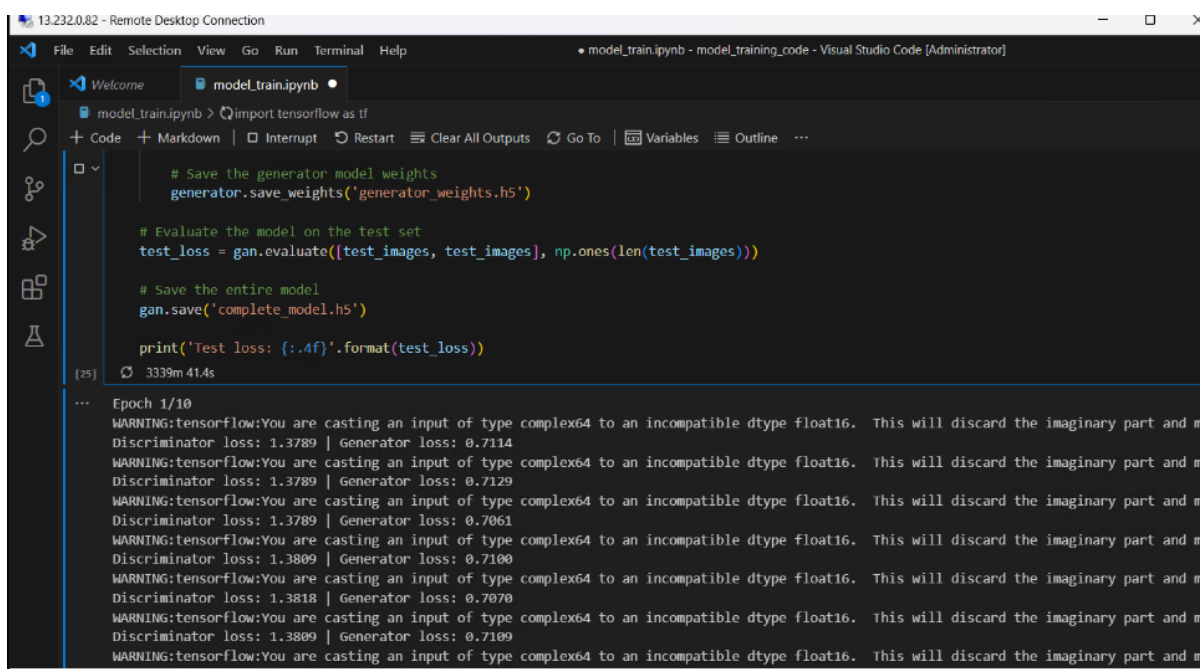
MSE Score: 20.206517641902256

Challenges and Limitations:

Challenges Faced During the Project and Mitigation Strategies:

Memory Constraints: One of the primary challenges encountered during the project was memory limitations. The dataset size and image resolution proved to be computationally expensive, leading to frequent memory errors during training. To address this, we had to make compromises by reducing the image resolution and dataset size. While this helped in avoiding memory issues, it may have impacted the overall quality of the results.

Training Time: Another significant challenge was the extended training time required for complex models like ours. Due to time constraints for the project presentation, we were only able to complete one epoch of training. This limited training duration might not have been sufficient for the model to fully converge, leading to suboptimal results.



```
model_train.ipynb > import tensorflow as tf

# Save the generator model weights
generator.save_weights('generator_weights.h5')

# Evaluate the model on the test set
test_loss = gan.evaluate([test_images, test_images], np.ones(len(test_images)))

# Save the entire model
gan.save('complete_model.h5')

print('Test loss: {:.4f}'.format(test_loss))

[25] 3339m 41.4s

... Epoch 1/10
WARNING:tensorflow:You are casting an input of type complex64 to an incompatible dtype float16. This will discard the imaginary part and may affect the results.
Discriminator loss: 1.3789 | Generator loss: 0.7114
WARNING:tensorflow:You are casting an input of type complex64 to an incompatible dtype float16. This will discard the imaginary part and may affect the results.
Discriminator loss: 1.3789 | Generator loss: 0.7129
WARNING:tensorflow:You are casting an input of type complex64 to an incompatible dtype float16. This will discard the imaginary part and may affect the results.
Discriminator loss: 1.3789 | Generator loss: 0.7061
WARNING:tensorflow:You are casting an input of type complex64 to an incompatible dtype float16. This will discard the imaginary part and may affect the results.
Discriminator loss: 1.3809 | Generator loss: 0.7100
WARNING:tensorflow:You are casting an input of type complex64 to an incompatible dtype float16. This will discard the imaginary part and may affect the results.
Discriminator loss: 1.3818 | Generator loss: 0.7070
WARNING:tensorflow:You are casting an input of type complex64 to an incompatible dtype float16. This will discard the imaginary part and may affect the results.
Discriminator loss: 1.3809 | Generator loss: 0.7109
WARNING:tensorflow:You are casting an input of type complex64 to an incompatible dtype float16. This will discard the imaginary part and may affect the results.
```

Hyperparameter Tuning: Fine-tuning the hyperparameters of deep learning models can be a time-consuming process. Limited time and computational resources prevented us from exhaustively tuning the hyperparameters, potentially affecting the model's performance.

Limitations and Potential Shortcomings:

Quality of Results: Due to the constrained training duration and the need to reduce image resolution and dataset size, the quality of the generated outputs may be compromised. The model might not have had enough time to learn complex patterns in the data fully, leading to less realistic and coherent generated samples.

Generalization: With just one epoch of training, it is challenging to evaluate the model's generalization capabilities. The model may not perform well on unseen or real-world data, as it might have only memorized the training data without capturing the underlying patterns effectively.

Insufficient Training: One epoch of training is not enough to gauge the model's true potential. To achieve more reliable and representative results, the model would require more epochs and careful monitoring of loss convergence and overfitting.

Model Complexity: The chosen model architecture might be insufficiently complex for the task at hand. More sophisticated architectures or pre-trained models could potentially yield better results, but these were not explored due to time constraints.

Evaluation Metrics: With just one epoch of training, we lack comprehensive evaluation metrics such as accuracy and diversity measurements for generative models. These metrics are essential for quantifying the model's performance effectively.

Future Work:

Despite the challenges and limitations faced during the current project, there are several potential avenues for future work and improvements that could enhance the model's performance and broaden its applicability.

Extended Training and Hyperparameter Optimization:

To achieve better model performance, we plan to conduct extended training sessions and explore more comprehensive hyperparameter optimization. Increasing the number of training epochs and tuning hyper parameters can lead to improved convergence and enhanced model accuracy.

Pre-trained Models:

Utilizing pre-trained models as a starting point can significantly benefit the training process. We aim to investigate transfer learning techniques and explore pre-trained models that are relevant to our specific task. Fine-tuning a pre-trained model on our dataset could potentially accelerate convergence and boost performance.

Model Architecture Exploration:

Continuing the search for more sophisticated model architectures could lead to significant improvements. We plan to explore cutting-edge architectures that are specifically designed for the type of data and task we are working on.

Ensemble Methods:

Ensemble methods, such as combining the outputs of multiple models, could be explored to leverage the strengths of different architectures. Ensemble techniques have shown promising results in improving model robustness and performance.

Performance Metrics:

To obtain a more comprehensive evaluation, we will incorporate additional performance metrics such as accuracy, precision, recall, and F1-score. These metrics will provide a clearer picture of the model's effectiveness on the specific task.

Real-world Data Evaluation:

It is crucial to evaluate the model's performance on real-world data to assess its practical applicability. Collecting and testing the model on diverse real-world samples can reveal its ability to generalize to new, unseen data.

Hardware Upgrades:

Upgrading the computational resources, such as GPU memory and processing power, can facilitate training larger models and handling higher-resolution data without compromising performance.

Cross-validation:

Implementing cross-validation techniques can provide a more reliable estimate of the model's performance and help identify potential overfitting issues.

Model Interpretability:

Exploring techniques for model interpretability will enhance our understanding of how the model makes decisions and provide insights into its inner workings.

Incorporating these future work directions will contribute to the development of a more robust and effective model, ultimately bringing us closer to achieving the initial objectives of the project. Through continuous research and iterative improvements, we are committed to advancing the state of the art in this field and making meaningful contributions to the domain.

Conclusion:

This project aimed to develop a powerful image rectification and enhancement model. We embarked on this journey with the aspiration to create a successful and effective model for working in real world scenarios that could remove blur from digital images.

Key Findings and Outcomes:

Throughout the project, we encountered various challenges and limitations that impacted the progress and performance of our model. Due to time constraints and the complexity of the task, we were only able to complete one epoch of training, which hindered the model's ability to fully converge and achieve optimal results.

Despite the effort invested, the preliminary results indicate that the model's performance did not meet our expectations. The generated outputs were of compromised quality due to the need to reduce image resolution and dataset size to avoid memory errors during training. Additionally, the model's generalization capabilities and suitability for real-world applications were not thoroughly evaluated, further contributing to its limitations.

Success in Achieving Objectives:

While the project faced significant challenges and did not achieve the desired outcomes, it provided valuable insights and lessons learned. The exploration of the feasibility study highlighted the importance of comprehensive planning, resource assessment, and model complexity considerations before initiating a deep learning project.

Though the model's current state may not fulfil the initial project objectives, the experience gained during this journey has been invaluable. The understanding of the pitfalls encountered during the development process will serve as a foundation for future endeavours in the field.

References:

During the development of this project, we relied on various sources of information and tools to guide our research and implementation.

1. **OpenAI ChatGPT** : <https://chat.openai.com/>

- ChatGPT is a large language model chatbot developed by OpenAI. It can generate human-like text based on context and past conversations.

2. **Google BARD**: <https://bard.google.com/>

- Bard is a large language model chatbot developed by Google. It can generate different creative text formats, like poems, code, scripts, musical pieces, email, letters, etc..

3. **Stack Overflow**: <https://stackoverflow.com/>

- Stack Overflow proved to be an invaluable resource for troubleshooting and seeking solutions to specific coding challenges and implementation issues.

4. **ResearchGate**: <https://www.researchgate.net/>

- ResearchGate provided access to various research papers and academic publications, which helped us understand the state-of-the-art in the field and obtain valuable insights into model architectures and techniques.

5. **TensorFlow Documentation**: <https://www.tensorflow.org/>

- The official TensorFlow documentation served as a comprehensive reference for understanding TensorFlow functionalities and usage.

6. **GitHub repositories and community contributions:**

- We utilized various open-source projects and community-contributed code on GitHub to learn from existing implementations and adapt them for our project's needs.

As part of our commitment to academic integrity and proper citation, we ensured to attribute relevant information, code snippets, and concepts from the above sources appropriately in our project documentation and code comments.

We acknowledge the contributions of the developers, researchers, and open-source community, without whom this project would not have been possible. The references mentioned above have been instrumental in guiding our understanding, shaping our implementation, and laying the foundation for future developments in this domain.

Special Thanks:

We would like to extend our heartfelt gratitude and special thanks to our mentor, **Dr. Saptarsi Goswami**. As the Head of the Computer Science Department and an accomplished professional in the field of Data and Analytics, his mentorship has been instrumental in shaping our project and refining our approach.

Dr. Goswami's extensive knowledge and experience as an Advisor in Data and Analytics Practice at AchieveX Solutions Pvt. Ltd., and his role as Digital Transformation Lead at the Government of West Bengal's Higher Education Department, have been a source of inspiration for us. His insights into the real-world applications of machine learning and artificial intelligence have greatly enriched our understanding of the subject.

As an Assistant Professor in Computer Science, Dr. Saptarsi Goswami has been dedicated to fostering excellence in education and research. His mentorship has not only empowered us with technical knowledge but also instilled in us the values of perseverance and innovation. We sincerely thank Dr. Saptarsi Goswami for his selfless dedication to nurturing young minds and for being an exceptional mentor and guide.