Generative AI Assignment 2: Generative Modeling for Signature Synthesis and Image-to-Image Translation Using VAE, GAN, Conditional GAN, and CycleGAN

Imama Amjad
BS CS
FAST-NUCES University
Islamabad, Pakistan
i201819@nu.edu.pk

***Abstract*—This assignment explores the implementation of Variational Autoencoders (VAE), Generative Adversarial Networks (GAN), Conditional GAN (cGAN), and CycleGAN across multiple datasets, each focusing on unique applications. In the first task, a VAE and a simple GAN are developed to generate synthetic signatures from a latent space, where signature augmentation techniques such as scaling, rotation, and noise addition are applied to increase dataset diversity. The models are trained and evaluated using reconstruction loss and qualitative visual comparison. The second task introduces a custom GAN applied to the CIFAR-10 dataset, focusing on cats and dogs. A specialized discriminator network computes a similarity score between real and generated images, enhancing the realism of the generated outputs. The third task implements a Conditional GAN (cGAN) that generates realistic face images from sketches using the Person Face Sketches dataset, emphasizing the conditional mapping between input sketches and real faces. Lastly, the CycleGAN is utilized to perform image-to-image translation between sketches and real face images, achieving bidirectional transformations between sketches and corresponding face images. These models demonstrate the power of generative architectures in creative tasks involving image generation and transformation.**

***Index Terms*—Variational Autoencoder (VAE), Generative Adversarial Network (GAN), Conditional GAN (cGAN), CycleGAN, Image Generation, Signature Recognition, Data Augmentation, CIFAR-10, Person Face Sketches, Neural Networks, Image-to-Image Translation, Similarity Scoring, Discriminator Network**

I. Introduction

A. Question 1: VAE and Simple GAN Implementation

The first task in this assignment focuses on implementing a Variational Autoencoder (VAE) and a simple GAN to generate synthetic signatures. Given a potentially limited dataset of real signatures, data augmentation techniques such as scaling, rotation, and noise addition are employed to diversify the dataset. The VAE and GAN are trained separately to learn latent representations of the augmented signatures and generate new signatures by sampling from the learned latent space. The generated outputs are evaluated using metrics like reconstruction loss and compared visually with the original dataset to assess their quality and similarity to real signatures.

B. Question 2: Custom GAN Implementation for CIFAR-10

In this task, a custom GAN model is trained on the CIFAR-10 dataset, specifically focusing on the cats and dogs classes. Unlike a typical GAN, where the discriminator differentiates between real and fake images, this custom GAN leverages a discriminator that evaluates the similarity between real and generated images. The network computes a similarity score, and the generator is trained to minimize this score by producing images that closely resemble real ones. This modification to the discriminator enhances the model's ability to generate high-quality and realistic images that closely match the structure and features of the original dataset.

C. Question 3: Conditional GAN (cGAN) Implementation for Person Face Sketches

The third task involves the implementation of a Conditional GAN (cGAN) to generate realistic face images from input sketches. The model is trained using the Person Face Sketches dataset, where the generator learns to map sketches to their corresponding real face images. The cGAN leverages the conditional input of the sketch to produce accurate and visually coherent face outputs. The primary goal is to enable the model to generalize well on unseen sketches, generating realistic face images that match the given sketch input in terms of features and expressions.

D. Question 4: CycleGAN Implementation for Person Face Sketches

The final task involves implementing a CycleGAN to perform image-to-image translation between sketches and real face images. The CycleGAN architecture enables bidirectional transformation, allowing the conversion of a face image into a sketch and vice versa. Using the Person Face Sketches dataset, the model learns to preserve features across both directions

of the transformation, producing visually consistent outputs that resemble the input images. The CycleGAN is trained for multiple epochs, with regular monitoring and hyperparameter tuning to optimize its performance on generating realistic face images and sketches.

II. Methodology

A. Generative Modeling of Signatures Using Variational Autoencoders and Generative Adversarial Networks with Augmentation Techniques

1) Dataset: The dataset used for this task consisted of real-world signature images. Due to the limited size of the dataset, augmentation techniques were applied to increase its diversity. The augmentation included transformations such as scaling, rotation, and noise addition. This resulted in a sufficiently large dataset to train the VAE and GAN models effectively.

2) Pre-processing Steps: Each signature image was pre-processed by resizing it to a 128x128 grayscale format. The images were normalized to fall within the range of [0,1]. Augmentation techniques were applied to generate variations of the original signatures, ensuring the models could learn from diverse representations.

3) Model Architecture:

- **VAE Architecture:** The encoder used convolutional layers to map the signature images to a lower-dimensional latent space. The decoder was designed to reconstruct the original image from this latent space. KL divergence and reconstruction loss were used to train the VAE.
- **GAN Architecture:** The GAN consisted of a generator and a discriminator. The generator was designed to map random noise vectors from the latent space into signature images, while the discriminator aimed to distinguish between real and fake signatures. The models were trained alternately in a min-max fashion.

B. Custom GAN Implementation for CIFAR-10

1) Dataset and Pre-processing: he dataset used for this GAN training was the CIFAR-10 dataset, specifically filtered to include only images of cats and dogs, represented by the labels 3 and 5. The data was pre-processed using standard image normalization techniques, scaling the pixel values between -1 and 1. This was done by transforming the dataset into tensors and normalizing with means and standard deviations of 0.5. This process helps in stabilizing the training and improves convergence rates.

2) Model Architecture: The GAN architecture implemented consisted of two key components:

- **Generator:** The generator was built using the DCGAN architecture with a latent input vector of size 100. The network utilized transpose convolutional layers to upsample the input into a 32x32x3 image. Batch normalization and ReLU activation were applied in each layer, except the last layer where a Tanh activation was used to ensure the pixel values were in the range [-1, 1].
- **Siamese Discriminator:** The discriminator was designed as a Siamese network, where it processed both real and generated images through a shared convolutional network. This network computed two outputs: an adversarial score for classifying real/fake images, and a similarity score that measured the distance between real and generated images. The architecture included convolutional layers with LeakyReLU activations and batch normalization.

3) Training: The models were trained using the Adam optimizer with a learning rate of 0.0002 and momentum parameters betas of (0.5, 0.999) for both the generator and discriminator. The Binary Cross-Entropy Loss (BCELoss) was applied to train both networks. The loss function for the discriminator combined the adversarial loss and similarity-based score, while the generator minimized both the adversarial and dissimilarity losses.

Checkpoints were saved every 10 epochs, and the best generator model was saved based on the lowest generator loss. Throughout training, losses for both the generator and discriminator were recorded for later analysis.

C. Conditional GAN Implementation for for Person Face Sketches

1) Dataset: The dataset used in this project is the Person Face Sketches dataset available on Kaggle. It consists of paired images of human sketches and corresponding real photos. The dataset is divided into three parts:

- **Training Set:** Contains 20.7k images each in the sketches and photos subfolders.
- **Validation Set:** Contains 1000 images each in the sketches and photos subfolders for model validation.
- **Test Set:** Contains 679 images each in the sketches and photos subfolders for testing the model's performance.

Each sketch in the dataset is grayscale, while the corresponding photo is in RGB format. The dataset is designed for the task of conditional image generation, where the model learns to map a sketch to its corresponding real photo.

2) Pre-processing: The dataset was pre-processed to ensure consistency in the input format:

- **Resizing:** Contains 20.7k images each in the sketches and photos subfolders.
- **Normalization:** Both sketches and photos were normalized to have pixel values between -1 and 1 using a mean and standard deviation of 0.5, which helps in stabilizing GAN training.
- **Transformation:** The sketches were loaded as grayscale images, while the photos were loaded as RGB images to match their respective formats.

A custom dataset class, SketchPhotoDataset, was implemented using PyTorch to handle the loading and transformation of both sketch and photo pairs during training.

3) Model Architecture: The Conditional GAN (cGAN) architecture used for this task consists of two main components: the U-Net Generator and the PatchGAN Discriminator.

- **U-Net Generator:** The generator is responsible for converting input sketches into realistic face images. We used

a U-Net architecture with skip connections to preserve high-level features from the input sketch while generating the output image. The architecture follows an encoder-decoder structure:
  - **Encoder (Downsampling Blocks):** The generator starts with a series of downsampling blocks (conv layers) that reduce the spatial dimensions of the input, extracting key features from the sketch.
  - **Decoder (Upsampling Blocks):** The decoder uses upsampling layers to progressively reconstruct the image from the encoded features.
  - **Skip Connections:** To retain spatial information, skip connections were introduced between corresponding downsampling and upsampling layers.
  - **Final Output:** The final layer of the generator outputs an RGB image, using a tanh activation function to map the pixel values to the range [-1, 1].
- **PatchGAN Discriminator:** The discriminator is a PatchGAN that works by evaluating patches of the image rather than the entire image at once, which helps in distinguishing real and fake image details at a local level:
  - **Input:** The discriminator takes both the sketch and the real/generated photo as inputs.
  - **Architecture:** The PatchGAN discriminator is composed of convolutional layers that progressively reduce the spatial dimensions, classifying each patch of the image as real or fake.
  - **Output:** The discriminator outputs a matrix of probabilities, where each value corresponds to whether a patch of the image is real or generated.

4) Training Process: The model was trained using the Conditional GAN (cGAN) framework, where the generator aims to fool the discriminator by generating realistic face images conditioned on the input sketch. The training process involves alternating between updating the generator and the discriminator.

- **Loss Functions:**
  - **Adversarial Loss:** A Binary Cross-Entropy (BCE) loss was used for the adversarial loss. The generator's objective is to minimize this loss to generate images that can fool the discriminator, while the discriminator tries to maximize this loss to correctly distinguish real from generated images.
  - **L1 Loss:** An additional L1 loss was used to ensure that the generated images were similar to the target photos, encouraging pixel-level accuracy in the generated outputs. A weighting factor of $\lambda = 100$ was used to balance the adversarial loss and L1 loss.
- **Optimizers:** Both the generator and the discriminator were optimized using Adam optimizers with learning rates of 0.0002 and 0.00005, respectively. The Adam optimizer was chosen for its ability to handle sparse gradients efficiently.

5) Mixed Precision Training: To improve training efficiency, mixed precision training was applied using Automatic Mixed Precision (AMP). This allowed for faster training by utilizing lower precision (16-bit floating point) where possible, without a significant loss in model accuracy.

6) Training Procedure: The training loop involved alternating between the generator and discriminator updates:

- **Discriminator Update:** The discriminator was trained to distinguish real photos from the generator's output (fake images).
- **Generator Update:** The generator was updated to minimize both the adversarial loss (fooling the discriminator) and the L1 loss (similarity to the real photo).
- **Checkpointing:** Model weights were saved after each epoch to resume training if interrupted and for evaluation purposes.
- **Validation:** The model's performance was evaluated after each epoch on the validation set using the average L1 loss and the generator's ability to produce realistic images.

D. CycleGAN Implementation for Person Face Sketches

1) Dataset: The dataset used in this project is the Person Face Sketches dataset available on Kaggle. It consists of paired images of human sketches and corresponding real photos. The dataset is divided into three parts:

- **Training Set:** Contains 20.7k images each in the sketches and photos subfolders.
- **Validation Set:** Contains 1000 images each in the sketches and photos subfolders for model validation.
- **Test Set:** Contains 679 images each in the sketches and photos subfolders for testing the model's performance.

Each sketch in the dataset is grayscale, while the corresponding photo is in RGB format. The dataset is designed for the task of conditional image generation, where the model learns to map a sketch to its corresponding real photo.

2) Pre-processing: Before feeding the images into the model, they undergo a series of preprocessing steps:

- **Resizing:** All images are resized to 128x128 pixels to maintain consistency in input size.
- **Normalization:** Images are normalized to the range of [-1, 1] by converting the image to a PyTorch tensor and normalizing the pixel values with a mean of 0.5 and a standard deviation of 0.5 for all channels.

3) Model Architecture: The CycleGAN architecture used in this project consists of two primary components: generators and discriminators.

- **Generator Network (U-Net Architecture):** Two U-Net-based generators are used:
  - netG-A2B: Converts real face images to sketches (RGB to grayscale).
  - netG-B2A: Converts sketches to real face images (grayscale to RGB).

  Each generator follows a U-Net architecture with encoder and decoder blocks:

  - Encoder: Composed of convolutional layers followed by instance normalization and LeakyReLU activation.
  - Decoder: Consists of transposed convolution layers, instance normalization, and ReLU activation, with a final tanh activation for output image generation.
- **Discriminator Network (PatchGAN Architecture):** Two PatchGAN discriminators are used:
  - netD-A: Evaluates real face images (RGB) and generated face images.
  - netD-B: Evaluates real sketches (grayscale) and generated sketches.

  The discriminators consist of convolutional layers with instance normalization and LeakyReLU activation, culminating in a single scalar output representing whether the input is real or fake.

4) Cycle Consistency Mechanism: CycleGAN employs the cycle consistency loss to ensure that when an image is translated from one domain to another and back, the original image is reconstructed. This is achieved by minimizing two types of losses:

- Cycle loss: Ensures that when an image is transformed from one domain to the other and back, it closely resembles the original image.
- Identity loss: Enforces that the generators learn to preserve identity when an image from the target domain is inputted to the corresponding generator.

5) Training Procedure: The CycleGAN was trained for 100 epochs using both the generators and discriminators. The training loop involves the following steps:

- Adversarial Loss: Both the generators and discriminators are trained in an adversarial manner. The generator tries to minimize the adversarial loss by fooling the discriminator, while the discriminator tries to correctly classify real and fake images.
  - Generator Loss: Consists of the adversarial loss, cycle consistency loss, and identity loss.
  - Discriminator Loss: Measures how well the discriminator distinguishes between real and generated images.
- Mixed Precision Training: To accelerate training, mixed precision was enabled using PyTorch's autocast() and GradScaler(), allowing faster computation without sacrificing accuracy on the A100 GPU.
- Checkpoints and Evaluation: After each epoch, the model saves the generator and discriminator states, along with the optimizer states, for later evaluation or resumption of training. During training, the models are evaluated on the validation set, and images are displayed to visualize the progress.
- Loss Tracking: Generator and discriminator losses are logged and plotted over time to track convergence. Losses for each epoch are saved.

6) Optimizers: The Adam optimizer is used to train both the generators and discriminators with the following hyperparameters:

- Learning rate (lr): 0.0004.
- Betas:(0.5, 0.999) to stabilize training and prevent mode collapse.

Each optimizer is responsible for updating the parameters of the respective networks.

III. Results

A. Generative Modeling of Signatures Using Variational Autoencoders and Generative Adversarial Networks with Augmentation Techniques

1) Training and Validation Loss: During training, the Variational Autoencoder (VAE) model's KL loss and reconstruction loss were monitored across 100 epochs. At around epoch 25, the reconstruction loss reached a value of 1698.09 and continued decreasing until epoch 55, where the reconstruction loss reached 1314.64. The VAE model successfully minimized the reconstruction loss, demonstrating its ability to generate realistic signatures from the latent space. Specifically, at the best epoch (101), the reconstruction loss was 1127.61 with a KL loss of 2501.45, showcasing the model's effectiveness in learning meaningful latent representations for generating signatures.

Similarly, the Generative Adversarial Network (GAN) model's generator and discriminator loss were tracked across 300 epochs. Early in the training, at epoch 36, the discriminator loss $(d-loss)$ was 0.5959, while the generator loss $(g-loss)$ was 1.8270. By epoch 40, the GAN exhibited improvement with a $g-loss$ of 2.0594. However, during epoch 45, there was a sharp increase in $d-loss$ to 8.7582 and a decrease in $g-loss$ to 0.7628, indicating instability in the training process. The training was restored to the best model weights at epoch 44, where the GAN achieved a $d-loss$ of 0.6238 and a $g-loss$ of 1.9303, signifying the best performance before the model destabilized.

2) Generated Examples: Samples of the generated signatures were visually compared with the original dataset. Both the VAE and GAN were able to generate signatures that closely resembled the real signatures. The VAE produced consistent and smooth outputs, particularly at lower reconstruction losses (e.g., 1314.64 at epoch 55). The GAN generated sharper and more defined images early in training, but its performance deteriorated in later epochs as evidenced by the sharp increase in $d-loss$ and collapse in $g-loss$ after epoch 45.

B. Custom GAN Implementation for CIFAR-10

The GAN model was trained for 100 epochs, with notable fluctuations in the generator and discriminator losses across the training process. Below are some key results:

- **Epoch 21:** Loss-D: -1.24, Loss-G: 2.99
- **Epoch 23:** Loss-D: -1.16, Loss-G: 2.90
- **Epoch 31:** Loss-D: 0.18, Loss-G: 4.22

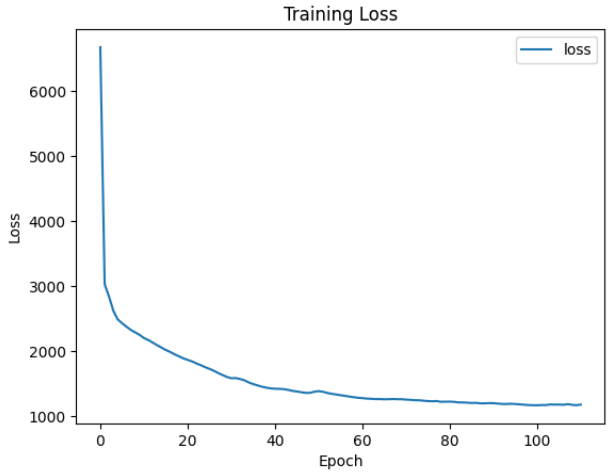

Fig. 1. Training Loss for VAE

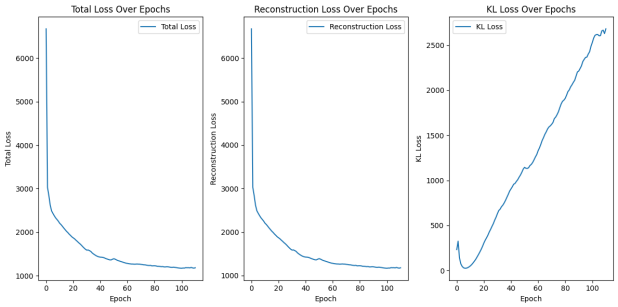

Fig. 2. Total Loss, Reconstruction Loss and KL Loss for VAE

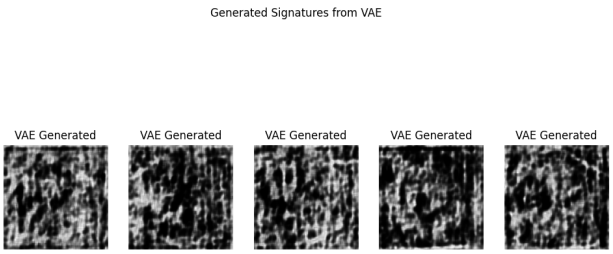

Fig. 3. Signatures generated by VAE

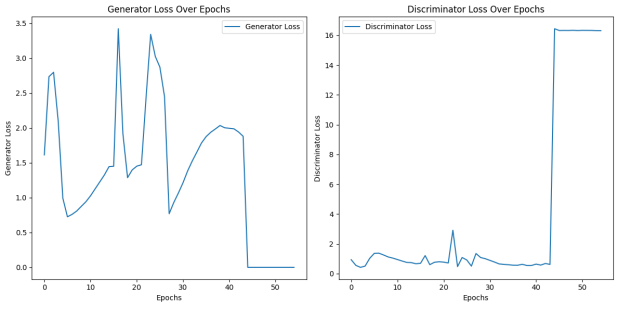

Fig. 4. Losses for GAN

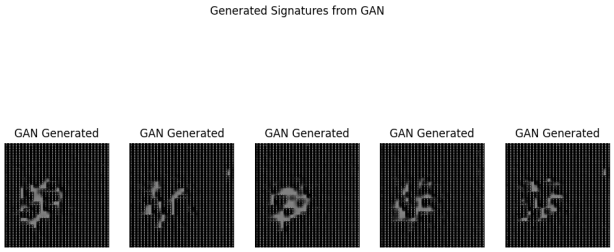

Fig. 5. Signatures generated by GAN

- **Epoch 37:** Loss-D: 0.51, Loss-G: 6.37
- **Epoch 50:** Loss-D: -1.34, Loss-G: 3.78
- **Epoch 60:** Loss-D: -1.51, Loss-G: 4.88
- **Epoch 80:** Loss-D: -1.06, Loss-G: 7.43
- **Epoch 90:** Loss-D: -1.61, Loss-G: 5.71
- **Epoch 99:** Loss-D: -1.61, Loss-G: 5.63

During early epochs, the discriminator loss was highly negative, indicating that the discriminator was confident in distinguishing between real and fake images. Over time, both the generator and discriminator improved, leading to more stable loss values. The generator produced images of increasing quality, as evidenced by higher Loss-G values over time.

Generated images at each epoch show the gradual improvement in realism, with the generator learning to produce more recognizable cats and dogs.

Despite the efforts to stabilize training, the generated images remained somewhat blurry and lacked fine detail. This could be due to several factors, such as the architecture's complexity and the limited number of training epochs, as well as potential overfitting or underfitting. Additionally, the adversarial loss and similarity score in the custom Siamese network could have introduced challenges in properly balancing feature extraction and adversarial training.

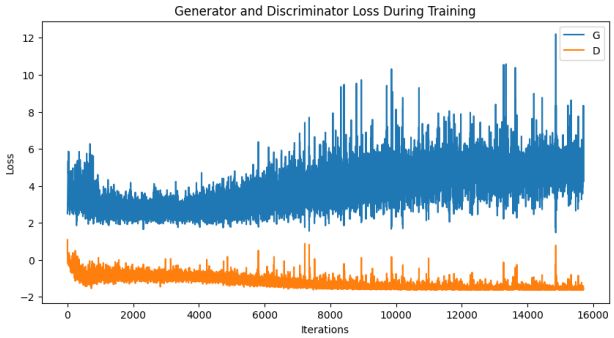

Fig. 6. Generator and Discriminator Loss

C. Conditional GAN Implementation for for Person Face Sketches

The Conditional GAN (cGAN) model was evaluated at two significant points during training: after the 79th epoch and the 100th epoch. For the 79th epoch model, the average L1

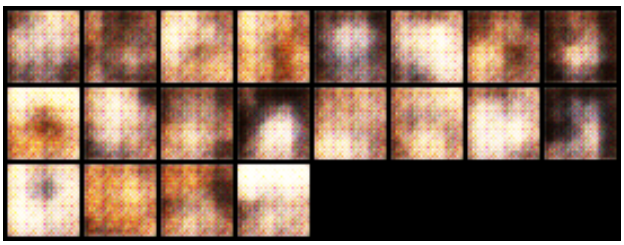

Fig. 7. Generated Images for Epoch 1

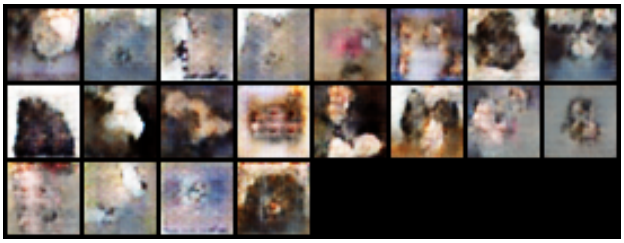

Fig. 8. Generated Images for Epoch 25

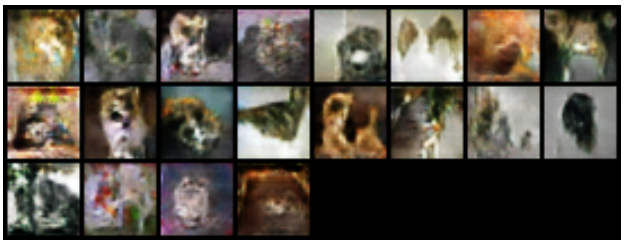

Fig. 9. Generated Images for Epoch 50

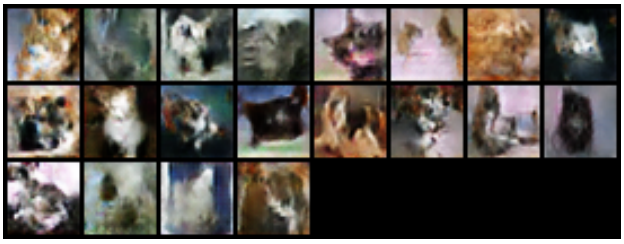

Fig. 10. Generated Images for Epoch 80

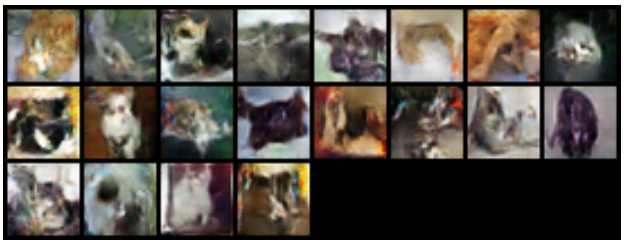

Fig. 11. Generated Images for Epoch 100

loss on the test dataset was 0.1623, while the 100th epoch model achieved a slightly better average L1 loss of 0.1579. These L1 loss values indicate that both models performed reasonably well in generating realistic face images from sketches, with the 100th epoch model demonstrating a marginal improvement in image quality and overall accuracy. However, the differences in L1 loss are minimal, suggesting that the model's performance began to plateau after the 79th epoch, implying that further training might result in diminishing returns.

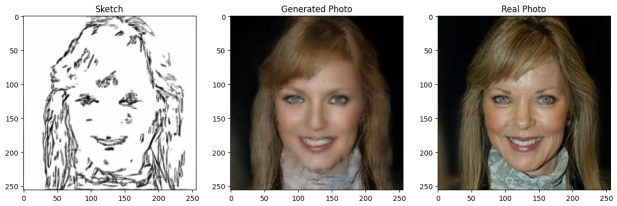

Fig. 12. Output of CGAN on Test Data after 79 epochs - Image 1

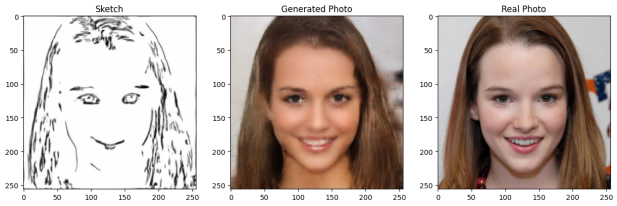

Fig. 13. Output of CGAN on Test Data after 79 epochs - Image 2

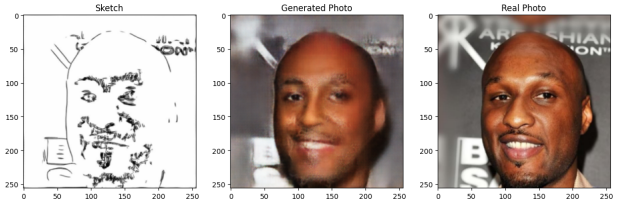

Fig. 14. Output of CGAN on Test Data after 79 epochs - Image 3

Fig. 15. Output of CGAN on Test Data after 79 epochs - Image 4

D. CycleGAN Implementation for Person Face Sketches

Throughout the training process, the discriminator (D) and generator (G) losses showed significant improvements, indicating that both models were learning effectively. In the early stages of training (Epochs 0-10), the D loss began at a

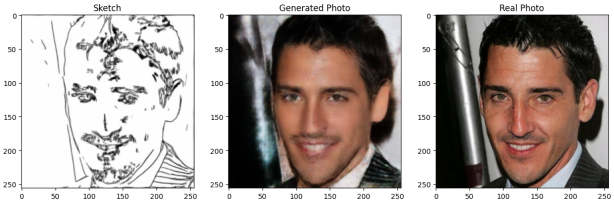

Fig. 16. Output of CGAN on Test Data after 79 epochs - Image 5

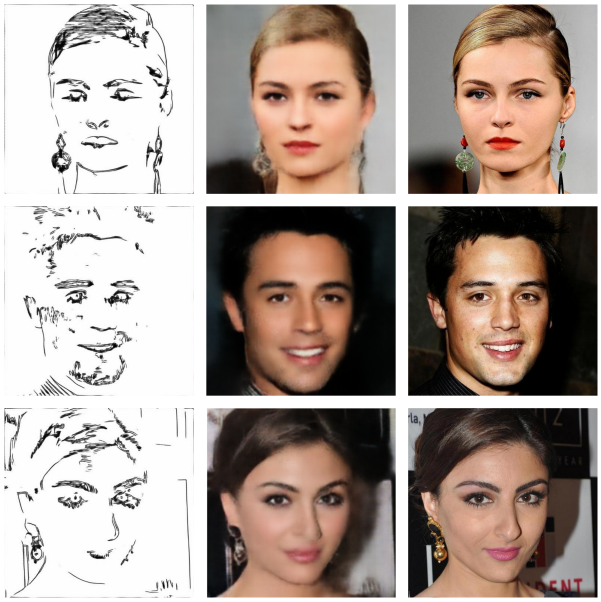

Fig. 17. Output of CGAN on Test Data after 100 epochs

high value of around 0.2635, while the G loss started at a much larger 28.65, reflecting the initial struggle of both models to compete. As training progressed into the middle phase (Epochs 40-60), the D loss had stabilized around 0.043, and the G loss reduced to approximately 5.2, indicating the generator was improving in producing realistic outputs. By the final epochs (90-100), both losses had further stabilized, with the D loss reaching a minimal value of around 0.0086, and the G loss settling around 4.9, suggesting that the models had reached a point of equilibrium where the generator was producing high-quality images that were difficult for the discriminator to differentiate from real images.

IV. Discussion

A. Generative Modeling of Signatures Using Variational Autoencoders and Generative Adversarial Networks with Augmentation Techniques

Both the Variational Autoencoder (VAE) and Generative Adversarial Network (GAN) models showed poor performance in generating realistic signatures. Despite the models being trained over multiple epochs, the generated signatures lacked clarity and consistency, often deviating from the original patterns. This could be attributed to the

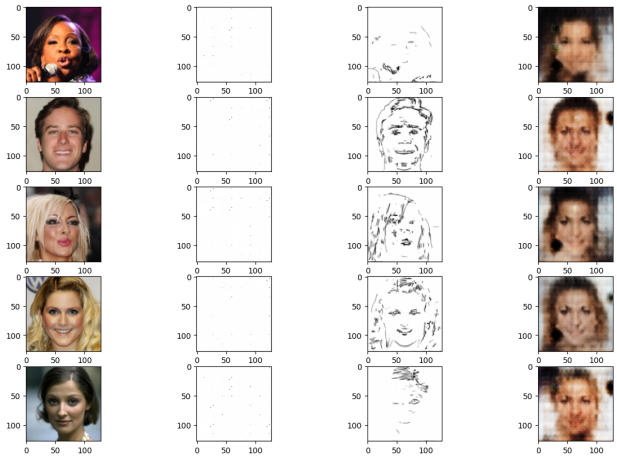

Fig. 18. Output of CycleGAN after 1 epoch

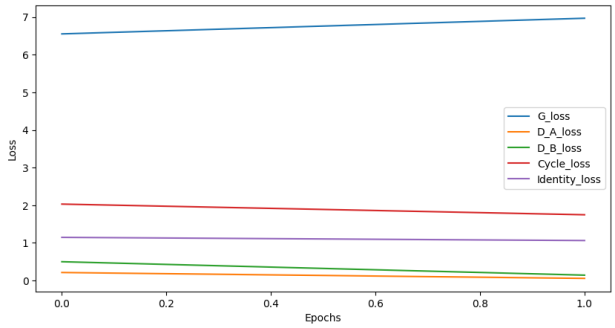

Fig. 19. Loss of CycleGAN after 1 epoch

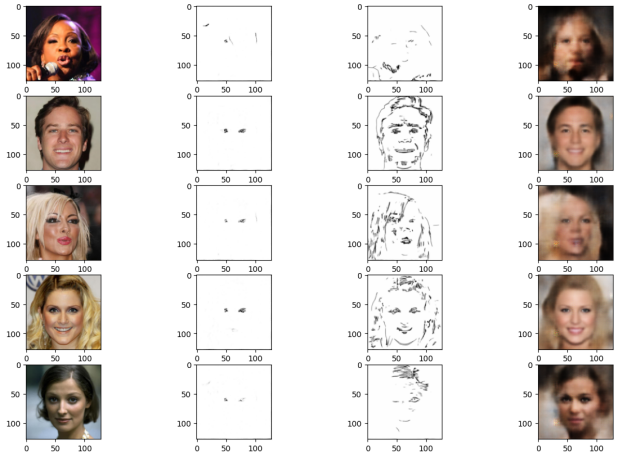

Fig. 20. Output of CycleGAN after 25 epoch

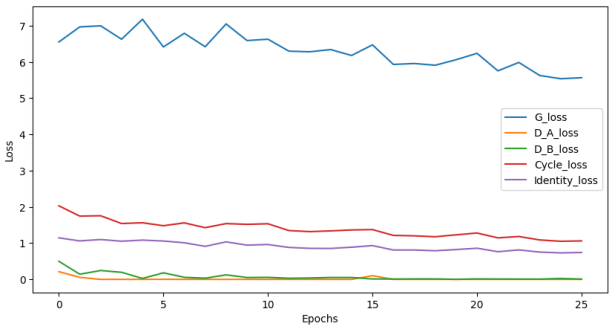

Fig. 21. Loss of CycleGAN after 25 epoch

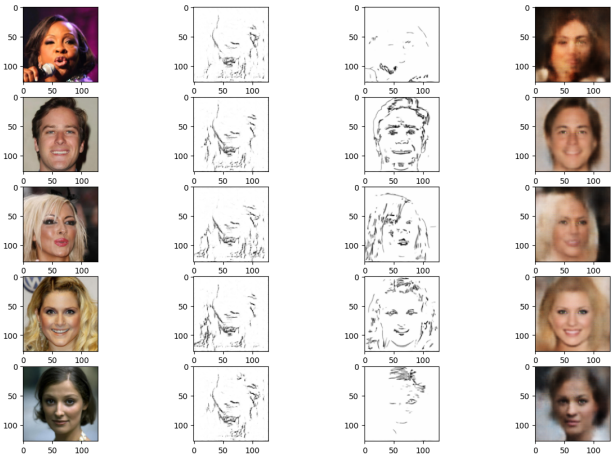

Fig. 22. Output of CycleGAN after 50 epoch

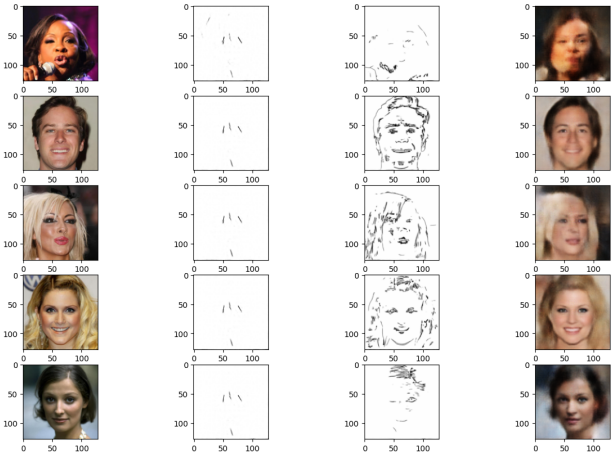

Fig. 23. Output of CycleGAN after 100 epoch

small size of the dataset, insufficient diversity in the data, and potential overfitting during training. To improve the results, future efforts could focus on increasing the dataset size through more extensive augmentation techniques, such as adding elastic deformations, affine transformations, and larger rotations. Additionally, experimenting with advanced GAN architectures like StyleGAN or improving the latent space exploration in VAE may yield better results. Furthermore, fine-tuning the model hyperparameters, such as learning rates, regularization techniques, and network depth, may contribute to more stable and realistic signature generation.

B. Custom GAN Implementation for CIFAR-10

1) Improvements:

- **Loss Stabilization:** Throughout training, the discriminator loss fluctuated, especially during the initial epochs. Techniques like feature matching or improved gradient penalty terms could potentially stabilize the training process further.
- **Siamese Network Architecture:** While the Siamese discriminator added complexity by including a similarity score, tuning the weighting between adversarial and similarity losses would likely enhance performance. Balancing these two components is critical, as the generator needs to focus both on fooling the discriminator and producing images close to the real ones.

2) Challenges:

- **Siamese Network Integration:** Incorporating a Siamese network for the discriminator introduced additional computational complexity. The shared network for real and generated images added to the training time and resource requirements. Furthermore, tuning the similarity score and its contribution to the overall loss posed a challenge during training.
- **Resource Constraints:** Due to limited computational resources, longer training times were needed, especially for high-quality image generation. Utilizing more powerful hardware, such as multiple GPUs, would significantly reduce training time and allow for more extensive hyperparameter tuning.

3) Drawbacks of Architecture:

- **Model Convergence:** Throughout training, the discriminator loss fluctuated, especially during the initial epochs. Techniques like feature matching or improved gradient penalty terms could potentially stabilize the training process further.

C. Conditional GAN Implementation for for Person Face Sketches

One of the main challenges encountered during the development of the Conditional GAN (cGAN) model was the significant computational resource requirement. Training the model with a U-Net generator and PatchGAN discriminator

proved to be computationally expensive, which made the process prone to interruptions such as laptop crashes or system restarts. These disruptions caused data loss during the training process, requiring recovery from saved checkpoints, which introduced delays and potential inconsistencies. Moreover, the model's L1 loss between epochs 79 and 100 exhibited minimal improvement, suggesting that the model might have reached a plateau, with additional training providing only marginal gains. This highlights the challenge of tuning GANs effectively, particularly in terms of determining the ideal number of epochs for training. Another potential drawback was the use of fixed learning rates and loss function weights. Although the learning rate schedulers were implemented, they might not have been fully optimized, as the performance improvements became smaller over time. The model could benefit from a more dynamic adjustment of hyperparameters to enhance its performance further.

1) Possible Improvements:

- **Early Stopping and Dynamic Learning Rate:** Implementing early stopping based on validation losses could prevent overfitting and save computational resources. The model's performance plateaued after the 79th epoch, and early stopping could have been triggered to prevent unnecessary training. Additionally, employing a more adaptive learning rate schedule might help maintain a faster improvement rate throughout training.
- **Better Resource Management:** Since the model encountered system limitations, moving to cloud-based GPU services or more powerful hardware could prevent interruptions and accelerate training. This could also help in experimenting with more complex architectures or hyperparameters.
- **Data Augmentation:** While the dataset size was relatively large, introducing data augmentation techniques could enhance the diversity of training examples, thereby improving the model's generalization abilities, especially in generating more varied face outputs from sketches.
- **Improved Generator-Discriminator Balance:** Further tuning of the loss weights between the generator and discriminator may help in preventing mode collapse and ensuring that the discriminator doesn't overpower the generator, leading to more consistent results in face generation.
- **Utilizing Other Loss Functions:** While L1 loss performed well in this task, experimenting with perceptual loss or feature matching loss could improve the visual quality of the generated images, as these losses are better at capturing high-level image features.

D. CycleGAN Implementation for Person Face Sketches

The steady reduction and stabilization of both the generator and discriminator losses demonstrate successful adversarial training. The discriminator learned to identify generated images effectively early on, as reflected by its decreasing loss values. However, as the generator improved, its loss also stabilized at lower values, signaling that it was learning to produce images closer to the real distribution. This is supported by the final G loss stabilizing at around 4.9, a low value, indicating the generator's ability to produce convincing, high-quality images. Simultaneously, the small D loss around 0.0086 highlights that the discriminator struggled to consistently distinguish real from generated images as the generator improved.

1) Possible Improvements:

- **Data Augmentation:** Incorporate data augmentation techniques (e.g., random cropping, flipping, rotation) to introduce more variability into the training set and improve the model's robustness to different sketch inputs.
- **Advanced Loss Functions:** Use more sophisticated loss functions like Wasserstein loss or Perceptual loss instead of basic binary cross-entropy and L1 loss. This can help improve the quality and realism of the generated images by making the model more sensitive to high-level feature differences.
- **Attention Mechanisms:** Add attention layers to the generator and discriminator to help the model focus on important regions of the sketch and generated image, which may lead to more accurate and realistic outputs.
- **Progressive Growing of GANs:**Implement Progressive GANs, where the model starts generating low-resolution images and gradually increases the resolution over training epochs. This can lead to better stability and sharper images.
- **Improved Discriminator:** Add multiple PatchGAN discriminators operating at different scales (multi-scale discriminators). This can make the discriminator more effective in detecting high- and low-level features, thus improving the overall performance.

V. Conclusion

A. Generative Modeling of Signatures Using Variational Autoencoders and Generative Adversarial Networks with Augmentation Techniques

In this task, both the VAE and simple GAN faced challenges in generating high-quality, realistic signatures. While the VAE was effective in learning latent space representations, the generated signatures lacked fine details and clarity. Similarly, the GAN struggled with producing sharp, convincing images, as evidenced by unstable discriminator loss after several epochs. To improve the performance, increasing the dataset size with more diverse augmentation techniques, fine-tuning the model architectures, and exploring more advanced GAN variants could lead to better quality outputs. Additionally, incorporating deeper networks or adjusting the training strategies might help the models capture more complex features of the original signatures.

B. Custom GAN Implementation for CIFAR-10

In the second question of the assignment, we successfully implemented a custom GAN with a Siamese discriminator on a filtered CIFAR-10 dataset. The model was able to generate images of cats and dogs, with gradual improvement in quality over the course of training. Despite facing challenges such as resource limitations and model instability, the use of a similarity score in the discriminator showed potential in enhancing the realism of the generated images. Future work could explore more sophisticated architectures and improved training techniques to further boost the performance and stability of the GAN.

C. Conditional GAN Implementation for for Person Face Sketches

In this Conditional GAN (cGAN) implementation task, the primary objective was to generate realistic face images from input sketches using the Person Face Sketches dataset. Throughout the project, significant progress was made in designing a robust U-Net-based generator paired with a PatchGAN discriminator, following guidelines from the original cGAN paper. The model was trained for 100 epochs, with checkpoints saved periodically to allow for recovery in case of disruptions.

However, several challenges were encountered, including interruptions due to system limitations that led to the loss of data during training, requiring restarts from saved checkpoints. Despite these interruptions, the model exhibited consistent improvements in performance, particularly up to the 79th epoch, after which the gains in loss reduction became marginal. The average L1 loss dropped from 0.1623 at the 79th epoch to 0.1579 at the 100th epoch, indicating some level of plateau in performance.this project highlighted both the potential and the limitations of cGANs in face image generation tasks. While the model achieved its primary goal of generating realistic face images from sketches, further optimizations in terms of resource management, hyperparameter tuning, and potentially more advanced loss functions could improve performance.

D. CycleGAN Implementation for Person Face Sketches

The training results indicate that the CycleGAN model successfully learned to generate high-quality images over the course of training. The final generator loss of 4.9 and discriminator loss of 0.0086 suggest a well-balanced adversarial relationship between the two models, with the generator producing images that closely resemble the real distribution. These results demonstrate the effectiveness of the adversarial training process, with both models reaching a competitive balance and providing high-quality outputs.

VI. Prompts

A. Generative Modeling of Signatures Using Variational Autoencoders and Generative Adversarial Networks with Augmentation Techniques

1) VAE and GAN Implementation:

- Discussion on implementing VAE and GAN to generate synthetic signatures.
- Preprocessing steps included data augmentation techniques such as scaling, rotation, and noise addition to increase dataset diversity.
- VAE and GAN training was conducted, but the models performed poorly in generating realistic signatures.
- Key insights included the importance of dataset size and model architecture in improving performance.

2) Training Results and Issues:

- VAE training monitored KL loss and reconstruction loss over 100 epochs.
- GAN training tracked generator and discriminator loss over 300 epochs.
- GAN model experienced instability, with discriminator loss fluctuating and failing to generate meaningful outputs in later epochs.
- Early stopping was used in both models, but the GAN generator produced very poor outputs in final epochs.

3) Possible Improvements:

- Suggested improvements for GAN include using advanced architectures like StyleGAN and fine-tuning hyperparameters (learning rates, batch sizes).
- VAE could benefit from improvements in latent space exploration and dataset size.
- Increasing diversity in the signature dataset with more advanced augmentation techniques (elastic deformations, affine transformations).

4) Evaluation and Visualization:

- Metrics such as reconstruction loss for VAE and generator loss for GAN were discussed and visualized during training.
- Samples of generated signatures were compared visually to the original dataset but showed limited success in realism.

5) LaTeX and Report Writing Assistance:

- Provided LaTeX code to structure the report based on the IEEE conference paper template.
- Assistance with writing abstract, introduction, and methodology for the report.
- Helped with references, including links to datasets (Kaggle Person Face Sketches, CIFAR-10, and Google Drive folder).
- Advised on linking the chat conversation in the report using LaTeX's `\href` command.

For more details on the chat, visit the following link: Chat Link

B. Custom GAN Implementation for CIFAR-10

1) Dataset Preparation and Preprocessing:

- Dataset: The CIFAR-10 dataset was used, with a focus on filtering and using only the "cats" and "dogs" classes (labels 3 and 5).

- Preprocessing: Images were transformed using normalization techniques, scaling them to a range of [-1, 1] to help with stable GAN training.
- DataLoader: A custom `DataLoader` was created to load the filtered dataset into batches for training.

2) Model Architecture:

- Generator (DCGAN-based): The generator used a DCGAN-style architecture with transposed convolution layers, batch normalization, and ReLU activation to generate 64x64 RGB images. The output layer used a `tanh()` activation function to rescale the pixel values to the range of [-1, 1].
- Custom Discriminator (Siamese Network): A custom discriminator was created, which takes both real and generated images as input to compute a similarity score between them. The architecture involved a shared convolutional backbone, followed by two fully connected heads: one for adversarial classification and the other for the similarity score.

3) Training Strategy:

- Loss Functions: A combination of adversarial loss (binary cross-entropy for real vs. fake classification) and a similarity-based loss was used. A combination of adversarial loss (binary cross-entropy for real vs. fake classification) and a similarity-based loss was used.
- Tracking Losses: The losses for both the generator and discriminator were tracked and saved throughout training for later visualization.

4) Challenges Faced:

- Blurry Images: Despite using a custom GAN architecture, the generated images remained blurry and lacked recognizable features even after 100 epochs.
- Siamese Network Complexity: The incorporation of a Siamese network for similarity scoring added complexity, making it harder to stabilize training.
- Limited Resources: The quality of the generated images could have been improved with more computational resources and longer training times.

5) Results and Visualizations:

- Training Loss: The generator and discriminator losses were saved after each batch. Graphs of these losses were plotted to visualize training dynamics.
- Generated Images: Images generated at the end of each epoch were saved and used for evaluation. Although there was some improvement over time, the images were still not highly recognizable.

6) Improvements and Future Work:

- Architecture Adjustments: Future improvements could include experimenting with different architectures, such as using residual blocks in the generator or refining the discriminator's similarity mechanism.
- Hyperparameter Tuning: Further optimization of hyperparameters such as the learning rate, batch size, and loss weights (for balancing similarity vs. adversarial loss) might yield better results.

C. Conditional GAN Implementation for CIFAR-10

1) Dataset and Preprocessing: The Person Face Sketches dataset was used, containing over 20,000 images for training and validation, split between the "sketches" and "photos" subfolders. For preprocessing, both sketches and photos were resized to 256x256, normalized between [-1, 1], and converted to grayscale for sketches and RGB for photos. DataLoader was used to load the images into batches.
2) Model Architecture: The model architecture followed the original Conditional GAN (cGAN) paper. The generator was built using a U-Net structure with skip connections, consisting of multiple down-sampling and up-sampling blocks. The discriminator was based on PatchGAN, designed to classify image patches as real or fake. Both models were optimized using Adam optimizers with learning rate schedulers to reduce learning rates based on validation loss.
3) Training and Challenges: The model was trained for 100 epochs, saving checkpoints periodically at epochs 1, 37, 79, and 100. Mixed precision training was employed to improve training efficiency using autocast and GradScaler. Despite a well-defined training loop, challenges included system crashes due to computational limitations, causing loss of data and requiring restarts from saved checkpoints. Balancing the learning process between the generator and discriminator was another challenge.
4) Results: At the 79th epoch, the model achieved an average L1 loss of 0.1623 on the test dataset, improving to 0.1579 by the 100th epoch. The model consistently reduced loss over time, but improvements slowed after the 79th epoch, indicating diminishing returns in training performance.
5) Discussion: The main drawbacks were the need for computational resources and system crashes, which affected training continuity. The challenges of balancing the generator and discriminator and achieving efficient training with large datasets and high-resolution images were central. Improvements include using more powerful hardware, exploring advanced loss functions, and improving the stability of training pipelines.
6) Conclusion: The project successfully demonstrated the ability of cGANs to generate realistic face images from sketches. Despite system interruptions and the slow convergence in later epochs, the model produced high-quality outputs. Future work should focus on optimizing resource management and refining the model to further enhance performance and reliability.
For more details on the chat, visit the following link: Chat Link

D. CycleGAN Implementation for Person Face Sketches

1) CycleGAN Architecture : The architecture used U-Net generators and PatchGAN discriminators, tailored for face-to-sketch and sketch-to-face conversions. Both models were trained adversarially, with discriminators distinguishing real from generated images and generators learning to produce convincing sketches and face images.

2) Training Process and Loss Function : The training loop utilized mixed precision to speed up computations, with gradient scaling to maintain stability. The training combined adversarial, cycle-consistency, and identity losses, which are essential to preserving image details and style during the conversion.

3) Challenges and Errors : The training loop utilized mixed precision to speed up computations, with gradient scaling to maintain stability. The training combined adversarial, cycle-consistency, and identity losses, which are essential to preserving image details and style during the conversion.

- Loading Pretrained Models: Initially, there was confusion about how to load saved checkpoints (e.g., netG-A2B-epoch-39.pth) and resume training, which was resolved by modifying the code to load specific epoch checkpoints.
- Stabilizing Losses: The generator and discriminator losses fluctuated early in training, which is typical, but later stabilized, showing successful adversarial competition between the models.
- Handling Long Training Time: Mixed precision training with an A100 GPU was employed to improve training speed and manage large datasets.
- User Interface Development: Implementing a UI for live camera input and image upload was another area of focus. The UI allowed users to upload or capture an image and receive a converted output (either face-to-sketch or sketch-to-face). Errors like UI layout and display issues were addressed, and visual improvements were implemented.

For more details on the chat, visit the following link: Chat Link

REFERENCES

[1] Imama Amjad, "Shared Folder for Signatures Dataset," [Online]. Available: https://drive.google.com/drive/u/0/folders/1XZ5Vz5oDimChKIAAaNZ4nAN2coj6g0yg. [Accessed: 18-Oct-2024].

[2] Kaggle, "Person Face Sketches Dataset," [Online]. Available: https://www.kaggle.com/datasets/almightyj/person-face-sketches. [Accessed: 18-Oct-2024].

[3] A. Krizhevsky, "CIFAR-10 (Canadian Institute for Advanced Research) Dataset," [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 18-Oct-2024].

[4] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967-5976, 2017. doi: 10.1109/CVPR.2017.632.

[5] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784.