Conditional GAN Implementation for Person Face Sketches

Abdul Manan
Department of Computer Science
FAST NUCES
Email: i210641@nu.edu.pk

***Abstract*—In this project, we implement a Conditional GAN (cGAN) to generate realistic face images from input sketches using the Person Face Sketches dataset. The cGAN model consists of two primary components: the Generator and the Discriminator. The Generator aims to translate a sketch into a corresponding realistic face, while the Discriminator attempts to distinguish between real and generated face images. Both networks are trained in an adversarial manner, guided by the architecture proposed in the original Conditional GAN paper and the pix2pix paper [1].**

I. Introduction

Generative Adversarial Networks (GANs) are a powerful class of machine learning models that can generate realistic images by competing two neural networks: the generator and the discriminator. In this project, we utilize the Person Face Sketches dataset to train a GAN where we generate faces based on sketches.

II. Dataset

The dataset comprises paired sketches and real photos of human faces. Each sketch has a corresponding real photo, and our task is to generate the real photo conditioned on the sketch input.

- Training Dataset: Used to train the cGAN model.
- Validation Dataset: Utilized to monitor the performance during training and prevent overfitting.
- Test Dataset: Used for evaluating the final performance of the model.

III. Model Architecture

The generator and custom discriminator networks are described below:

A. Generator Architecture

The Generator is designed as a U-Net architecture, where the input is a grayscale sketch (shape: 256x256x1), and the output is a color face image (shape: 256x256x3). The network is composed of several downsampling and upsampling layers. Skip connections are employed to pass the output of each downsampling layer to the corresponding upsampling layer, allowing the model to retain high-frequency details from the sketch input.

- **Input:** A grayscale sketch of size $256 \times 256 \times 1$.
- **Downsampling:** The generator consists of 8 downsampling layers, each using a convolutional layer with stride 2, followed by batch normalization (except for the first layer) and LeakyReLU activation. Each downsampling step reduces the spatial dimension of the input by half.
  - Layer 1: Conv2D$(64, 4, 2)$ $\rightarrow$ LeakyReLU
  - Layer 2: Conv2D$(128, 4, 2)$ $\rightarrow$ BatchNorm $\rightarrow$ LeakyReLU
  - Layer 3: Conv2D$(256, 4, 2)$ $\rightarrow$ BatchNorm $\rightarrow$ LeakyReLU
  - Layer 4-8: Repeat similar structure, increasing filters to 512
- **Upsampling:** After reaching the bottleneck, the network starts upsampling using transposed convolutional layers. Each upsampling step doubles the spatial dimensions and reduces the filter count, applying ReLU activation and batch normalization. Dropout is applied to the first three layers of upsampling to prevent overfitting.
  - Layer 1: Conv2DTranspose$(512, 4, 2)$ $\rightarrow$ BatchNorm $\rightarrow$ Dropout $\rightarrow$ ReLU
  - Layer 2: Conv2DTranspose$(512, 4, 2)$ $\rightarrow$ BatchNorm $\rightarrow$ Dropout $\rightarrow$ ReLU
  - Layer 3: Conv2DTranspose$(256, 4, 2)$ $\rightarrow$ BatchNorm $\rightarrow$ ReLU
  - Layer 4-7: Continue upsampling, reducing filters to 64
- **Output:** The final layer applies a transposed convolution with 3 filters (to produce an RGB image) and a $\tanh$ activation function, outputting an image of size $256 \times 256 \times 3$.

The skip connections between each downsampling and upsampling layer are concatenated, ensuring that the model can preserve essential details during the transformation process.

B. Discriminator Architecture

The Discriminator in the cGAN is a PatchGAN classifier, which classifies each 70×70 patch in the image as real or fake. It receives both the input sketch and the real/generated image as inputs, and determines whether the image is a plausible translation of the sketch.

- **Input:** A concatenated tensor consisting of a grayscale sketch ($256 \times 256 \times 1$) and either a real or generated image ($256 \times 256 \times 3$).
- **Downsampling:** The Discriminator consists of four downsampling layers, reducing the spatial dimensions by half with each step.

  - Layer 1: Conv2D$(64, 4, 2) \rightarrow$ LeakyReLU
  - Layer 2: Conv2D$(128, 4, 2) \rightarrow$ BatchNorm $\rightarrow$ LeakyReLU
  - Layer 3: Conv2D$(256, 4, 2) \rightarrow$ BatchNorm $\rightarrow$ LeakyReLU
- **Final Layer:** After downsampling, the Discriminator applies a convolution with a single filter to predict the authenticity of each patch. The final output is a $30 \times 30 \times 1$ patch matrix, where each value indicates the probability of the patch being real.

C. Loss Functions

The model is trained using two loss functions: the **Generator Loss** and the **Discriminator Loss**.

- **Generator Loss:** The generator aims to fool the discriminator into classifying the generated image as real. The total generator loss is composed of the GAN loss (measuring how well the generator deceives the discriminator) and the L1 loss (measuring the pixel-wise difference between the generated image and the target image). This encourages the generator to produce images that are both visually realistic and semantically accurate.

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{GAN}} + \lambda \cdot \mathcal{L}_1$$

  where $\mathcal{L}_1$ is the mean absolute error between the generated image and the target image, and $\lambda = 100$ controls the relative importance of this term.
- **Discriminator Loss:** The discriminator loss is a sum of the loss for real images and the loss for generated images. The discriminator tries to correctly classify real images as real and generated images as fake.

$$\mathcal{L}_{\text{disc}} = \mathcal{L}(\text{real\_output}, 1) + \mathcal{L}(\text{generated\_output}, 0)$$

  where $\mathcal{L}$ is the binary cross-entropy loss.

D. Training Procedure

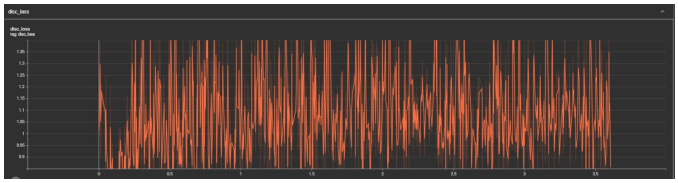
The training loop is designed to alternate between updating the Generator and Discriminator. For each batch:

- The generator produces a fake image conditioned on a sketch.
- The discriminator classifies the real image and generated image.
- The generator is updated based on how well it deceived the discriminator and how accurately it recreated the real image (using the L1 loss).
- The discriminator is updated based on its ability to distinguish real images from fake ones.

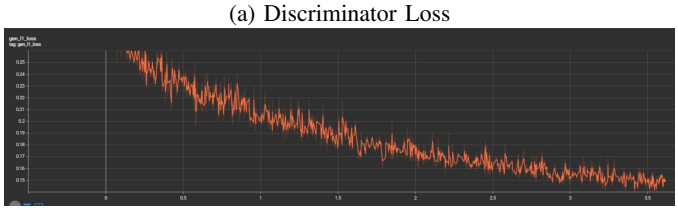
The models are optimized using the Adam optimizer with a learning rate of 2×10^{-4} and $\beta_1 = 0.5$.

IV. Results

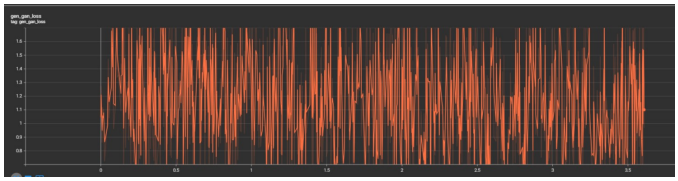
After training the model, the generator was able to produce realistic face images based on the input sketches. These generated images were visually coherent with the ground truth images, capturing important details such as facial features and overall structure. The Fig. **??**shows an example of the input sketch, the ground truth image, and the image generated by our Conditional GAN model.



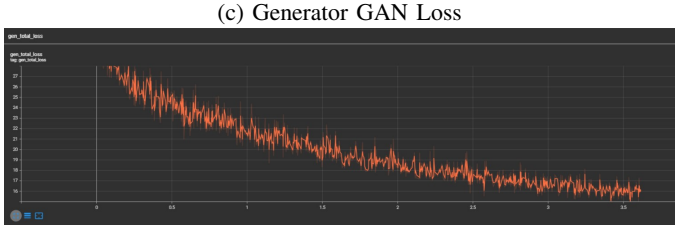
(a) Discriminator Loss



(b) Generator L1 loss



(c) Generator GAN Loss



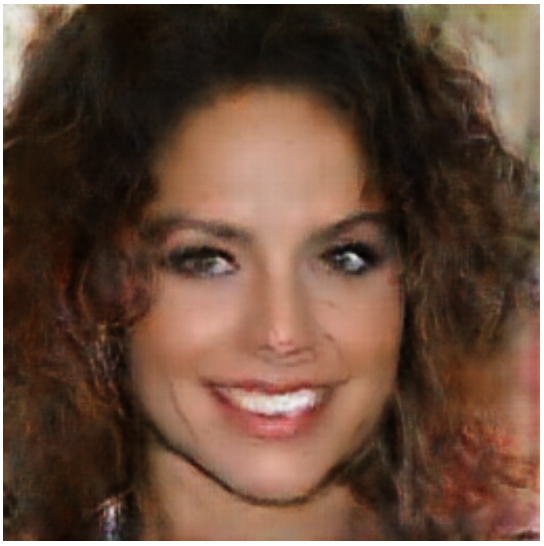
(d) Generator Total Loss

V. Generated Images Over Epochs

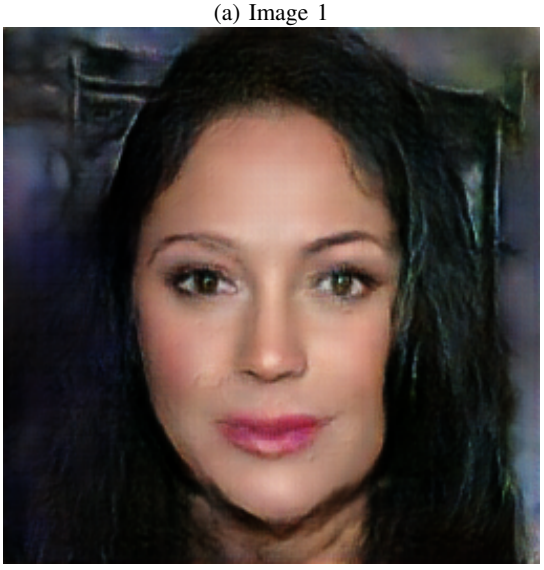
In this section, we present images generated by the GAN at different stages of training (over epochs). The progression shows how the quality of the generated images improves as the training continues.

VI. Conclusion

In this project, we successfully implemented a Conditional GAN for generating realistic face images from input sketches using the Person Face Sketches dataset. The architecture, based on U-Net for the generator and PatchGAN for the discriminator, effectively captured the mapping between sketches and real images. Future improvements could include exploring more advanced loss functions or introducing additional regularization techniques.



(a) Image 1



(b) Image 2

(c) Image 3

Fig. 2: Images generated by GAN at various epochs.

REFERENCES

[1]P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, Image-to-Image Translation with Conditional Adversarial Networks. 2018. [Online]. Available: https://arxiv.org/abs/1611.07004