ATML Report

Claudio Ricci *riccicl@usi.ch*
Damiano Ficara *ficard@usi.ch*
Stefano Andreotti *andres@usi.ch*
Stefano Quaggio *quaggs@usi.ch*

Abstract

Implementation of Efficient Training for Efficient GANs in Image-to-Image Translation. Modify images by applying various concepts such as the Van Gogh style, changing hair color to blonde, and more. The motivations behind this project are to create a flexible GAN for lightweight image modifications, to train a model without the need to retrain the entire model, and to develop a model that performs faster than a diffusion model while maintaining the same quality.

1 Introduction

The paper *E2GAN: Efficient Training of Efficient GANs for Image-to-Image Translation* Gong et al. (2024) focuses on an efficient training and deployment techniques for Generative Adversarial Networks (GANs) in image-to-image translation tasks. E2GAN is positioned as a significant advancement in the context of enabling real-time, on-device image editing. It addresses the inefficiency of diffusion models, which are computationally intensive, and the prohibitive cost of training GANs for new concepts from scratch.
E2GAN achieves real-time performance on mobile devices while maintaining high-quality image editing results, bridging the gap between the computational demands of diffusion models and the efficiency of GANs.

2 Scope of reproducibility

The work addresses the problem of making GANs more efficient for image-to-image translation tasks. Specifically, it focuses on reducing the computational cost and storage requirements associated with training GANs for various image editing concepts, while ensuring the models are suitable for real-time, on-device applications. Unlike conventional GAN training, which starts from scratch and incurs high costs, the proposed approach fine-tunes a pre-trained base GAN model using methods that significantly lower the training requirements without sacrificing image quality. The claims we are testing are:

1. **Efficient Transfer Learning with Mixed Datasets:** Training a base GAN model on a mixed dataset of diverse concepts enables efficient transfer learning to new tasks, eliminating the need to train separate models for each concept from scratch.
2. **Low-Rank Adaptation for Fine-Tuning:** Fine-tuning only 1% of the parameters in a pre-trained base GAN model using Low-Rank Adaptation (LoRA) achieves high-quality image editing results comparable to or better than conventional GAN training approaches.
3. **Data Efficiency in Fine-Tuning:** The training data required for fine-tuning can be significantly reduced by selecting a small subset of representative samples using clustering techniques, while still maintaining high performance in terms of fidelity and quality.
4. **Improved FID:** E2GAN achieves better FID (Fréchet Inception Distance) scores than baseline methods (e.g. pix2pix and pix2pix-zero-distilled), while reducing training time.

5. **Low latency on mobiles:** The proposed model architecture achieves real-time image editing on mobile devices (e.g. iPhone 14) with lower latency and computational cost compared to both diffusion models and other GAN-based methods.

3 Methodology

Our reproduction approach focused on the reimplementation of E2GAN based exclusively on the description provided in the paper, as the official code was not available. We developed the implementation using PyTorch, using Kaggle's GPU T4. To generate the training data, we used Hugging Face's InstructPix2Pix model Brooks et al. (2022). The original paper used an NVIDIA H100 GPU with 80GB of memory, forcing us to significantly adapt the architecture and training procedures to work with the more limited computational resources of the T4 GPU. For areas where the paper provided limited technical details, such as discriminator architecture and specific LoRA configurations, we made implementation decisions based on common practices in the GAN literature and empirical test results.

3.1 Model descriptions

Relative to the model, we tried to extract as much information as possible about the structure described in the paper. The idea behind the model is a GAN. In Appendix A Figure 1 you can see all the components and their order.

3.1.1 Generator

The generator exploits a combination of a CLIP (Contrastive Language-Image Pretraining) based encoder Radford et al. (2021) and a ResNet50 subsample ResNet50. In detail we have:

- **Downsampling and ResNet50:** This first part of layers allows to reduce the spatial dimensions of images and extract mid- and low-level hierarchical representations of them (such as textures, shapes, edges, etc.).
  - Initial convolution (`conv1`) and `maxpool` for dimensional reduction.
  - Two ResNet blocks (`layer1` and `layer2`) to extract important visual features. Wanting to optimize the structure as much as possible, instead of creating the ResNet from scratch (as we initially tried to do) we used a pretrained model such as ResNet50 from the `torchvision.models` library, importing the weights from `ResNet50_Weights.IMAGENET1K_V2`. Wanting to maintain ResNet's ability to extract important features, we freeze training the weights from this initial block
- **Text-Image Fusion:** The merging of textual and visual representations in the model occurs to integrate the semantic context of the textual prompt with the visual features extracted from the image. At this stage we are helped by the use of another pretrained model which allows us to have an embedding representation of our prompt passed in as input. Specifically, from the `transformers` library, we import both `CLIPTextModel` and `CLIPTokenizer`, both components from the CLIP framework, which allowed us to tokenize and create the embeddings without having to tow a separate model to capture all the semantic part of the concepts.
- **Transformer:** The next block consists of a transformer designed to combine image and prompt, analyzing the global relationships between the visual features extracted from the image and the semantic embedding generated by the textual prompt. As in the paper, additional downsampling is performed prior to this stage. Initially, we integrated the CLIP pretrained transformer, known for its ability to learn visual concepts from supervision via natural language. Although the results were promising, the number of parameters and the size of the model increased exponentially. Therefore, for the final version we opted to use a standard transformer from the `torch` library, which is a good compromise between model size and output quality.

Main parameters: Number of head-attention = 8, Number of Encoder Layers = 3, Number of Decoder Layers = 3, Feedforward Size = 1024, Activation = GELU, Dropout = 0.1, Batch First = True to have the order *(batch_size, sequence_length, feature_dim)* output.

- **Post-Transformer:** The post-transformer part is needed as preparation for the final output. It presents an upsampling layer to return to the same dimensionality obtained before downsampling for the transformer and another ResNet block to stabilize and improve the quality of the representations. Finally, we apply upsampling to restore image resolution.

3.1.2 Discriminator

The paper with respect to this part did not provide information about the implementation of the discriminator, both in terms of structure and with respect to hyperparameters. It was therefore decided to retain the discriminator from the basic GAN, adjusting it in terms of input dimensions. The specific choices regarding the structure are as follows: the `LeakyReLU` activations introduce nonlinearities to capture complex patterns, the `InstanceNorm2D` normalizations stabilize the training process by reducing the dependence on batch size, and the number of convolutional layers comes from a series of various tests carried out to identify a configuration that is not too complex but at the same time would allow a good discriminative degree to be maintained.

3.1.3 Training

The training function allows us to train our GAN, so that generator and discriminator enter into favorable competition for their improvement simultaneously. In fact, the goal is to obtain a generator that produces realistic images consistent with the textual prompt and at the same time a discriminator that is able to distinguish real and generated images. In more detail we have the loss functions `MSELoss` (mean square error), which evaluates the distance between the score assigned by the discriminator and the target values, and pixel-wise loss via `L1Loss` (mean absolute error), which measures the similarity between the generated image and the real one at the pixel level. After each epoch, a validation step is performed to monitor the quality of the model using the validation set.

3.1.4 K-Means

To identify a small subset of data, necessary for fine-tuning, we conducted unsupervised learning to analyze the structure of the dataset. Again using the CLIP model, we extract embeddings from the images, which are then clustered using the K-Means algorithm Ikotun et al. (2023) to select a smaller, meaningful representation for each concept. The parameters used are those recommended in the paper, especially the number of clusters 400. This technique allows obtaining the most representative data for a given concept, which will later be entered through LoRA, reducing training time.

3.1.5 LoRA

We implemented LoRA as described in the paper, integrating it into two downsampling blocks, the transformer, and two upsampling blocks. Specifically, we assigned different ranks of LoRA at various stages: a rank of 1 in the first downsampling block, a rank of 4 in the second downsampling block, a rank of 1 in the transformer, and a rank of 8 in the two upsampling blocks. To ensure efficient implementation, we used the *peft* package Xu et al. (2023) in Python. However, implementing the structure as described in the paper did not yield results comparable to the authors' findings. This is likely due to our model's smaller size, which caused LoRA to dominate the base model's original knowledge, leading to poor performance, see Appendix A Figures 4 and 5. We also experimented with an alternative approach, replacing the transformer with a custom multi-head attention mechanism to better integrate the algorithm. This method showed that the concept is reproducible: LoRA can effectively teach new concepts without completely overwriting the model's initial knowledge as shown in Appendix A Figure 6 that shows Van Gogh style and sculpture style. While results on existing knowledge were weaker than the base model, it was evident that the model retained its understanding of prior concepts.

3.2 Datasets

The dataset we used is Flickr-Faces-HQ (FFHQ) FFH (2022), a high-quality image dataset of human faces originally created as a benchmark for GANs, containing 70,000 images. We selected this dataset because it is one of the two datasets used by the authors of E2GAN (the other being Flickr-Scenery). We selected 1,000 images and passed them through the `Instruct-Pix2Pix` Brooks et al. (2022) diffusion model to create a modified version of each image based on specific prompts. In our implementation, we created a custom `TextGuidedImageDataset`, which links the following elements for each example: the original image, the corresponding modified image and the prompt used by the diffusion model to generate the modification. The prompt used by us are 5: *convert to albino style*, *make the hair blonde*, *apply van gogh style*, *make the person old* and *make the hair pink*. Examples illustrating an original image and the outputs for each prompt are provided in the Appendix Figure 2. The final dataset comprises 5,000 images, which were split into training, validation, and test subsets with an 80-10-10 ratio. Classes, are evenly distributed. Dataset is available at dataset url.

3.3 Hyperparameters

To search for hyperparameters, we initially relied on the fifth section of the paper (Experiments), which suggests a set of basic parameters. As a result, a trial-and-error empirical approach had to be taken to adapt the parameters to our computational resources. The suggested learning rate of $2e-4$ was changed to $3e-4$ for the generator and $1e-4$ for the discriminator, after observing instability in training with the original values. To better manage the learning rate decay, we implemented a `CosineAnnealingLR` scheduler, not present in the original paper, which allowed a more stable convergence by gradually reducing the learning rate to a minimum of $1e-6$. The `AdamW` optimizer was chosen instead of the traditional Adam suggested in the paper to take advantage of the weight decay of $1e-1$, which helped to limit overfitting. The optimizer betas (0.5, 0.999) were kept at the standard values in the GAN literature. A crucial aspect was the balancing of the loss functions. The weight of pixel loss was increased by a factor of 20 compared to GAN loss, which allowed for more visually accurate results despite hardware limitations. The batch size was used equal to 32 as in the paper.

3.4 Experimental setup and code

We used Kaggle for our experiments, where we added our dataset containing 1,000 images per concept to train and test the base model. Regarding LoRA, we have an additional 1,000 images, from which 400 images are extracted using the k-means algorithm. Experiments can be replicated by running our notebooks; we have two different notebooks: one for the main training and another for the LoRA fine-tuning. Each notebook contains the necessary packages for installation. During the training phases, we continuously monitored the generator loss, discriminator loss, and, most importantly, the validation loss. Our goal was to keep the validation loss as low as possible. In fact, we achieved a validation loss of 0.3 at the end of the training. After running the experiments, we evaluated the base model using the FID score, which is used to compare the similarity between images generated by the model and input images generated by the diffusion model. For each prompt, 30 images are taken from the test set, and for each concept, the final FID score is calculated using the `Clean-FID` function Parmar et al. (2022). For the fine-tuned model, we evaluated it differently. To replicate the idea of the paper, the fine-tuned model must understand the new prompt, generate the correct image, and also retain what it learned during the initial training. Therefore, the evaluation is qualitative rather than relying on specific scores. To execute the code in a ready-made environment, you can visit this notebook: E2GAN Implementation, which contains the E2GAN base model along with FID calculations, and LoRA Implementation, which includes k-means clustering for the images and the LoRA implementation using `peft` to add concepts to the base model. For the complete code, we also provide the link to the repository: GitHub.

3.5 Computational requirements

We used the Kaggle GPU T4 for our experiments. In terms of time, a full training run for the model over 5 concepts (a total of 5,000 images) takes an average of 150 minutes with a batch size of 32 (a hyperparameter specified in the paper). Fine-tuning with LoRA is significantly faster, requiring only 15 minutes.

4 Results

Our results confirm, though loosely, some of the main claims of the original paper. This support is based on observations that, while not exactly reproducing the conditions or levels of accuracy described in the original study, are in line with the general conclusions proposed by the authors. Result images can be seen at Appendix A Figure 3.

4.1 Results reproducing original paper

The following subsections present the results of our experiments aimed at reproducing the key claims of the original paper.

4.1.1 Result 1: Efficient Transfer Learning with Mixed Datasets

The first claim was successfully validated. It was possible to train a GAN on a mixed dataset comprising diverse concepts, enabling efficient transfer learning to new tasks. This eliminates the need to train separate models from scratch for each new concept, demonstrating the utility of this approach.

4.1.2 Result 2: Low-Rank Adaptation for Fine-Tuning

Fine-tuning the GAN model using LoRA, by updating only approximately 1% of the model's parameters, demonstrated the ability to incorporate new concepts into the model. However, we did not achieve high-quality image results comparable to or better than those produced by conventional GAN training methods (see Appendix A Figure 6 for details).

4.1.3 Result 3: Data Efficiency in Fine-Tuning

The third claim was partially validated. It was shown that a GAN could be fine-tuned on a new concept by leveraging a small subset of representative images selected using clustering techniques. This approach, combined with LoRA, significantly reduced training time while maintaining reasonable performance.

5 Discussion

Our experimental results provide varying levels of support for the claims in the paper. The first claim is supported. Training a base GAN on a diverse dataset enabled efficient transfer learning, eliminating the need for separate models. However, we could not evaluate the limits of this generalization due to time constraints.

Second claim is partially supported. LoRA effectively added new knowledge by fine-tuning only 1% of the parameters, but image quality did not match conventional GAN training methods. Testing alternative parameter ratios could offer further insights.

The third claim is also partially supported. Using a subset of representative images reduced training time while maintaining reasonable performance. However, slight image fidelity degradation suggests a trade-off between data efficiency and quality.

About the fourth claim is inconclusive. Due to lacks of information by the paper about the details, we could not fully replicate this claim. Preliminary results suggest potential improvements in FID and training efficiency, warranting further exploration.

The fifth claim is still not evaluated. We did not implement the model on mobile devices, leaving this claim untested. We tried to convert the model to a mobile version (ptl) but unfortunately did not get any relevant results for the purpose of reproducing the paper.

5.1 What was easy

Unlike typical reproducibility studies where often at least some aspects-such as the implementation of standard components or the use of established frameworks-are relatively straightforward, the reproduction of this paper presented no particularly easy elements.

The total absence of source code, as already pointed out, was a key obstacle that affected every aspect of implementation. Even components that would normally be considered standard, such as the implementation of the basic GAN architecture or the integration of a simple prompt, required significant interpretation and development work from scratch. The lack of implementation details necessitated a trial-and-error approach to even seemingly basic aspects.

Even the use of standard libraries such as `PyTorch` required significant work to adapt and optimize, as many crucial details about the architecture and training process were not sufficiently specified in the paper. The LoRA implementation itself, which in other contexts might benefit from reference implementations, required substantial reinterpretation to fit the specific context of this work.

The only element that could be considered relatively manageable was the generation of images with diffusion models, but even this was difficult due to lack of generation parameters.

5.2 What was difficult

The main difficulty encountered was the total absence of reference implementations of the E2GAN model, both official and unofficial, further complicated by significant discrepancies between the computational resources described in the paper and those actually available. While the paper used an NVIDIA H100 GPU with 80GB of memory, our implementations had to adapt to more limited hardware, requiring modifications to the architecture and training parameters.

One critical aspect was the use of Clean-FID metrics instead of traditional FID, a detail not immediately apparent in the paper that required significant changes to the evaluation pipeline. This methodological difference created difficulties in interpretation and analysis.

Replication of the experimental conditions proved virtually impossible, particularly with regard to the number of concepts used. The paper described the use of 260 different concepts, but in practice this proved unfeasible: generating a single set of images using diffusion models took about an hour, making training on the entire set of concepts computationally prohibitive.

In addition, we encountered significant problems with specific prompts such as “green lantern” and “silver sculpture,” which produced inconsistent results when used with diffusion models. This caused a cascade of problems in the training and testing phase, as the quality of the training dataset was compromised by the poor quality of the generated images. These limitations were not adequately discussed in the original paper, which presented only the successful cases.

Replication of the results was further complicated by these inconsistencies between theoretical description and practical implementation challenges. In particular, generating high-quality training datasets proved much more complex and time-consuming than suggested in the paper, requiring significant trade-offs between fidelity to the original implementation and practical feasibility.

These difficulties required a pragmatic approach, with substantial modifications to the training process and concept selection, inevitably leading to divergence from the results reported in the original paper. Our final implementation thus represents a balance between fidelity to the original design and adaptations necessary to achieve a workable system with limited computational resources.

In the following, the Flickr-Scenery dataset could not be found. In addition, pre-trained components had to be used, which, however, were not fully adequate for the specific needs of the project.

5.3 Communication with original authors

Given the range of issues described above, it was decided to establish direct communication with the authors of the paper. Using the email addresses provided in the paper, we contacted the research team with specific questions regarding model architecture, LoRA implementation details, and training configurations.

Despite attempts at communication, the response was extremely limited. Only one of the authors initially responded to our inquiries, providing some general implementation guidance. However, even this communication quickly broke down after a few exchanges, leaving many of our technical questions unanswered.

The main barrier to sharing detailed information turned out to be intellectual property: the code and implementation details are owned by Snapchat, significantly limiting the authors' ability to share specific information. This situation further complicated the reproduction process, forcing us to rely solely on the descriptions provided in the paper and to develop alternative solutions for the parts that were not sufficiently detailed.

The experience highlighted one of the key challenges in reproducing research developed in industrial settings, where intellectual property can often limit the sharing of implementation details crucial to scientific reproducibility.

6 Conclusion

Our project demonstrated significant progress in validating key claims, including the efficiency of transfer learning with mixed datasets, the feasibility of fine-tuning GANs using LoRA with only 1% of parameters, and the effectiveness of clustering techniques for reducing training data requirements. However, fine-tuned models struggled to achieve consistently high-quality outputs, and time constraints limited our ability to evaluate E2GAN's full potential or test its real-time performance on mobile devices. Future work could focus on improving image quality, testing on mobile platforms, and conducting broader evaluations to strengthen the findings and explore generalization capabilities.

Member contributions

To manage the work, it was decided to start with an initial phase of in-depth understanding of the paper. Each member of the group spent time on a personal and technical study to analyze the content, understand the methodologies described and evaluate the objectives to be achieved.

Following this phase, a collective brainstorming session was held, during which each of the four members expressed their ideas, observations and suggestions. This discussion helped to identify the areas on which to focus more attention and to define a shared operational strategy.

To ensure a balanced contribution, each member worked on all the main points of implementation. Weekly update calls were held during the weekends, where each presented his or her progress. One of the first collaborative activities was the implementation of independent versions of the Generator: each member developed his or her own version, and then a mix of the various proposed architectures was made, integrating the best ideas that emerged.

The work was conducted incrementally: after establishing the basis of the Generator, subsequent components were added in a pyramidal fashion. Given the complexity of reproducing the paper, it was not easy to break the work down into isolated phases; instead, the project was based on a team approach, with shared sessions devoted to understanding technical details and troubleshooting.

This method required a coordinated and continuous effort, but allowed the challenges posed by the project to be met with input from all team members. Adopting a collaborative structure ensured not only a better understanding of the problem and technical solutions, but also greater efficiency in solving the difficulties encountered,making the project collaborative and dynamic.

References

Ffhq-256 (images only), October 2022. URL https://www.kaggle.com/datasets/denislukovnikov/ffhq256-images-only.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. *arXiv preprint arXiv:2211.09800*, 2022.

Yifan Gong, Zheng Zhan, Qing Jin, Yanyu Li, Yerlan Idelbayev, Xian Liu, Andrey Zharkov, Kfir Aberman, Sergey Tulyakov, Yanzhi Wang, et al. E2gan: Efficient training of efficient gans for image-to-image translation. *arXiv preprint arXiv:2401.06127*, 2024.

Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2022.11.139. URL https://www.sciencedirect.com/science/article/pii/S0020025522014633.

Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Torchvision ResNet50. Resnet50 — torchvision main documentation. URL https://pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. URL https://arxiv.org/abs/2312.12148.

A Appendix

Figures Here are reported some images cited into our report.

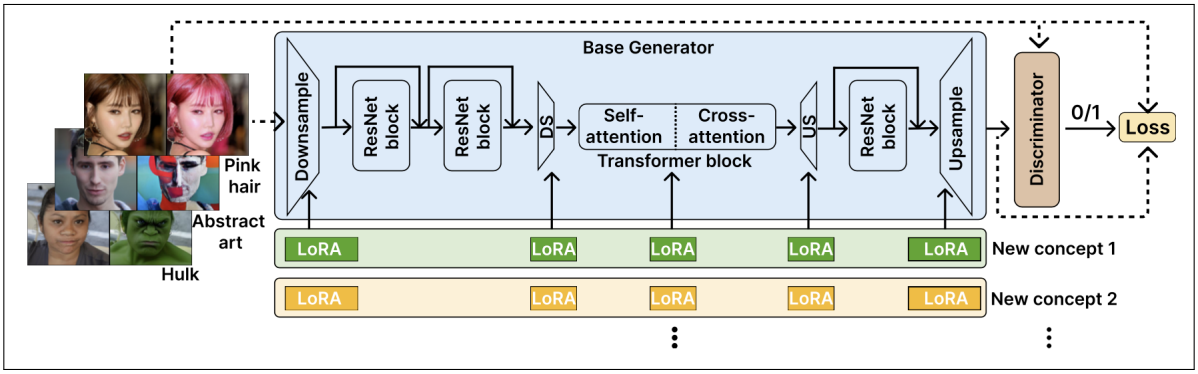

Figure 1: E2GAN structure

Figure 2: Dataset example

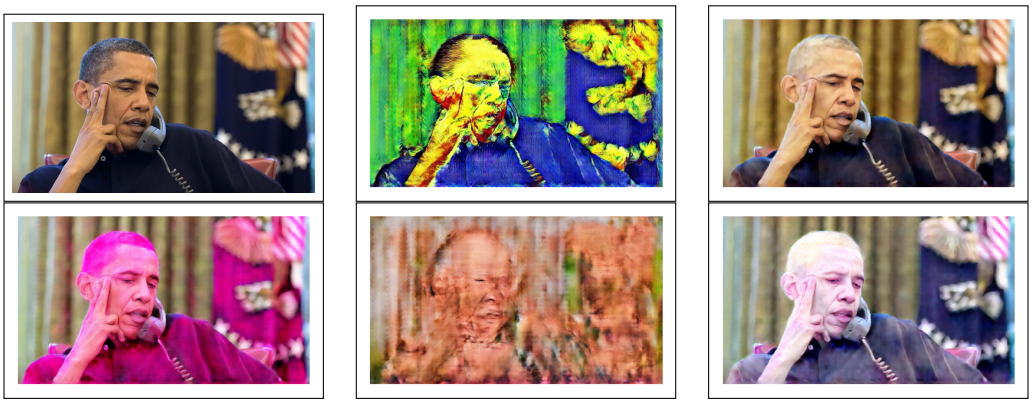

Figure 3: Generated examples

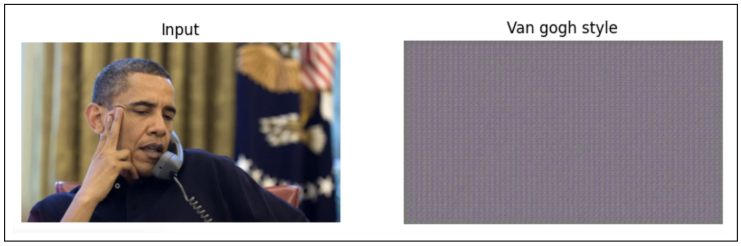

Figure 4: First LoRA Van Gogh

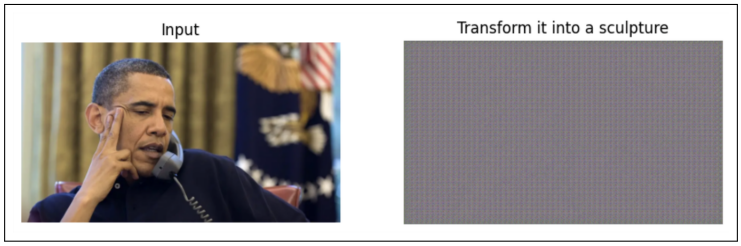

Figure 5: First LoRA Sculpture

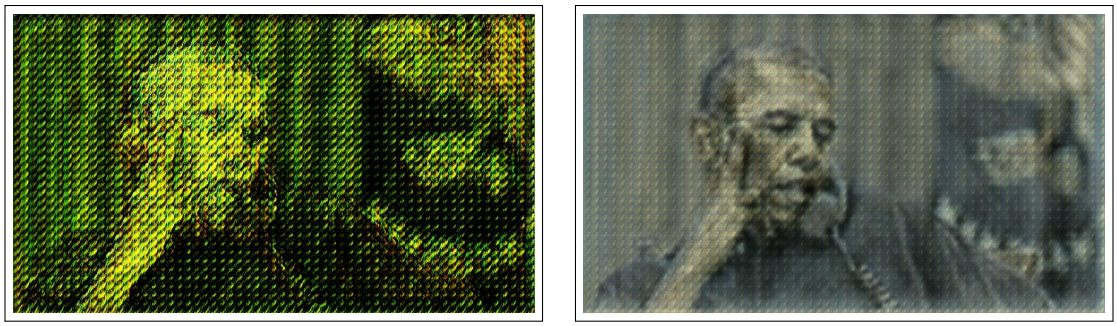

Figure 6: Generated LoRA images

B Appendix

Table Following there is the table relative to our base model images.

Table 1: FID Score Table

	Van Gogh	**Blonde Hair**	**Pink Hair**	**Old Person**	**Albino**
FID	138	150	87	218	114