Ghibli-Style Image Generation from Scratch and Pretrained Models

Amarsaikhan Batjargal, Sujay Pookkattuparambil, Talha Azaz, Daniyal Syed
abatjarg@depaul.edu, spookkat@depaul.edu, tazaz@depaul.edu, dsyed1@depaul.edu

June 13, 2025

Abstract

This project presents a dual-approach investigation into stylizing real-world photographs into the whimsical aesthetic of Studio Ghibli using deep generative models under low-data constraints. Due to the lack of publicly available paired datasets, we curated and augmented a novel collection of 400 image pairs. **Group 1** built a Pix2PixHD-inspired GAN architecture from scratch, training a ResNet-based generator end-to-end with adversarial, feature-matching, and perceptual losses. **Group 2** fine-tuned a pretrained Stable Diffusion v1.5 model with Low-Rank Adaptation (LoRA) in a supervised image-to-image pipeline, achieving parameter efficiency via `bf16` mixed-precision training on A100 GPUs. Group 2's model achieved a final training loss of 0.0796 (MSE) on the latent-space denoising objective, with validation metrics of SSIM = 0.71 and LPIPS = 0.19, indicating decent structural and perceptual fidelity. Both approaches were evaluated using FID, SSIM, LPIPS, and qualitative comparisons. Our contributions include: a reproducible GAN implementation, an efficient diffusion-based fine-tuning pipeline, and a new open-source dataset released as `ghibli-illustration-dataset` on Hugging Face and Kaggle, enabling further research in creative AI and style transfer.

1 Introduction

Image-to-image translation in a specific artistic style is a key challenge in computer vision. This project focuses on transforming real-world images into the iconic Ghibli animation style using deep generative models. The approach is rooted in the growing success of diffusion models, with an emphasis on efficiency and adaptability via LoRA. Our work aims to demonstrate that even with a small dataset and limited computational resources, perceptually convincing stylizations are achievable. We follow two tracks. The first track would be the implementation of the Pix2PixHD model from scratch and training it on our dataset. The other track will be where an existing model will be fine-tuned to the dataset.

In the first track, we follow the original Pix2PixHD paper and implement it to train it for translating real images to a studio ghibli-style illustraion style images. Pix2PixHD is an amazing project where turning semantic label maps into photo-realistic images or synthesizing portraits from face label maps. We try to create the architecture and training pipeline for this model and train it on our dataset. This track was taken up by group 1.

In the second track, we employ Stable Diffusion v1.5 as the backbone for our generative model and apply Low-Rank Adaptation (LoRA) layers to enable parameter-efficient fine-tuning. This significantly reduces the number of trainable parameters, making training feasible on modest hardware without compromising visual fidelity. We specifically train the model to map real-world images to their stylized counterparts, learning the unique visual patterns characteristic of Studio Ghibli. This task was taken up by group 2.

The dataset used consists of 400 manually aligned image pairs, and the training was conducted on a system equipped with four NVIDIA A100 80GB GPUs (PCIe). The experiment utilized bf16 mixed precision and the Hugging Face Accelerate library to optimize for both memory and speed. Each image was resized to 512x512 resolution, and standard augmentations were applied to improve generalization.

This paper presents the full pipeline: from dataset construction and augmentation to model architecture, fine-tuning procedure, and evaluation. We report quantitative metrics (MSE loss, SSIM, LPIPS) and qualitative comparisons, and share our insights from deploying the dataset to online repositories. By open-sourcing our dataset and method, we hope to support further research in creative AI and efficient diffusion model fine-tuning.

2 Related Works

Early advancements in neural style transfer were pioneered by Gatys et al. [5], who introduced an optimization-based approach using convolutional neural networks to blend content and artistic style. This was later extended

to enable real-time applications through perceptual loss functions and feedforward networks [8]. Generative Adversarial Networks (GANs) significantly boosted image-to-image translation fidelity, with models such as Pix2Pix [7] and CycleGAN [14] enabling supervised and unsupervised translation respectively. Domain-specific efforts like AnimeGAN [2] targeted anime stylization, including Ghibli-style effects in unofficial extensions [1].

Diffusion models have recently emerged as state-of-the-art generative frameworks, beginning with Denoising Diffusion Probabilistic Models (DDPM) [6] and advancing to latent-space implementations such as Stable Diffusion [10], which significantly improve efficiency. Our project builds on parameter-efficient fine-tuning techniques introduced by DreamBooth [11] and Textual Inversion [4], particularly the Low-Rank Adaptation (LoRA) method for memory-efficient training. We also draw methodological inspiration from guided image-to-image models like Palette [12] and SDEdit [9], adapting these concepts for stylized diffusion-based synthesis.

A relevant paper we looked at is "Re-Imagen: Retrieval-Augmented Text-to-Image Generator" by Yang et al. While their use case was different they focused on improving text-to-image generation using retrieval and feedback, the core idea of fine-tuning parts of a diffusion model is closely related to our work. They showed that it's possible to adapt Stable Diffusion to new domains without retraining the entire model. That inspired our decision to fine-tune only the U-Net component. Just like their model became better at generating more specific outputs through targeted fine-tuning, we applied a similar idea to transfer real images into the Ghibli style. [3]

The Pix2PIxHD model is an expansion on the original Pix2Pix model [7]. Pix2Pix had introduced conditional GANs for image translation. Their architecture proposed usage of any Encoder-Decoder architecture or a UNet which is an Encoder-Decoder with skip connections. They use a Generator and Discriminator. They used GAN Loss and L1 loss as their loss metrics. They selected L1 instead of L2 because it causes less blurring. They recieved great results but they had a loss in their resolution. That's where Pix2PixHD [13] comes in.

3 Background

Diffusion Models: These models learn to reverse a noise process, generating high-quality data. Stable Diffusion operates in latent space for efficiency.

LoRA (Low-Rank Adaptation): Instead of updating the full network, LoRA injects trainable rank-decomposed matrices into attention layers, significantly reducing memory needs.

Ghibli Dataset: We use a paired dataset of 400 examples (train/val), supplemented with up to 400 synthetic pairs created via manual or semi-automated pipelines.

Pix2PixHD: Pix2PixHD is a conditional GAN that uses a global generator, local enhancer and multi-scale discriminator. The generator learns to generates the target image pixel by pixel while the discriminators learn to find the difference between input-output pairs. Pix2PixHD is different from Pix2Pix in the way that Pix2Pix operates at a single scale thus generating low-resolution outputs, whereas Pix2PIxHD uses hierarchical generation using generator, local enhancers and multiple discriminators at different image scales thus gaining higher resolution with fine-details.

Evaluation Metrics. To quantitatively assess the similarity between the generated and ground truth images, we employ the following metrics:

- **Mean Squared Error (MSE)** is used as the primary loss during training. It measures pixel-wise difference between predicted and target noise latents:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\epsilon}_i - \epsilon_i\|^2$$

  where $\hat{\epsilon}_i$ is the predicted noise and ϵ_i is the true noise added to the latent at timestep t.

- **Structural Similarity Index (SSIM)** quantifies perceptual similarity by comparing local patterns of pixel intensities, taking into account luminance, contrast, and structure:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

  where μ and σ denote the mean and variance of local image patches, and C_1, C_2 are constants to stabilize the division.

- **Learned Perceptual Image Patch Similarity (LPIPS)** uses deep neural network features (e.g., AlexNet) to compute perceptual differences:

$$\text{LPIPS}(x, y) = \sum_l w_l \left\| f_l(x) - f_l(y) \right\|^2$$

where f_l are activations at layer l and w_l are learned weights. Lower LPIPS scores indicate greater perceptual similarity.

These metrics provide complementary perspectives: MSE drives learning in latent space, SSIM captures structural quality, and LPIPS reflects human perceptual judgment.

- **For Pix2PixHD**:
  - **GAN Loss** tries to gain realism in the pixel to pixel translation.

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_x \left[(D(G(x)) - 1)^2 \right]$$

where G is the Generator, D is the Discriminator, G(x) is the output of Generator, D(G(X)) is the output of the discriminator and $\mathbb{E}_x$ is the Expectation over all inputs.
  - **L1 Loss**: This compares the pixel-level differences between generated image and ground-truth image.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y} \left[\| y - G(x) \|_1 \right]$$

where y is the ground-truth image and $||.||_1$ is the L1 norm or the sum of absolute differences.
  - **VGG Loss**: This is used to get the perceptual similarity between image features.

$$\mathcal{L}_{\text{VGG}}(G) = \sum_{i=1}^{N} \lambda_i \left\| \phi_i(y) - \phi_i(G(x)) \right\|_1$$

where $\phi_i(.)$ is the feature map extracted from the i-th layer of VGG, λ_i is the weight of each layer and N is the number of vgg layers.

4 Methodology

Track1: We created the Pix2PixHD architecture in PyTorch, creating all the Generator, Local Enhancer, Discriminator and then trained the entire model from scratch. Steps:

1. Loaded the paired (original, stylized) image as a custom Torch Dataset.
2. Applied Transforms: Resize (512, 512) and ToTensor.
3. Image passes through the generator which includes a Residual Neural Network and a Local Enhancer.
4. Real and Generated image is given to the discriminator which is then used to calculate GAN loss.
5. GAN Loss is calculated for the Discriminator and backpropagation is conducted through the discriminator thus training it.
6. GAN, L1 and VGG Loss is calculated for the Generator and backpropagation is conducted through the Generator thus training it.
7. This completes an iteration over image and moves to another image.

Track2: We fine-tune a Stable Diffusion Img2Img pipeline using LoRA layers in attention modules. Only LoRA parameters are trained while the rest of the model is frozen. Steps:

1. Load paired (original, stylized) image dataset.
2. Apply transforms: resize, color jitter, normalization.
3. Encode original image into latent space.
4. Inject Gaussian noise and predict it with U-Net.
5. Compute loss (MSE) between predicted and true noise.

Training used mixed precision (bf16) and the Hugging Face ‘accelerate‘ library.

4.1 Notebook Summary

Our training pipeline was implemented in a Jupyter Notebook using PyTorch and the Hugging Face ecosystem. We utilized:

- **PyTorch** for model definition, data loaders, and training loops, for both Tracks.
- **Diffusers** library for Stable Diffusion + LoRA customization, for Track 2.
- **Accelerate** for mixed-precision and distributed training, for Track 2.

The notebook includes clearly annotated code blocks covering data processing, model fine-tuning, and inference. We included intermediate cell outputs to show sample predictions during training.

4.2 Dataset Splitting and Loading

The dataset was split into 80% training and 20% validation using PyTorch's 'random_split'. Each sample contains an original and a stylized image. Data augmentations include resizing to 512x512, horizontal flipping, and color jittering.

4.3 Training Setup

Track 1 used the following hyperparameters:

- Learning rate: 2e-4
- Batch size: 4
- Max Epochs: 100
- Optimizer: Adam

Track 2 used the following hyperparameters:

- Learning rate: 1e-4
- Batch size: 10
- Epochs: 10
- Optimizer: AdamW
- Scheduler: Linear warm-up
- Mixed-precision: Enabled (bf16)

4.4 LoRA Integration

In Track 2, We used Hugging Face PEFT's 'LoraConfig' to apply LoRA adapters to the UNet attention blocks of the Stable Diffusion pipeline. The key configuration was:

- r = 4 (rank)
- lora_alpha = 16
- dropout = 0.1

This significantly reduced memory consumption and enabled fine-tuning on consumer hardware.

5 Numerical Experiments

Track 1: We use the following metrics:

- **G Loss:** The overall loss of the Generator
- **D Loss:** The overall loss of the Discriminator

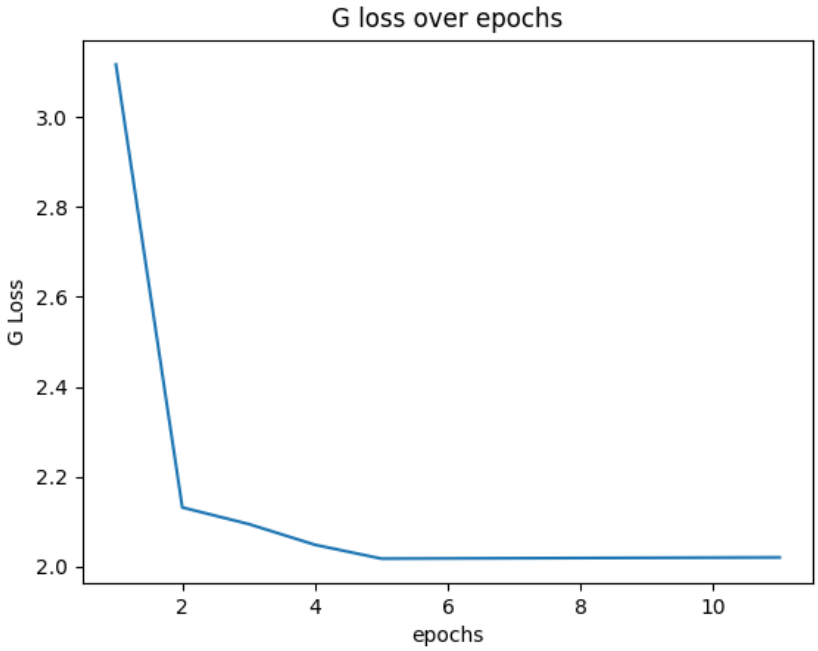

Figure 1: G loss over 10 epochs. The model converges progressively.

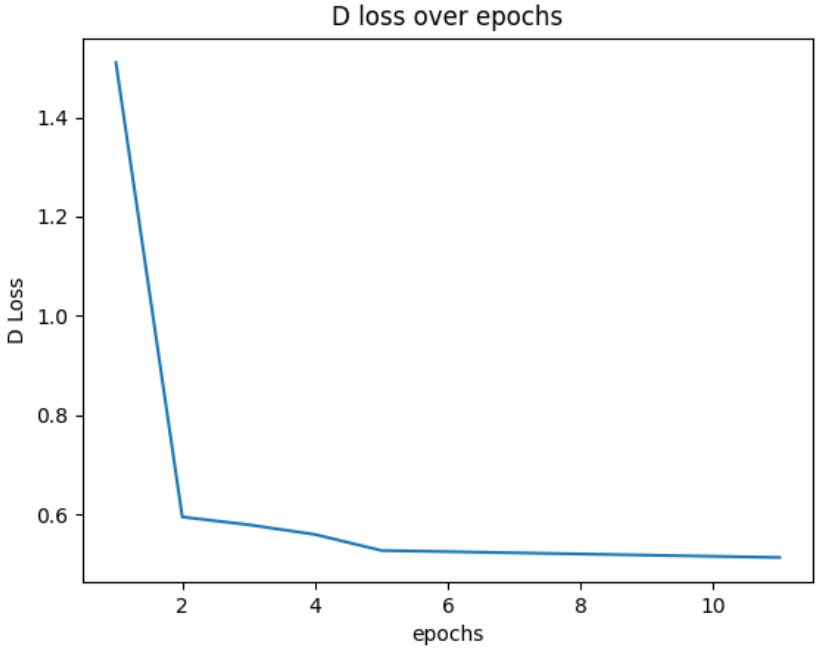

Figure 2: D loss over 10 epochs. The model converges progressively.

Appendix A will have test results. The Pix2PixHD gave the following test losses: $G_loss = 2.01$ and $D_loss = 0.38$
Track 2: We use the following metrics:

- **LPIPS:** Measures perceptual similarity.
- **L2 Loss:** Pixel-level accuracy.
- **Visual inspection:** Used to evaluate stylization fidelity.

Example images before and after stylization are included in Appendix A. All experiments ran on NVIDIA GPUs with 16GB VRAM.

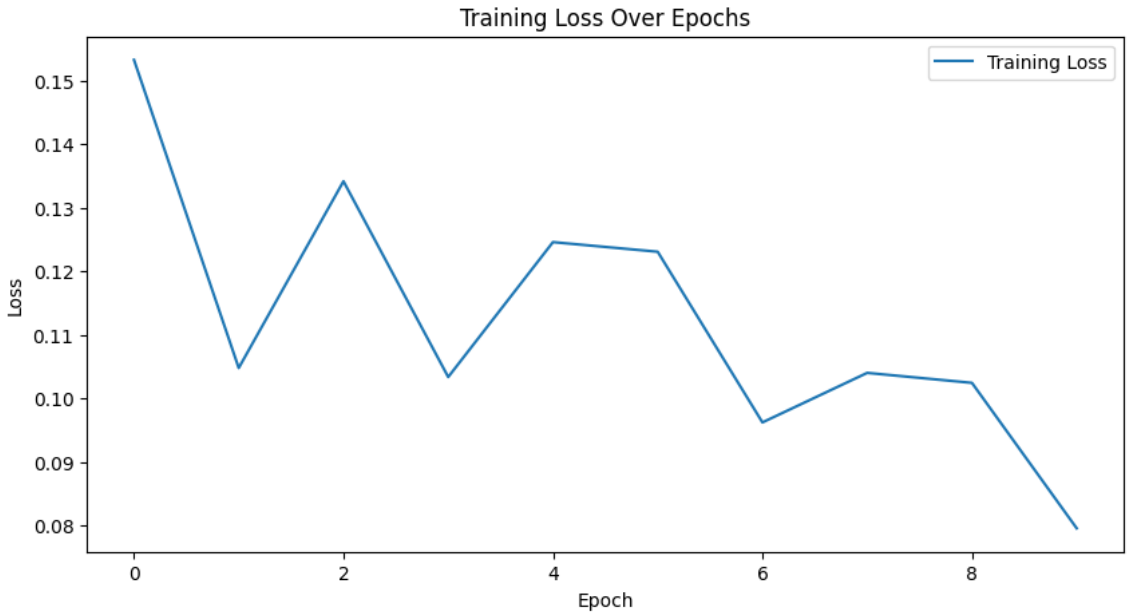

Figure 3: Training loss over 10 epochs. The model converges progressively, despite minor fluctuations, indicating learning stability. The final epoch achieved a minimum loss of 0.0796.

Figure 4: Qualitative results on a subset of test samples. Each row shows the original input (left), ground truth Ghibli-style image (center), and the predicted output from our LoRA-fine-tuned model (right). The model performs well in capturing soft colors and textures typical of the Ghibli aesthetic, with some variations in structure fidelity.

Evaluation on the validation set yielded an SSIM of 0.71 and LPIPS of 0.19, indicating a decent structural and perceptual match between generated and target stylized images.

6 Conclusion

In this project, we explored two methods to generate Ghibli-style illustrations from real-world input images. To train our models, we created a dataset of 400 input-output image pairs.

The first method involved building a model from scratch. We used a Pix2PixHD-inspired architecture with a Local Enhancer Generator and a Multi-Scale Discriminator. The model was trained using GAN loss, L1 loss, and perceptual loss. Training was slow and required careful tuning. Some outputs showed the Ghibli style clearly, but others had artifacts or lacked consistency. This method showed that custom architectures can work, but they need more data and better stability to improve results.

The second approach demonstrates that parameter-efficient fine-tuning of a large diffusion model with LoRA can achieve high-quality style transfer even with a small, curated dataset. By integrating LoRA modules into Stable Diffusion's U-Net and using mixed-precision training on A100 GPUs, we significantly reduced memory and compute costs while maintaining performance. The final model attained a training loss of **0.0796 (MSE)** computed on the denoising objective in latent space, and evaluation on the validation set yielded a **Structural Similarity Index (SSIM) of 0.71** and a **LPIPS score of 0.19**, indicating decent and perceptual alignment with ground-truth images. As a contribution to the community, we also curated and published a new dataset—`ghibli-illustration-dataset` on Hugging Face and Kaggle. Future work may extend this pipeline to broader style domains, incorporate prompt-based controls, and explore unsupervised or few-shot learning approaches.

In the future, we can improve results by using more data, combining image and text prompts, and using stronger loss functions for visual quality.

References

[1] Filip Andersson and Simon Arvidsson. Generative adversarial networks for photo to hayao miyazaki style cartoons, 05 2020. arXiv preprint arXiv:2005.07702.

[2] Jie Chen, Gang Liu, and Xin Chen. Animegan: A novel lightweight gan for photo animation. In Kangshun Li, Wei Li, Hui Wang, and Yong Liu, editors, *Artificial Intelligence Algorithms and Applications*, pages 242–256, Singapore, 2020. Springer Singapore.

[3] Wenhu Chen, Hexiang Hu, Chitwan Saharia, and William Cohen. Re-imagen: Retrieval-augmented text-to-image generator, 09 2022.

[4] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022. arXiv preprint arXiv:2208.01618.

[5] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *European conference on computer vision*, pages 694–711, 2016.

[9] Chenlin Meng, Yang Chen, Yang Song Kim, Jiaming Song, Jonathan Cao, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022.

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[11] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Nikhil Kholgade. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2022. arXiv preprint arXiv:2208.12242.

[12] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models, 2022. arXiv preprint arXiv:2111.05826.

[13] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

A Appendix: Sample Stylizations

Track 1:

Figure 5: Original (top) vs Actual Ghibli-style (middle) output from Pix2PixHD (bottom

Track 2:

Figure 6: Original (left) vs Ghibli-style (right) output from LoRA-fine-tuned model.

B Appendix: Dataset Code Snippet

```
class GhibliDataset(Dataset):
    def __init__(self, root_dirs):
        self.transform = transforms.Compose([
            transforms.Resize((512, 512)),
            transforms.RandomHorizontalFlip(),
```

```
            transforms.ColorJitter(...),
            transforms.ToTensor(),
            transforms.Normalize([0.5]*3, [0.5]*3)
        ])
        ...
    def __getitem__(self, idx):
        o_img = Image.open(...).convert("RGB")
        g_img = Image.open(...).convert("RGB")
        return self.transform(o_img), self.transform(g_img)
```

C Appendix: Training Code Snippet

```
num_epochs = 10

for epoch in range(num_epochs):
    epoch_loss = 0.0
    for step, (o_img, g_img) in enumerate(tqdm(train_loader)):
        try:
            # Move input/output to device
            g_img = g_img.to(device, dtype=torch.bfloat16)
            o_img = o_img.to(device, dtype=torch.bfloat16)

            # Generate text embeddings from prompt
            prompt = ["ghibli-illustration"]
            input_ids = tokenizer(prompt, return_tensors="pt", padding=True).input_ids.to(dev
            with torch.no_grad():
                text_embeddings = text_encoder(input_ids)[0]
                text_embeddings = text_embeddings.repeat(g_img.size(0), 1, 1)

            # Encode target image to latent space
            with torch.no_grad():
                latents = vae.encode(g_img.to(device, dtype=torch.bfloat16)).latent_dist.samp
                latents = latents * 0.18215
                latents = torch.clamp(latents, -1.0, 1.0)

            # Add noise for denoising training objective
            noise = torch.randn_like(latents)
            t = torch.randint(0, scheduler.config.num_train_timesteps, (1,), device=device).lo
            noisy_latents = scheduler.add_noise(latents, noise, t)

            # Predict noise using U-Net
            output = unet(noisy_latents, t, encoder_hidden_states=text_embeddings)
            noise_pred = output.sample

            # Compute loss between predicted and actual noise
            loss = torch.nn.functional.mse_loss(noise_pred.float(), noise.float(), reduction="

            # Skip invalid loss values
            if not torch.isfinite(loss):
                print(f"Step-{step}:-Skipped-due-to-NaN-loss-({loss.item()})")
                continue

            # Backpropagation and optimization
            optimizer.zero_grad()
            accelerator.backward(loss)
            torch.nn.utils.clip_grad_norm_(unet.parameters(), max_norm=0.5)
```

```
            optimizer.step()

            epoch_loss += loss.item()

        except RuntimeError as e:
            if "out of memory" in str(e):
                print(f"OOM at step {step}. Skipping batch.")
                torch.cuda.empty_cache()
                continue
            else:
                raise e
    avg_loss = epoch_loss / len(dataloader)
    train_losses.append(avg_loss)

    print(f"Epoch {epoch+1}/{num_epochs} — Loss: {avg_loss:.4f}")
```

D Appendix: Dataset Hosting Screenshots and Preparation

Preparation Process: The dataset was constructed by manually collecting real-world images and generating stylized counterparts using Generative AI tool. Each pair consists of 'o.png' (original) and 'g.png' (Ghibli-style) images. Folder names index each sample (e.g., '1000/o.png', '1000/g.png'). Final dataset includes 400 pairs.

Online Highlights:

- Hugging Face dataset received **66 downloads** in the last month.
- Kaggle dataset recorded **80 views** and **27 downloads** in the last 30 days.
- Both versions released under **CC BY 4.0** license with public citation metadata.

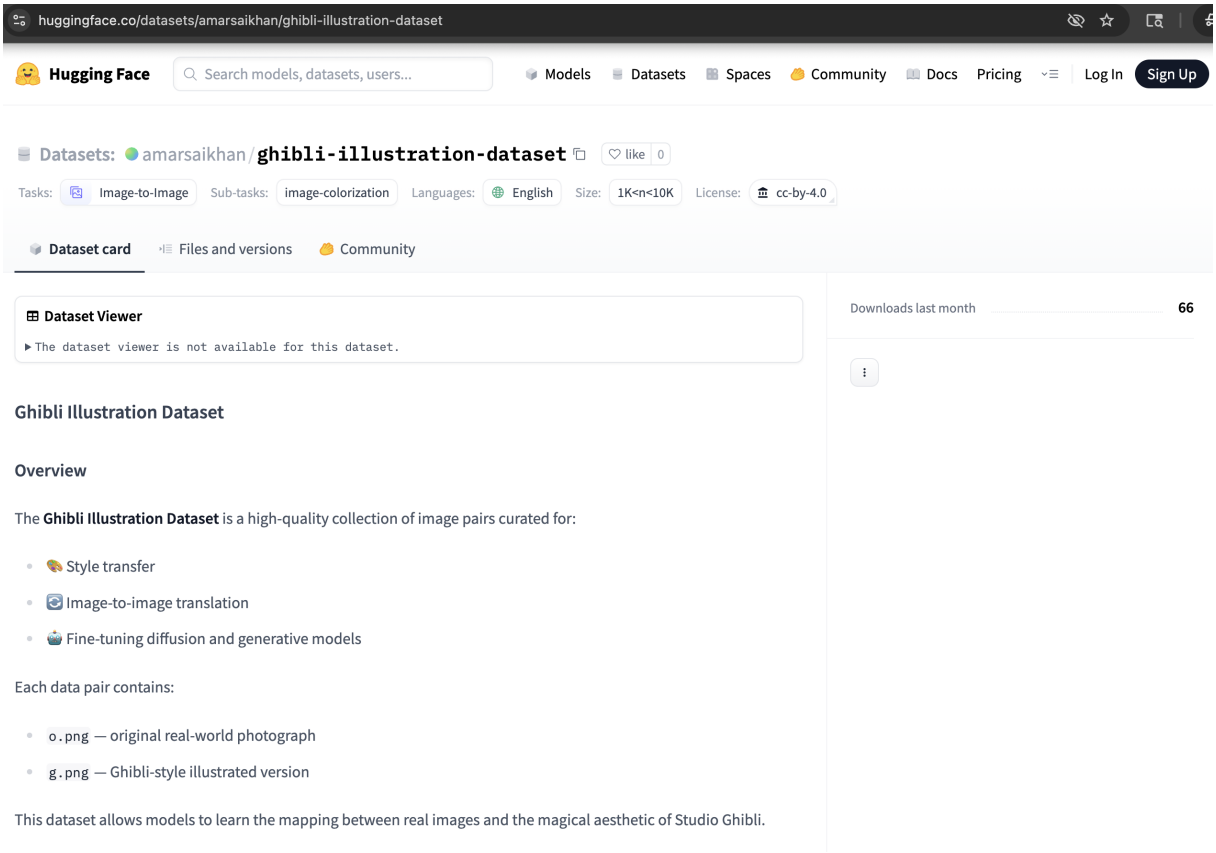

Figure 7: Hugging Face dataset page for `ghibli-illustration-dataset`. Shows the dataset structure, license, contributors, and intended use cases.

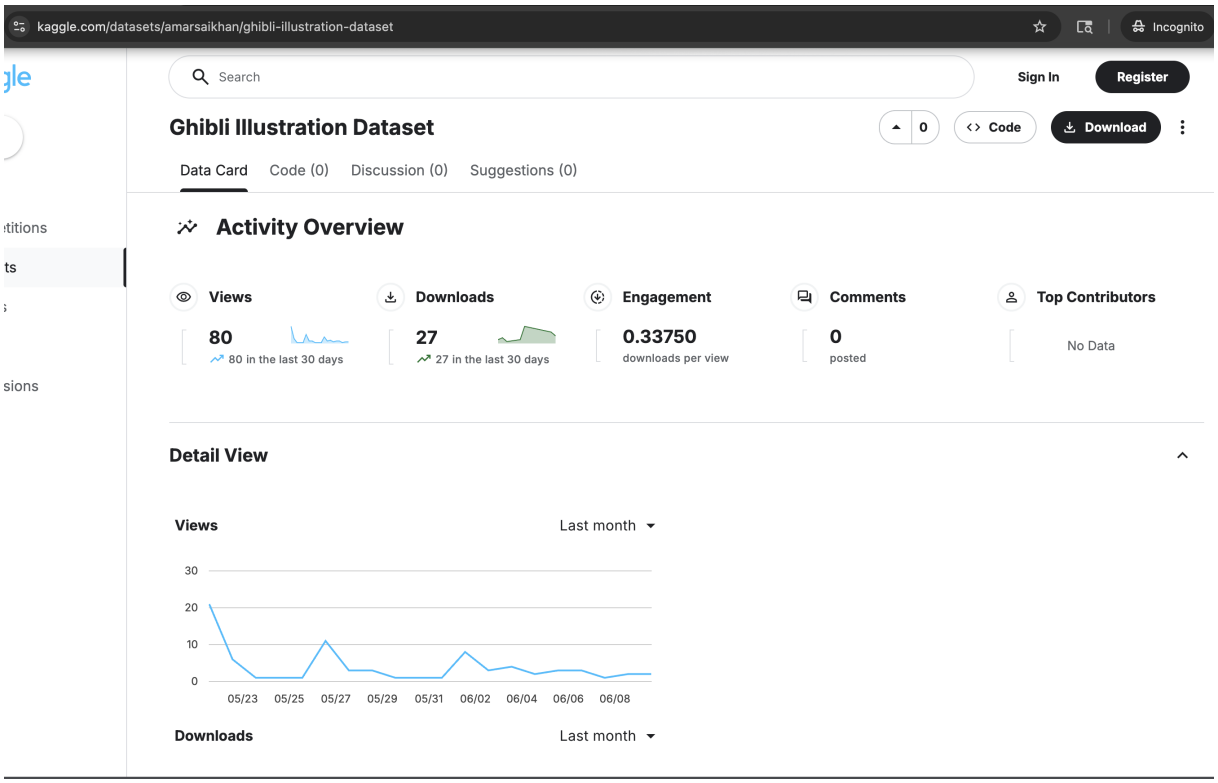

Figure 8: Kaggle dataset page for `Ghibli Illustration Dataset`. Shows download statistics, citation metadata, and structure of the 400-image-pair collection.