6.8370 Project Writeup

Harrison Zhang
zhanghy@mit.edu

1 Introduction

For my final project, I selected to do the new problem set on diffusion by CS 180 at University of California, Berkeley, "Fun With Diffusion Models! In this project, I played around with diffusion models, implemented diffusion sampling loops, and applied these techniques to inpainting and creating optical illusions. The code is linked here, which includes the deliverables requested.

1.1 Motivation

The foundations of 6.8370 have always been classical, in the sense that we did not use any deep learning techniques for image synthesis, editing, or composition in the normal assignments. Nor did I learn any of these techniques when I took CS 175 Computer Graphics at Harvard, which also only featured classical methods (though focused on OpenGL and not in image generation). Since this class was my first exposure to photography and vision, I do not have prior exposure to vision or deep learning techniques (e.g. diffusion) for image synthesis. Additionally, I have recently developed interest in robotic perception and simulation from 6.4212 Robotic Manipulation. Because of these recently developed interests in the fields of computational photography and computer vision, and because I have learned about but not worked directly with generative models, I explored an avenue where I could learn more about diffusion, deep learning for image synthesis, and generative models in general. Therefore, I decided to do this assignment as my final project.

1.2 Relevant work

1.2.1 Diffuse3D: Wide-Angle 3D Photography via Bilateral Diffusion

This work presents a novel method to generate wide-angle 3D photography using a single image. It addresses challenges in creating novel views, especially with large occlusions or extreme viewing angles. Existing work falls short because they treat all pixels equally and do not maintain depth consistency in the synthesized parts of the image. The core contribution is modifying the traditional diffusion model to incorporate depth in the denoising process and using bilaterial kernels to combine spatial and depth components. Starting from an RGB image and depth map, they construct layered depth images, then use the modified model to inpaint occluded regions, resulting in a well synthesized wide-angle view.

1.2.2 Learning 3D Photography Videos via Self-supervised Diffusion on Single Images

This work addresses limitations in previous 3D photography synthesis, which rely on monocular depth estimation, view synthesis, and particularly, inpainting models, which are often trained on out-of-domain data. To bridge the training and inference gap, this paper proposes a self-supervised diffusion model designed specifically for inpainting occluded regions during novel view synthesis. The model uses a random cycle rendering process to generate training data from single images, so ground truth labels are not required (they render the input from a new viewpoint to identify occluded regions, then back-project to the original viewpoint to form a training pair). They also plug a Masked Enhanced Block (MEB) into the UNet architecture, improving ability to handle occluded regions by making better use of spatial and semantic information.

1.2.3 Imagic: Text-Based Real Image Editing With Diffusion Models

This work proposes a novel method for editing real-world images using text prompts and diffusion models. Previous works are limited to synthetic images or require multiple input views, but this method uses only one single high-resolution image. It can perform complex, non-rigid edits such as changing an object's pose (e.g., making a dog sit or jump) or other semantic modifications, while maintaining high fidelity to the original image's details. Imagic uses a pre-trained model to apply text-guided edits without needing masks or additional inputs. It produces a text embedding that aligns with both the input and target, while fine-tuning the model to capture the desired image-specific appearance.

1.3 Specification

This project uses the DeepFloyd IF diffusion model, which is a two stage model trained by Stability AI. The first stage produces images of size 64×64 and the second stage takes the outputs of the first stage and generates images of size 256×256. The premise is implementing and deploying diffusion models for image generation. The random seed for this project is 180.

Preprint. Under review.

2 Architecture

At the core of this project is the diffusion model architecture. A diffusion model is a generative model that defines a distribution on the data as the result of a noise-removal process over time. It relies on a sequence of latent variables that evolve over time according to a discrete-time (for this project), finite Markov chain. State transitions of this chain are learned to reverse the diffusion process. The reverse process is also a Markov chain (with learned transitions) that adds noise to the data in the opposite direction of sampling until signal is destroyed.

2.1 Forward (diffusion) process

The forward process gradually adds Gaussian noise to the data (matrix) $x_0 \sim q(x_0)$ according to a schedule over $t = 1, \ldots, T$ controlled by parameters $\{\alpha_t\}_{t=1}^T$, transforming it into a sequence of intermediate states $\{x_t\}_{t=1}^T$. The approximate posterior is:

$$q(x_1, \ldots, x_T | x_0) = \prod_{t=1}^{T} q(x_t | x_{t-1}), \;\; q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)I\right)$$

2.2 Reverse (generative) process

The reverse process starts from zero signal state x_T and gradually denoises it to generate sample x_0. It is the joint $p_\theta(x_0, x_1, \ldots, x_T)$, and the Markovian formulation is also clear from the following:

$$p_\theta(x_0, x_1, \ldots, x_T) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

$$p(x_{t-1}|x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right), \;\; p(x_T) = \mathcal{N}(x_T; 0, I)$$

2.3 Training

The model is trained by optimizing the variational bound on log evidence to obtain loss $\mathcal{L}(x, q, p, \theta)$.

$$\mathbb{E}[-\log p_\theta(x_0)] \le \mathbb{E}_q \left[-\log \frac{p_\theta(x_0, x_1, \ldots, x_T)}{q(x_1, \ldots, x_T|x_0)} \right] = \mathbb{E}_q \left[-\left(\log p(x_T) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{x_t|x_{t-1}} \right) \right]$$

Essentially, the model attempts to reverse the denoising process. Given a noisy x_t and timestep t, the model attempts to learn noise (and variance) in the x_t. The amount of noise added at each step is dictated by fixed parameters $\bar{\alpha_t}$ (fixed by the model trainers).

3 Implementation

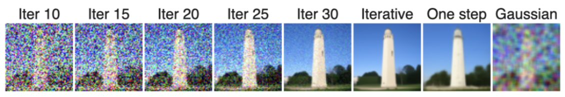
Here, we will walk through important algorithms implemented, (my) intuition behind some of them, and interesting results.

3.1 Classical, one-step, iterative denoising

We use Gaussian filtering for classical denoising, but these results are subpar for high levels of noise, which motivates non-classical methods. In one-step denoising, we use the model to denoise with text conditioning on "a high quality photo". We iterate over t with a noisy image as input. Then, we estimate the noise and variance, and retrieve the denoised image by recovering signal and scaling appropriately. However, to improve projection onto the natural image manifold even at low SNR, we should denoise over many iterations. Normally, we would need to run T one-step estimates, to obtain an estimate for x_{t-1} from x_t. But an (out of scope) optimization lets us set $t \leftarrow$ `strided_timesteps[i]` for the `i`-th step. On step `i`, we wish to get to $t' =$ `strided_timesteps[i+1]` from $t =$ `strided_timesteps[i]` via

$$x_{t'} = \frac{\sqrt{\bar{\alpha_{t'}}}\beta_t}{1-\bar{\alpha_{t'}}} x_0 + \frac{\sqrt{\alpha_{t'}}(1-\alpha_{t'})}{1-\bar{\alpha_{t'}}} x_t + v_\sigma$$

where $t' < t$, $\alpha_t = \bar{\alpha_t}/\bar{\alpha_t}$, $\beta_t = 1 - \alpha_t$ and v_σ is the computed random noise. This is quite neat because it can be interpreted as linear weights between x_0 (signal) and x_t (noise)!



3.2 Classifier free guidance

A corollary is that we can generate images from scratch by starting the strided timesteps at pure noise, `i_start=0`. However, these images are not very good quality. Classifier free guidance improves image quality at the expense of image diversity. In CFG, we compute both a conditional and unconditional noise estimate, ϵ_c, ϵ_u respectively, and the final noise estimate is controlled by scaling parameter γ. Below, we used the actual null prompt "" for unconditional guidance, and "a high quality photo" for conditional guidance. The implementation is the same, except the final noise estimate is a linear combination of the conditional and unconditional noise estimates (see Algorithm 2 for this weighting).

$$\epsilon = \epsilon_{\text{unconditional}} + \gamma(\epsilon_{\text{conditional}} - \epsilon_{\text{unconditional}})$$

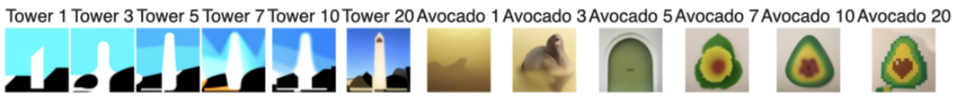
Algorithm 1 Iterative denoising

```
 1: Inputs:
        image, i_start, prompt_embeds, uncond_prompt_embeds, strided_timesteps, scale=7
 2: for i = i_start to len(timesteps)−1 do
 3:     Initialize:
        t, t_prev ← strided_timesteps[i], strided_timesteps[i+1]
        alpha_cumprod, alpha_cumprod_prev ← alphas_cumprod[t], alphas_cumprod[t_prev]
        alpha_t ← alpha_cumprod / alpha_cumprod_prev
        beta_t ← 1 - alpha_t
 4:     Core Algorithm:
 5:     noise_est, predicted_variance ← model run with current image state, encoder_hidden_states=prompt_embeds
 6:     x0_est ← (image - $\sqrt{1 - \text{alpha\_cumprod}} \cdot \text{noise\_est})/\sqrt{\text{alpha\_cumprod}})$
 7:     pred_prev_image ← $(\sqrt{\text{alpha\_cumprod\_prev}} \cdot \text{beta\_t}/(1 - \text{alpha\_cumprod})) \cdot \text{x0\_est} + (\sqrt{\text{alpha\_t}} \cdot (1 - \text{alpha\_cumprod\_prev})/(1 - \text{alpha\_cumprod})) \cdot \text{image}$
 8:     pred_prev_image ← add_variance(predicted_variance, t_prev, pred_prev_image)
 9: end for
10: clean ← image.cpu().detach().numpy()
11: return clean
```

3.3 Image-to-image Translation

3.3.1 Editing hand-drawn and web images

We start with a nonrealistic image (sketch) and project it onto the natural image manifold, using the CFG procedure. Indeed, the generated images are of great quality as seen below, but there is also a great degree of similarity across them.

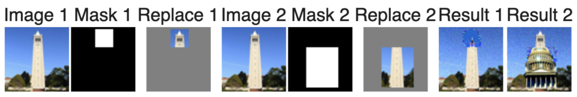


3.3.2 Inpainting

Another neat application is inpainting. We can create an indicator mask M where 1 represents new image content and 0 is forced to have the same pixels as the original image. Likewise, we can guide the manifold projection with a text prompt. However, then we are no longer doing the implicit natural image manifold projection. The projection is now controlled by the provided (conditional) prompt embedding.

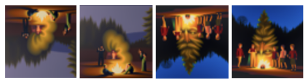
Algorithm 2 Masked inpainting

```
 1: Inputs:
        image, original, mask, i_start, prompt_embeds, uncond_prompt_embeds, strided_timesteps, scale=7
 2: for i = i_start to len(timesteps)−1 do
 3:     Initialize:
        t, t_prev ← strided_timesteps[i], strided_timesteps[i+1]
        alpha_cumprod, alpha_cumprod_prev ← alphas_cumprod[t], alphas_cumprod[t_prev]
        alpha_t ← alpha_cumprod / alpha_cumprod_prev
        beta_t ← 1 - alpha_t
 4:     Core Algorithm:
 5:     noise_est, predicted_variance ← inference with current image, encoder_hidden_states=prompt_embeds
 6:     uncond_noise_est, _ ← inference with current image, encoder_hidden_states=uncond_prompt_embeds
 7:     noise_est ← uncond_noise_est + scale * (noise_est - uncond_noise_est)
 8:     x0_est ← (image - $\sqrt{1 - \text{alpha\_cumprod}} \cdot \text{noise\_est})/\sqrt{\text{alpha\_cumprod}})$
 9:     pred_prev_image ← $(\sqrt{\text{alpha\_cumprod\_prev}} \cdot \text{beta\_t}/(1 - \text{alpha\_cumprod})) \cdot \text{x0\_est} + (\sqrt{\text{alpha\_t}} \cdot (1 - \text{alpha\_cumprod\_prev})/(1 - \text{alpha\_cumprod})) \cdot \text{image}$
10:     pred_prev_image ← mask ·pred_prev_image + $(1 - \text{mask}) \cdot$ forward(original, t)
11:     pred_prev_image ← add_variance(predicted_variance, t_prev, pred_prev_image)
12: end for
13: masked_clean ← image.cpu().detach().numpy()
14: return masked_clean
```



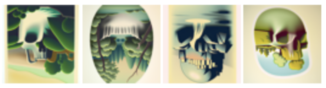
3.4 Visual anagrams

Using this algorithmic framework, we can create visual anagrams and create optical illusions, which are images that look different right-side-up and flipped upside-down. This is a matter of retuning the way we generate the final noise estimate. Fix an image x_t. We denoise x_t with the first prompt embedding, denoise a flipped x_t with the second, average the first noise and the flipped second noise, and add the predicted variance to form the final noise estimate. Here, we used the first prompt embedding "an oil painting of an old man" and the second "an oil painting of people around a campfire". **Note:** Because we are using the model for tasks it was not trained for, we need to run it a few times to get a good result.



3.5 Hybrid images

Using this same idea, we can implement factorized diffusion and create hybrid images. We will first take the noise estimates using the two prompts and create a final estimate by composing the low-pass of the first estimate and the high-pass of the second estimate using Gaussian blurring. Here, we used the first prompt embedding "a lithograph of a skull" and the second "a lithograph of waterfalls". This should give images that look like a skull from far away but a waterfall from close up (not sure why it is upside down).



4 Challenges

Some core challenges I faced was learning about the diffusion architecture itself and the math behind the training and inference process. These were not strictly required, but I still wanted to learn these anyway because this is the reason I chose to do this project. Because we are predicting the "backwards" state transitions, it was quite counterintuitive. Overall, creating the noise estimates seemed to be the most difficult, and testing the visual anagrams and hybrid images was hard because they do not consistently generate good images (expected due to OOD tasks).

5 Ethical issues in computational photography

5.1 Problems with current task

Implicitly, this task implies that there is a set standard, "ideal" definition, or a set of features that is characteristic of attractiveness or beauty in the first place. Therefore, if we were to carry out this task, then we would identify these definitions or features (either ideaized by the research team or human survey for a classical system, or generated with enough training data for a deep learning one). Either way, it would create a standard, which does not conform to my ethics. Attractiveness is a subjective concept, which is influenced by personal preference, cultural values, and societal norms. Therefore, attempting to standardize it would create an exclusionary society. If such a definition were to publicly be released, this could cause further discrimination to those who deviate from such a standard, and divide within our communities. The changes created by this system would demote genuinity and as a corollary, self-confidence. There is a parallel that can be drawn with social media. Due to influence and popularity of social media, where people show only the best sides of themselves, people are hesitant to express themselves now, even offline. For instance, the issue is already terrible with skin blemishes such as wrinkles or acne. If Meta were to, say, introduce warping on Instagram (an example of such a proposed system) for people to manipulate their face and body shape, that would be quite dangerous.

5.2 Rescoping the task

The main issue here is the definition of "attractive". To avoid ethical problems, I would rescope the project to focus on creating a system that enhances image quality rather than attractiveness. This would be done through introducing non-biased and universally-applicable adjustments like improving lighting, color balance, and resolution, rather than altering physical features like what the original task intended. This way, we prioritize improving image quality over subjective aesthetic judgments (e.g. skin color and composition, body shape), and avoid reinforcing harmful beauty standards. Moreover, we could also enforce transparency in this system by informing users about the types of enhancements applied and allow them to opt out or customize edits to align with their preferences. With this rescoped project, we do not exclude any groups or create or conform to any one specific beauty standard, which would be inherently created (intentionally or not) in developing the system for the task before. In this new system, we also aim to build tools for creative self-expression and empowerment rather than altering perceived attractiveness. For instance, we could develop customizable filters that convey artistic styles, allowing users to personalize their images based on their mood. By focusing on improving image quality, personal creativity, and inclusivity instead of creating and reinforcing this one standard definition of "attractive", we align with ethical principles while still delivering value to users.