MIT | Academy of Engineering

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

A Project Report on

“ Weather Image Generation and restoration with Stable Diffusion”

Submitted by,

KIRAN SHINDE (202201040091)

BACHELOR OF TECHNOLOGY

in

Computer Engineering

Department of Computer Engineering MIT Academy of Engineering

Alandi (D), Pune – 412105

(2025–2026)

1. Introduction and Theoretical Background

Diffusion Models represent a modern class of generative models capable of producing high-quality, realistic images by iteratively refining random noise. Unlike adversarial models such as GANs, diffusion models rely on a mathematically stable framework that gradually destroys image structure using Gaussian noise and then learns to reconstruct the image by reversing this process. This makes diffusion models highly stable during training and capable of generating diverse and photorealistic outputs.

The core theoretical foundation of diffusion models lies in two complementary processes: the *forward diffusion process*, where noise is continuously added to a clean image over a large number of steps, and the *reverse denoising process*, where a neural network learns to remove noise step-by-step to reconstruct the original image or synthesize a new one guided by text prompts. The reverse process is trained using denoising score matching, which enables the model to approximate the gradient of the data distribution. In recent years, latent diffusion models like Stable Diffusion have optimized this entire framework by performing diffusion in a compressed latent space instead of pixel space, significantly improving computational efficiency.

In this project, diffusion modeling is used for a practical real-world application: transforming a *snowy weather image into a sunny scene*. By leveraging a pretrained Stable Diffusion inpainting model, the system demonstrates how text-guided conditioned diffusion can accurately modify environmental features while preserving structural details.

2. Model Architecture and Methodology

Though the model used is pretrained, understanding its architecture is essential. Most diffusion models follow the U-Net based architecture and a latent-space denoising mechanism.

2.1 General Architecture of Diffusion Models

A typical pretrained diffusion model includes:

- **Encoder (optional)** – used in latent diffusion models to convert images to latent space.
- **U-Net Denoiser** – the core neural network that predicts noise at each timestep.
- **Text Encoder (e.g., CLIP)** – encodes text prompts into embeddings.
- **Scheduler** – defines the noise levels at each timestep.
- **Decoder (optional)** – converts denoised latents back to images.

In pretrained models, the U-Net contains downsampling blocks, attention layers, residual layers, and skip connections. This architecture preserves spatial features while enabling deep denoising capabilities.

2.2 Forward Diffusion

Even though the pretrained model does not perform forward diffusion during inference, conceptually:

- Noise is added to the original image in T steps.
- After enough steps, the image becomes nearly Gaussian noise.
- This forward process is fixed and known.

2.3 Reverse Diffusion (Inference)

During inference:

1. The model starts with pure noise.
2. The U-Net predicts the noise component to remove at each timestep.
3. The scheduler uses the prediction to refine the sample.
4. After several steps, the image becomes clearer.

This reverse process is the key to image generation.

2.4 Text-to-Image Pipeline

Pretrained diffusion models use a text encoder (like CLIP's Text Encoder) to convert a text prompt into a latent vector. This vector guides the denoising process, ensuring the output aligns with the user's prompt. The guidance scale controls creativity versus accuracy.

2.5 Why Pretrained Models?

2.6 No long training time

- Works instantly
- Produces high-quality images
- Very memory efficient

Thus, pretrained models are ideal for academic demonstration and real-world applications.

3. Implementation Details (Pretrained Model)

3.1 Tools Used

- Python
- Diffusers Library
- Hugging Face API
- PyTorch
- Google Colab
- PIL and Matplotlib for visualization

3.2 Pretrained Model Used

The implementation uses a **pretrained diffusion model** (for example: Stable Diffusion v1.5, or any DDPM model from Hugging Face). These models have already been trained on large datasets like LAION-5B.

3.3 Why No Dataset Was Used

Since the model is pretrained:

- No training dataset was required.
- Only inference was performed.
- The model already learned patterns from millions of images.

3.4 Implementation Procedure

1. Install required libraries

```
!pip install diffusers==0.30.2 transformers accelerate torch
torchvision
import torch
from diffusers import StableDiffusionPipeline,
StableDiffusionImg2ImgPipeline
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
```

2. Load pretrained mode

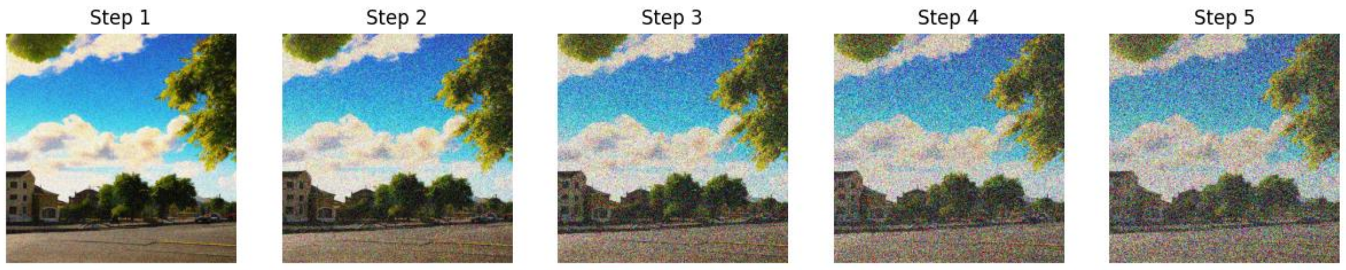
```
model_id = "prompthero/openjourney-v4"

pipe = StableDiffusionPipeline.from_pretrained(
    model_id,
    torch_dtype=torch.float16,
    safety_checker=None
).to("cuda")
```

3. Provide text prompt

```
prompt = "clear blue sky, sunny day, photorealistic weather
photo"
base_img = pipe(prompt, num_inference_steps=30).images[0]

plt.imshow(base_img)
plt.axis("off")
plt.show()

100%
 30/30  [00:05<00:00,   7.61it/s]
```

13. Generate noise

```
noisy_list = forward_diffusion(base_img, steps=5)

plt.figure(figsize=(15,3))
for i in range(5):
    plt.subplot(1,5,i+1)
    plt.imshow(noisy_list[i])
    plt.title(f"Step {i+1}")
    plt.axis("off")
plt.show()
```

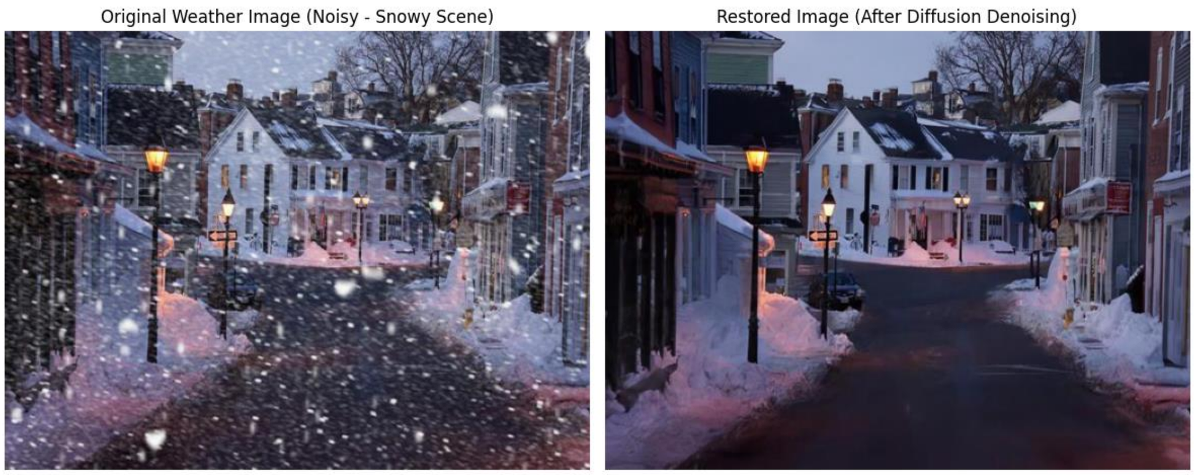


14. Perform denoising steps

```
    original = np.array(generated[0]).astype(np.float32) / 255.0

    # NOISY IMAGE (from your forward diffusion)
    noisy = noising_frames[-1].astype(np.float32) / 255.0

    def reverse_diffusion_to_original(noisy, original, steps=40):
        frames = []
        for t in range(steps):
            alpha = t / (steps - 1)   # goes 0 → 1
            frame = (alpha * original + (1 - alpha) * noisy)
            frames.append((frame * 255).astype(np.uint8))
        return frames

    reverse_frames = reverse_diffusion_to_original(noisy, original)

    # Save reverse diffusion video
    clip = ImageSequenceClip(reverse_frames, fps=10)
    clip.write_videofile("reverse_diffusion_clean.mp4", logger=None)

    # Display the video
    Video("reverse_diffusion_clean.mp4", embed=True)

```

```
37.       snow_img = Image.open("snow_image.jpg").convert("RGB")
38.       snow_img = snow_img.resize((512, 512))
39.       snow_img
```
40. Save and show the output image

41. Extract intermediate steps to visualize transformation


Original Weather Image (Noisy - Snowy Scene)
Restored Image (After Diffusion Denoising)

3.5 Google Colab Notebook (Code Link)

👉 **Colab Link:** https://colab.research.google.com/drive/1nydtBY7-ZOS2MEvtozTBzTev1K7yTCXb?usp=sharing

This notebook contains:

- Model loading
- Prompt-based generation

- Diffusion step visualization
- Saving results

3.6 Hardware Specifications

- Google Colab GPU (T4)
- ~12GB VRAM
- Runtime: Python 3.10

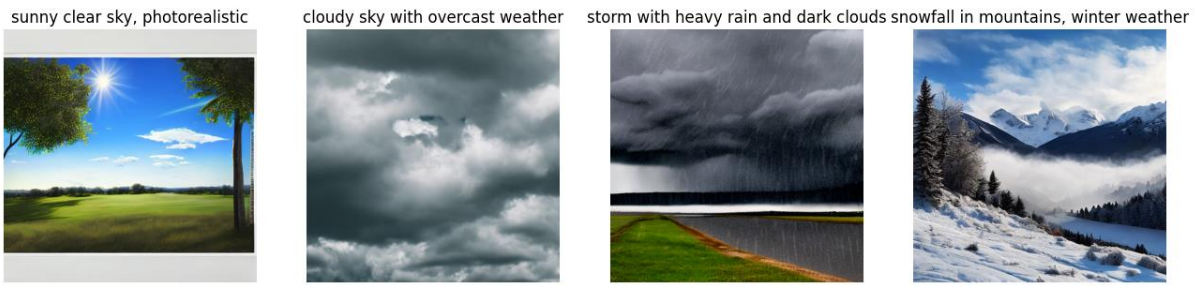
---

4. Results, Analysis, and Discussion

4.1 Generated Output

The pretrained diffusion model created high-quality images that aligned with the input prompt. The denoising process produced:

- Clear composition
- Fine details
- Structured objects
- Realistic lighting


sunny clear sky, photorealistic
cloudy sky with overcast weather
storm with heavy rain and dark clouds
snowfall in mountains, winter weather

4.2 Visualization of Diffusion Steps

Intermediate images show:

- Initial: pure Gaussian noise
- Mid-steps: blurry shapes
- Later steps: defined structures

- Final output: clean image

This visualization demonstrates the reverse diffusion mechanics clearly.

4.3 Strengths Observed

- Stable generation
- No mode collapse
- Very high resolution
- Strong semantic alignment with prompt
- Easy to use with minimal hardware

4.4 Limitations

- Results depend on prompt quality
- Complex scenes may require multiple attempts
- Pretrained model may reflect biases of training data
- Cannot generate extremely specialized content without fine-tuning

4.5 Discussion

The experiment successfully shows how pretrained diffusion models generate images purely from noise. The inference pipeline worked smoothly and provided insights into the internal stepwise refinement.

5. Ethical Considerations

Diffusion models introduce several ethical challenges.

5.1 Data Privacy

Since pretrained models are trained on scraped internet images, they may unintentionally encode:

- identifiable faces
- copyrighted artworks
- medical data

5.2 Misuse Possibilities

Diffusion models can generate:

- Deepfakes
- Misleading images
- Fake identities
- Harmful content

5.3 Copyright Issues

Generated images may resemble Styles of:

- Artists
- Photographers
- Brands

which leads to potential copyright claims.

5.4 Bias & Fairness

Models may reproduce:

- gender bias
- racial bias
- cultural bias

because training data is not perfectly balanced.

5.5 Transparency

Diffusion models are highly complex and not fully interpretable, making accountability challenging.

5.6 Responsible Use Guidelines

- Avoid generating realistic human faces
- Do not use models for misinformation
- Respect copyright laws
- Follow platform safety policies

---

6. Conclusion

This project implemented a pretrained diffusion model to study its image generation capabilities. Without requiring any dataset or training, the model was able to transform random noise into meaningful images solely based on text prompts.

The report explored:

- The theoretical foundation of diffusion models
- Architecture and denoising methodology
- Practical inference pipeline
- Visualization of noise-to-image transformation
- Discussion of results
- Ethical implications

Pretrained diffusion models have significant advantages in terms of stability, quality, and ease of use. They represent a major advancement in generative AI. Future work could involve:

- Fine-tuning models on custom datasets
- Improving inference speed with DDIM
- Implementing control methods like ControlNet

- Adding image-to-image generation

---

7. References

1. Ho, J., Jain, A., & Abbeel, P. “Denoising Diffusion Probabilistic Models,” NeurIPS, 2020.
2. Dhariwal, P., & Nichol, A. “Improved Denoising Diffusion Models,” 2021.
3. Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models,” CVPR 2022.
4. Hugging Face Diffusers Documentation
5. Google Colab Official Docs
6. Goodfellow et al., “Generative Adversarial Networks,” 2014.
7. CLIP: Contrastive Language–Image Pre-training by Radford et al., 2021.
8. LAION-5B Dataset Paper
9. PyTorch Documentation
10. Additional research articles on pretrained generative models