	Indian Institute of Information Technology Una Himachal Pradesh (An Institute of National Importance under MoE) **Saloh, Una -177209** www.iiitu.ac.in

Date: 11 Sep. 25

Review I
Project Phase – I (CSL503/ITL503)

Student Name	**Kush Gupta, Kavin Moudgil, Mirat Parmar**	**Roll No.**	23134, 23329, 23345
Batch No.	B23	**Semester**	V
Branch	CSE, IT, IT	**Supervisor(s)**	Dr. Shivdutt

1. **Title of the Project -** Text-to-Image Generation using Stable Diffusion

2. Introduction

Stable Diffusion is a state-of-the-art text-to-image generative model introduced in 2022, based on latent diffusion models. Unlike pixel-space diffusion models, it operates in a compressed latent space using a Variational Autoencoder (VAE), making the generation process computationally efficient. The goal of our project is to understand the mathematical foundations of diffusion probabilistic models and implement Stable Diffusion from scratch using PyTorch, focusing on the forward (noise addition) and reverse (denoising) processes, U-Net backbone, and text conditioning via CLIP embeddings.

3. Problem Definition

Generative models such as GANs and VAEs have limitations in stability and sample diversity. Diffusion models address these issues but are computationally expensive in pixel space. Stable Diffusion solves this by working in the latent space. Our challenge is to:

- Implement diffusion probabilistic modelling.
- Efficiently encode and decode high-dimensional image data.
- Integrate text conditioning for text-to-image synthesis

4. Objectives

1. Implement forward and reverse diffusion processes as defined in DDPM.
2. Build and train a U-Net backbone with attention mechanisms for denoising.
3. Incorporate CLIP-based text embeddings for conditioning.
4. Support text-to-image, image-to-image, and inpainting tasks.
5. Evaluate results using standard metrics such as FID and Inception Score.

5. Skillset additionally required to solve/address the problem

1. Mathematics: Probability, Gaussian distributions, KL divergence, ELBO.
2. Deep Learning: PyTorch, CNNs, Transformers, Attention mechanism.
3. Generative Models: Autoencoders, VAEs, Diffusion models.
4. Implementation: Efficient training loops, GPU optimization.

6. Timeline to achieve the skillset – 4 months

7. Block schematic

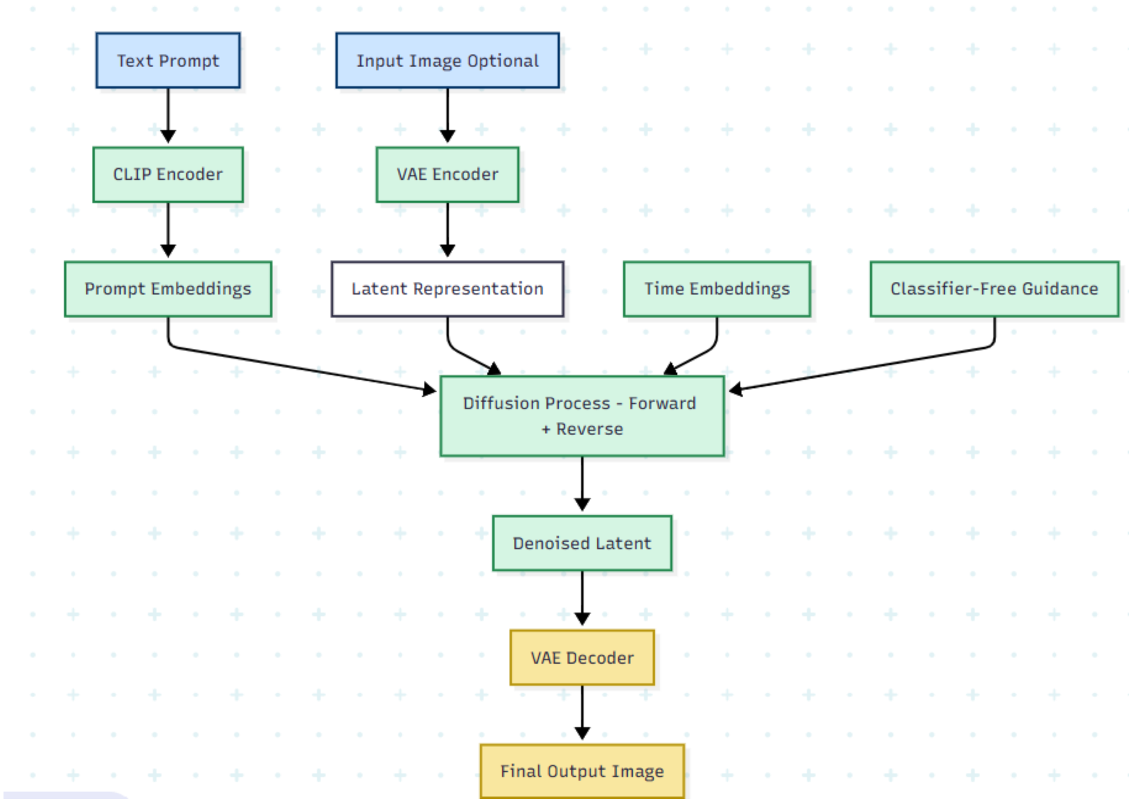

Figure 1: Block schematic of Stable Diffusion for text-to-image generation.

8. Weekly milestones

Week	Major Activities to be Completed
Week 1	**Literature Survey and Project Setup** • Review research papers on Diffusion Models (DDPM) and Stable Diffusion. • Study related works on text-to-image generation, CLIP, and Variational Autoencoders. • Set up development environment (Python 3.11.3, PyTorch, VS Code, Git).

Week 2	**Understanding Stable Diffusion** • Implement forward diffusion (noise addition process). • Visualize noise schedule across timesteps. • Experiment with different β-schedules (linear, cosine).
Week 3	**Reverse Process Basics** • Implement sampling step for reverse diffusion. • Test denoising function with simple datasets (e.g., MNIST). • Validate noise prediction using small UNet prototype.
Week 4	**Building the UNet Architecture** • Design UNet with convolutional + attention layers. • Implement residual blocks, group normalization, and skip connections. • Train/test UNet on toy data for denoising tasks.
Week 5	**Time Embeddings & Conditioning Mechanisms** • Implement sinusoidal time-step embeddings. • Study and implement classifier-free guidance. • Experiment with conditional vs. unconditional generation.
Week 6	**Variational Autoencoder (VAE) Integration** • Build VAE encoder/decoder for latent compression. • Train VAE on image dataset (e.g., CelebA). • Verify reconstruction quality from latent space.
Week 7	**CLIP Text Embedding Integration** • Integrate HuggingFace Transformers for text embeddings. • Align text embeddings with UNet conditioning. • Run first trials of text-to-image generation (low resolution).
Week 8	**Latent Diffusion Implementation** • Implement latent-space forward and reverse diffusion. • Test speed improvements compared to pixel-space diffusion. • Validate generated samples visually.

Week 9	**Training Pipeline Setup** • Prepare dataset (captioned images). • Implement training loop with PyTorch Lightning. • Add tqdm logging, checkpointing, and visualization.
Week 10	**Initial Model Training** • Train UNet + VAE + text conditioning jointly. • Generate first text-to-image samples. • Debug training instabilities and losses.
Week 11	**Advanced Features** • Implement image-to-image generation pipeline. • Implement inpainting (mask + diffusion). • Experiment with guidance scale hyperparameters.
Week 12	**Hyperparameter Tuning** • Tune learning rates, β-schedules, optimizer settings. • Improve training stability with gradient clipping. • Compare performance with/without attention layers.
Week 13	**Evaluation Metrics** • Implement FID and Inception Score evaluation. • Run ablations (ε-prediction vs. μ-prediction). • Compare sample quality with baseline DDPM.
Week 14	**Optimization & Acceleration** • Implement mixed precision training. • Add efficient schedulers (Euler, DDIM). • Reduce sampling steps for faster inference.
Week 15	**System Integration** • Build modular pipeline (text encoder, UNet, VAE, scheduler). • Create APIs for text-to-image generation. • Test across multiple prompts and seed variations.

Week 16	**Documentation and Final Presentation** • Compile documentation (architecture diagrams, results, code). • Create result graphs and sample generations. • Final presentation: showcase text-to-image, img2img, and inpainting. • Outline future enhancements (larger datasets, distributed training).

9. Completed Milestones -

Week	Major Activities Completed
1)	**Literature Survey and Project Setup** • Review research papers on Diffusion Models (DDPM) and Stable Diffusion. • Study related works on text-to-image generation, CLIP, and Variational Autoencoders. • Set up development environment (Python 3.11.3, PyTorch, VS Code, Git).
2)	**Understanding Stable Diffusion** • Implement forward diffusion (noise addition process). • Visualize noise schedule across timesteps. • Experiment with different β-schedules (linear, cosine).
3)	**Reverse Process Basics** • Implement sampling step for reverse diffusion. • Test denoising function with simple datasets (e.g., MNIST). • Validate noise prediction using small UNet prototype.
4)	**Building the UNet Architecture** • Design UNet with convolutional + attention layers. • Implement residual blocks, group normalization, and skip connections. • Train/test UNet on toy data for denoising tasks.

10. Milestones to be Completed –

1)	**Time Embeddings & Conditioning Mechanisms** • Implement sinusoidal time-step embeddings. • Study and implement classifier-free guidance. • Experiment with conditional vs. unconditional generation.
2)	**Variational Autoencoder (VAE) Integration** • Build VAE encoder/decoder for latent compression. • Train VAE on image dataset (e.g., CelebA). • Verify reconstruction quality from latent space.
3)	**CLIP Text Embedding Integration** • Integrate HuggingFace Transformers for text embeddings. • Align text embeddings with UNet conditioning. • Run first trials of text-to-image generation (low resolution).
4)	**Latent Diffusion Implementation** • Implement latent-space forward and reverse diffusion. • Test speed improvements compared to pixel-space diffusion. • Validate generated samples visually.
5)	**Training Pipeline Setup** • Prepare dataset (captioned images). • Implement training loop with PyTorch Lightning. • Add tqdm logging, checkpointing, and visualization.
6)	**Initial Model Training** • Train UNet + VAE + text conditioning jointly. • Generate first text-to-image samples. • Debug training instabilities and losses.
7)	**Advanced Features** • Implement image-to-image generation pipeline. • Implement inpainting (mask + diffusion). • Experiment with guidance scale hyperparameters.

8)	**Hyperparameter Tuning** • Tune learning rates, β-schedules, optimizer settings. • Improve training stability with gradient clipping. • Compare performance with/without attention layers.
9)	**Evaluation Metrics** • Implement FID and Inception Score evaluation. • Run ablations (ε-prediction vs. μ-prediction). • Compare sample quality with baseline DDPM.
10)	**Optimization & Acceleration** • Implement mixed precision training. • Add efficient schedulers (Euler, DDIM). • Reduce sampling steps for faster inference.
11)	**System Integration** • Build modular pipeline (text encoder, UNet, VAE, scheduler). • Create APIs for text-to-image generation. • Test across multiple prompts and seed variations.
12)	**Documentation and Final Presentation** • Compile documentation (architecture diagrams, results, code). • Create result graphs and sample generations. • Final presentation: showcase text-to-image, img2img, and inpainting. • Outline future enhancements (larger datasets, distributed training).

11. Expected Challenges –

1. High Computational Requirements – Training Stable Diffusion requires significant GPU resources and memory, which may limit dataset size, resolution, and training iterations.
2. Training Instability – Diffusion models are sensitive to hyperparameters; improper noise scheduling, learning rates, or optimizer settings may cause divergence.
3. Integration Complexity – Combining VAE, UNet, and CLIP embeddings into a single pipeline may lead to implementation bugs and compatibility issues.
4. Data Preparation – Curating and preprocessing a sufficiently large captioned dataset for text-to-image training can be time-consuming.

5. Slow Sampling Process – Reverse diffusion involves many iterative steps, leading to high inference time if not optimized.
6. Evaluation Difficulty – Quantitatively evaluating generated images (FID, Inception Score) requires careful benchmarking and may not fully reflect perceptual quality.
7. Overfitting on Small Datasets – Training on limited data may reduce generalization, leading to poor text-to-image results on unseen prompts.
8. Alignment of Text and Image – Ensuring the generated image correctly represents the given prompt (semantic alignment) is a challenging task.
9. Model Optimization – Implementing mixed precision training, gradient clipping, and efficient schedulers (DDIM, Euler) without introducing instability.

12. References –

1. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). *High-Resolution Image Synthesis with Latent Diffusion Models.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).
2. **Tokenizer Files (vocab.json, merges.txt):** https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5/tree/main/tokenizer
3. **Stable Diffusion v1.5 Weights (ckpt):** https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5/tree/main
4. **Fine-tuned Models:**
   - InkPunk Diffusion: https://huggingface.co/Envvi/InkpunkDiffusion/tree/main
   - Illustration Diffusion (Hollie Mengert): https://huggingface.co/ogkalu/Illustration-Diffusion/tree/main

Name and Signature of Student **Name and Signature of Supervisor**