Cyshield AI Assessment - GenCV003 Task

Ahmed Tamer Samir

Submitted to Cyshield Computer Vision Team

December 11, 2025

Contents

List of Figures

1 DDPM

1.1 Forward Diffusion Process

The forward process gradually adds Gaussian noise to the data over T timesteps. We can sample noisy images directly from the original image using:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{1}$$

where x_0 is the original image, x_t is the noisy image at timestep t, and $\bar{\alpha}_t$ controls the noise level.

1.1.1 Cosine Variance Schedule

The noise schedule follows a cosine-based formulation:

$$f(t) = \cos\left(\frac{t/T + s}{1 + s} \cdot \frac{\pi}{2}\right)^2 \tag{2}$$

$$\alpha_t = \frac{f(t)}{f(t-1)}, \qquad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \tag{3}$$

where $s = 0.008$ is an offset parameter that prevents β_t from reaching zero near $t = 0$, $\beta_t = 1 - \alpha_t$, and $T = 4000$ is used as the total number of diffusion steps in this implementation.

The cosine variance scheduler is preferred over linear scheduling because it introduces noise more smoothly across timesteps. Linear schedules tend to destroy image structure too aggressively at early steps, leaving insufficient signal for the model to learn effectively. In contrast, the cosine schedule, as defined in Equation (2), preserves more image information in the early phase while still ensuring full noise corruption by the final timestep. This leads to more stable training, higher sample quality, and improved FID scores.

1.2 Dataset Preparation

The FashionMNIST dataset, consisting of 28×28 grayscale images, was utilized for training. A key preprocessing step involves normalizing pixel values to the interval $[-1, 1]$ rather than the conventional $[0, 1]$ range. This normalization choice is motivated by several factors:

1. **Zero-Centered Noise**: Gaussian noise has zero mean. Zero-centered data ($[-1, 1]$) maintains distributional symmetry when noise is added.

2. **Convergence to Standard Normal**: The forward process $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$ converges to $\mathcal{N}(0, \mathbf{I})$ as $t \to T$ only when $\mathbf{x}_0$ is zero-centered.

1.3 U-Net Architecture

A custom U-Net architecture was implemented with the following key components:

- **Time Embedding**: Sinusoidal positional encodings transform timestep t into a 64-dimensional vector, which is injected at multiple scales throughout the network via dense layers and addition operations.

- **Encoder Path**: Three downsampling blocks with channel dimensions $\{32, 64, 128\}$ using SeparableConv2D layers, batch normalization, and max pooling. Residual connections are added via 1×1 convolutions.

- **Decoder Path**: Three upsampling blocks mirroring the encoder with channel dimensions $\{64, 32, 16\}$ using Conv2DTranspose layers and bilinear upsampling.

- **Skip Connections**: Cross-skip connections concatenate encoder features to corresponding decoder layers, while additive skip connections are used within encoder/decoder blocks.

- **Output Layer**: A final 3×3 convolution produces single-channel noise predictions matching the input dimensions $28 \times 28 \times 1$.

The model was trained using Huber loss and the Nadam optimizer for 100 epochs with batch size=32

Important Note: I intentionally avoided adding deeper layers since Fashion-MNIST is a simple, low-resolution dataset, and a lightweight U-Net is sufficient to model it effectively.

1.4 Reverse Process

Initialize noise: $x_T \sim \mathcal{N}(0, I)$

For each timestep $t = T, T-1, \ldots, 1$:

1) Predict the noise in the current image: $\hat{\epsilon}_t = \epsilon_\theta(x_t, t)$

2) Compute the denoised mean: $\mu_t = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}_t\Big)$

3) Sample the next image: $x_{t-1} = \mu_t + \sigma_t z, \quad z \sim \mathcal{N}(0, I)$ (if $t > 1$)

Output final generated image: x_0

1.5 Conditional DDPM

In addition to the baseline DDPM, a class-conditional variant was implemented to enable controlled generation of specific Fashion-MNIST categories. This extension allows the model to generate images from a desired class on demand, rather than producing random samples.

The conditional DDPM requires three key architectural modifications to the baseline implementation:

1.5.1 Model Modifications

1. Data Preparation Enhancement:

The training pipeline was extended to include class labels alongside the noisy images and timestep information. The modified input tuple becomes (x_t, t, c) where c represents the class label.

2. Class Embedding Integration:

A learnable embedding layer maps discrete class labels to continuous vector representations:

$$\text{class\_emb} = \text{Embedding}(c), \quad c \in \{0, 1, ..., 9\} \tag{4}$$

This class embedding is combined with the time encoding to create a unified conditioning signal:

$$\text{combined\_enc} = \text{time\_enc}(t) + \text{class\_emb}(c) \tag{5}$$

The combined encoding is then injected into the U-Net at each resolution level, allowing the model to condition its noise predictions on both timestep and class information.

3. Conditional Sampling:

During inference, users specify the desired class label c and batch size. The generation process then produces images belonging exclusively to the specified category. The sampling algorithm remains identical to the unconditional case, except the class label is provided as an additional input to the U-Net at each denoising step.

1.5.2 Generated Examples

Figure 1: Class-conditional generation example 1.

Figure 2: Class-conditional generation example 2.

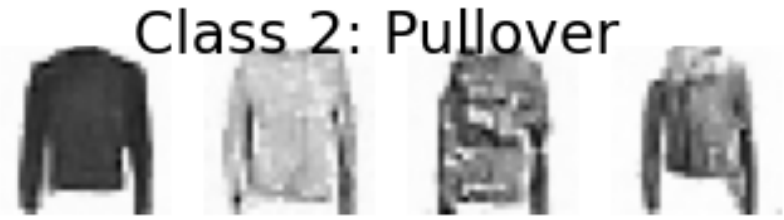

Figure 3: Class-conditional generation example 3.

Comment: Figures 1, 2, and 3 demonstrate the model's ability to generate class-specific Fashion-MNIST images across all ten categories: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot.

2 VAE

.

2.0.1 Dataset and Configuration

- **Dataset**: Fashion MNIST (28×28 grayscale images of clothing items)
- **Latent Space Dimension**: $d_z = 4$ (compact representation)
- **Training Size**: 55,000 images
- **Validation Size**: 5,000 images
- **Test Size**: 10,000 images

2.1 VAE Architecture Diagram

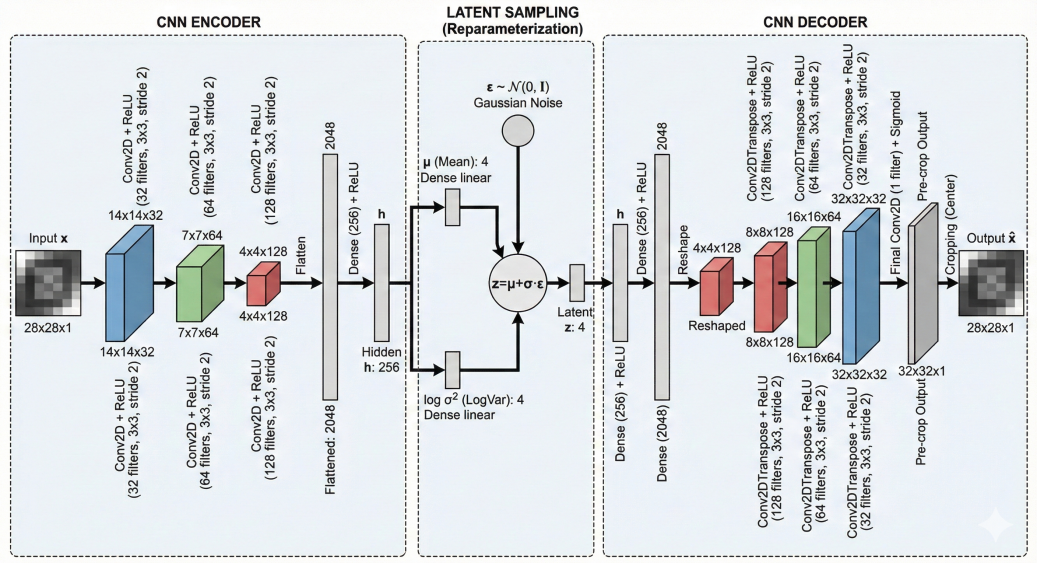

Figure 4: Complete VAE architecture showing the CNN encoder, latent sampling with reparameterization trick, and CNN decoder. The encoder progressively compresses the input through convolutional layers, while the decoder reconstructs the image through transposed convolutions.

I preferred using CNN-based encoder and decoder architectures instead of MLPs because convolutional layers naturally preserve the spatial structure of images. In contrast, MLPs require flattening the input, which discards important local relationships and makes it harder to learn meaningful visual patterns. CNNs, on the other hand, capture spatial correlations such as edges, textures, and shapes, allowing the VAE to form a richer latent representation and generate higher-quality reconstructions. Even though Fashion-MNIST is a relatively small and simple dataset, using CNNs remains beneficial, as they consistently produce more realistic outputs by leveraging spatial information that MLPs cannot retain.

2.2 Loss Function

2.2.1 Reconstruction Loss

This term measures how close the reconstructed image $\hat{\mathbf{x}}$ is to the original input $\mathbf{x}$, encouraging the decoder to accurately rebuild the data.

$$\mathcal{L}_{\text{recon}} = 1000 \cdot E_{q(\mathbf{z}|\mathbf{x})} \left[\|\mathbf{x} - \hat{\mathbf{x}}\|^2 \right] \tag{6}$$

2.2.2 KL Divergence

The KL divergence regularizes the latent space by pushing $q(\mathbf{z}|\mathbf{x})$ toward a standard normal distribution, ensuring smoothness and continuity in the latent representation.

$$\mathcal{L}_{\text{KL}} = \frac{1}{28 \times 28} E_{\mathbf{x}} \left[-\frac{1}{2} \sum_{i=1}^{d_z} \left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2 \right) \right] \tag{7}$$

2.2.3 Total Loss

The total loss combines both reconstruction accuracy and latent space regularization to train the VAE effectively.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{KL}} \tag{8}$$

2.3 Training Configuration

Parameter	**Value**
Optimizer	Adam
Learning Rate	1×10^{-4}
Epochs	50
Batch Size	128

Table 1: Training hyperparameters

3 Quantitative Analysis

We generated 10,000 samples from both DDPM and VAE models and computed evaluation metrics by comparing them against the test dataset.

Important Note: When Performing Quantitative Analysis on DDPM, I built a variant of 1000 steps (as the main DDPM i implemented had 4000 steps, but i decrease it only in this experiment to speed up the Generation process of samples for Metrics Computation)

3.1 Evaluation Metrics

Fréchet Inception Distance (FID): FID quantifies the similarity between generated and real images by comparing their feature distributions. Lower scores indicate higher realism and closer resemblance to real images.

Inception Score (IS): IS assesses both the quality and diversity of generated images, rewarding confident class predictions and varied samples. Higher scores indicate better image quality and greater diversity.

3.2 Results

Table 2: Quantitative Evaluation Metrics

Model	**FID Score (↓)**	**Inception Score (↑)**
VAE	141.67	2.06 ± 0.02
DDPM	82.03	1.92 ± 0.01

Comment on Results: The DDPM achieves substantially better sample quality with an FID score of 82.03 versus VAE's 141.67 (42% improvement), demonstrating DDPM-generated images better match the statistical distribution of real images through its iterative $T = 1000$ denoising process without VAE's latent bottleneck. However, both models exhibit comparably low Inception Scores ($\sim$1.9–2.0), indicating limited inter-class diversity. This suggests that while DDPM produces more realistic samples, neither architecture achieves strong class separation, likely due to Fashion-MNIST's fine-grained categories and domain mismatch with the ImageNet-pretrained Inception classifier.

4 Qualitative Analysis

(a) VAE Generated Samples (b) DDPM Generated Samples

Figure 5: Comparison of generated fashion items

DDPM (Figure 5b): Produces sharp and clear images with well-defined details. The clothing items look realistic with visible textures and features like collars, pockets, and patterns. Different types of clothing (jackets, pants, shoes) are easily distinguishable and look like real products.

VAE (Figure 5a): Generates blurry and unclear images. The clothing items look like fuzzy shadows with soft edges. While we can still recognize the basic shapes (shirts, pants, bags), the fine details are lost and the images lack sharpness.

Conclusion: DDPM produces much better quality images than VAE with sharper details and more realistic appearance. However, this comes with a trade-off: DDPM requires significantly more computational resources for training and takes much longer to generate images compared to VAE's faster single-pass approach.