Continuous & Discrete Denoising Diffusion Probabilistic Models (D3PM & DDPM) for MNIST Generation

Avdhoot Golekar (23B0060), Geet Sethi (23B2258), Panav Shah (23B3323)

September 23, 2025

1 Discrete Denoising Diffusion Probabilistic Model (D3PM)

1.1 Model Architecture

1.1.1 UNet Backbone

The core architecture is based on a U-Net with residual blocks and time embeddings:

- **Input/Output**: Single channel 28×28 images with 10 discrete classes (0-8 for pixel values, 9 for absorbing state)
- **Time Embedding**: Sinusoidal position embeddings of dimension 128
- **Downsampling Path**: Three residual blocks ($64 \to 128 \to 256$ channels) with 2×2 max pooling
- **Bottleneck**: $256 \to 512$ channel residual block
- **Upsampling Path**: Three residual blocks with skip connections and transposed convolutions
- **Output Layer**: 1×1 convolution producing 10-channel logits

1.1.2 Residual Block Design

Each residual block incorporates:

- Two 3×3 convolutions with group normalization
- Time embedding injection via learned linear transformation
- ReLU activation and dropout (0.1)
- Skip connection with 1×1 convolution for dimension matching

1.1.3 Conditional Architecture

The conditional variant extends the UNet with:

- Class embedding layer: 10 classes $\to$ 128 dimensions
- Class embedding addition to time embeddings
- Forward pass includes class labels as additional conditioning

1.2 Training Procedure

1.2.1 Data Preprocessing

MNIST images are preprocessed as follows:

1. Normalize to $[0, 1]$ range using `transforms.ToTensor()`
2. Discretize to 9 discrete values: $x_{discrete} = \text{round}(x \times 8)$
3. Values 0-8 represent pixel intensities, 9 is the absorbing state

1.2.2 Loss Function

The model uses cross-entropy loss between predicted and true discrete pixel values:

$$\mathcal{L} = \text{CrossEntropy}(\text{logits}_{flat}, \text{targets}_{flat}) \tag{1}$$

1.2.3 Noise Scheduling

We use the following noise scheduling methods to choose the mask probabilities for each of the input tokens:

1.2.4 Linear Schedule

$$\text{mask\_prob}(t) = \frac{t}{T-1} \tag{2}$$

where T is the total number of timesteps.

1.2.5 Cosine Schedule

$$\alpha_t = \cos\left(\frac{\pi}{2} \cdot \frac{t+s}{T+s}\right)^2 \tag{3}$$

where $s = 0.008$ for numerical stability.

1.2.6 Training Algorithm

Algorithm 1 D3PM Training Loop

Initialize model θ, optimizer, scheduler
for epoch = 1 to max_epochs **do**
  for batch in dataloader **do**
    Sample timesteps $t \sim \text{Uniform}(0, T)$
    Add noise: $x_t = \text{scheduler.add\_noise}(x_0, t, \text{num\_classes})$
    Forward pass: $\hat{x}_0 = \text{model}(x_t, t, \text{class\_label})$
    Compute loss: $\mathcal{L} = \text{CrossEntropy}(\hat{x}_0, x_0)$
    Backward pass and update θ
  end for
end for

1.3 Sampling Procedure

Algorithm 2 D3PM Sampling

1: Initialize x_T with absorbing states (value 9)
2: **for** $t = T - 1$ down to 0 **do**
3: Get logits: logits = model($x_t, t,$ class_label)
4: Convert to probabilities: $p = \text{softmax}(\text{logits})$
5: Sample: $x_{t-1} \sim \text{Categorical}(p)$
6: **end for**
7: Convert to continuous: $x_{final} = x_0/8$

1.4 Experimental Setup

1.4.1 Hyperparameters

All experiments use the following base configuration:

- **Epochs**: 300
- **Batch Size**: 64
- **Learning Rate**: 0.0001
- **Optimizer**: Adam
- **Evaluation Samples**: 1000
- **FID Batches**: 5 (64 samples each)

1.4.2 Experimental Variables

We vary the following parameters:

- **Diffusion Steps**: 100, 500, 1000
- **Scheduler Type**: Linear, Cosine
- **Model Type**: Unconditional D3PM, Conditional D3PM

1.5 Results and Analysis

1.5.1 FID Score Results

Table 1 presents the FID scores for all experimental configurations:

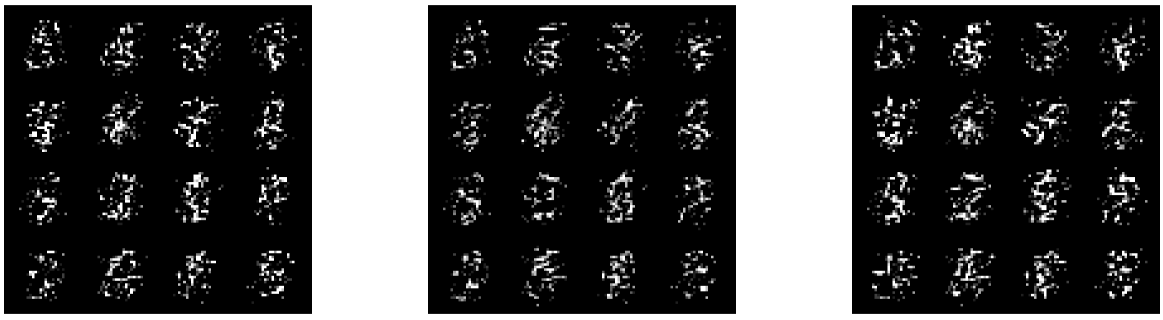
Table 1: FID Scores Across All Experimental Configurations

Model Type	Scheduler	Steps	FID Score
D3PM (Unconditional)	Cosine	1000	198.02
	Cosine	500	183.44
	Cosine	100	195.70
	Linear	1000	180.53
D3PM (Conditional)	Cosine	1000	133.13
	Cosine	500	149.55
	Cosine	100	175.50
	Linear	1000	226.67

1.5.2 Key Observations

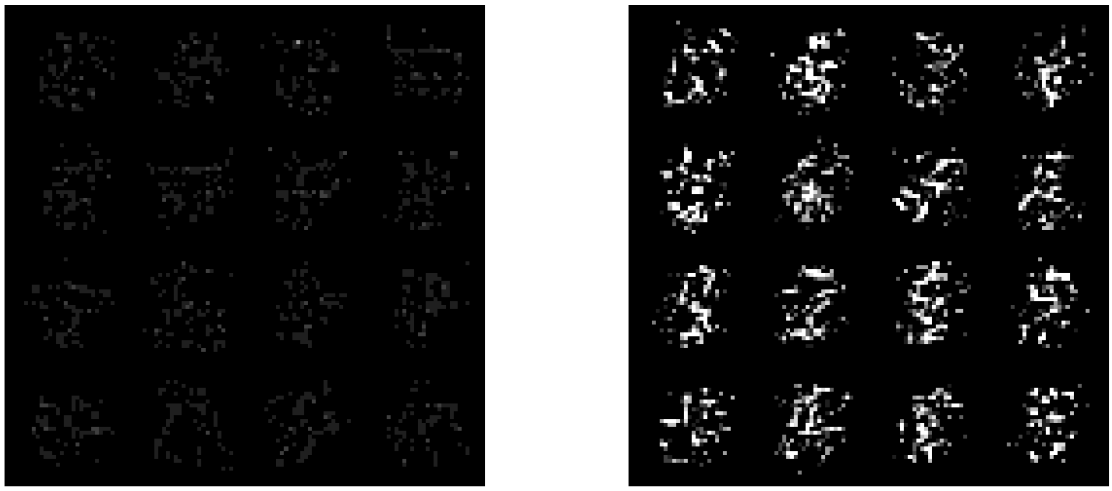
- Conditional models consistently outperform unconditional models across all configurations
- Cosine scheduler is generally superior for both model types but more prominently seen in conditional models
- More diffusion steps generally improve quality for both model types however the improvement is not as significant as the number of steps increases

1.5.3 Sample Visualizations



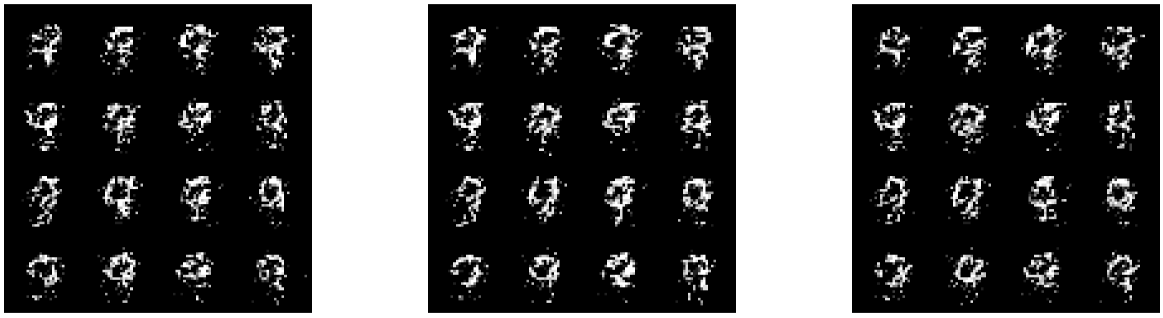
(a) T=100 (b) T=500 (c) T=1000

Figure 1: Final outputs of the unconditional D3PM with cosine schedule for different diffusion steps



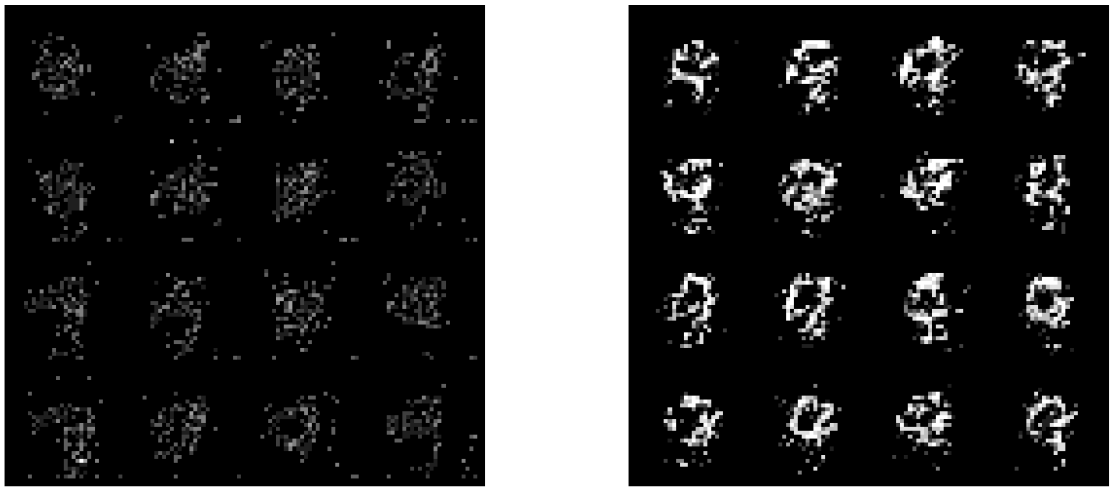
(a) Linear schedule (b) Cosine schedule

Figure 2: Final outputs of the unconditional D3PM with different schedules for 1000 steps



(a) T=100 (b) T=500 (c) T=1000

Figure 3: Final outputs of the conditional D3PM with cosine schedule for different diffusion steps.



(a) Linear schedule

(b) Cosine schedule

Figure 4: Final outputs of the conditional D3PM with different schedules for 1000 steps

2 Continuous Denoising Diffusion Probabilistic Model (DDPM)

2.1 Model Architecture

2.1.1 UNet Backbone

We use the same UNet backbone as the D3PM model described in the previous section.

2.2 Training Procedure

2.2.1 Data Preprocessing

MNIST images are preprocessed as follows:

1. Normalize to [0, 1] range using `transforms.ToTensor()`
2. Scale to model range [-1, 1]: $x_{model} = 2x - 1$
3. Maintain continuous values throughout training

2.2.2 Loss Function

The model uses Mean Squared Error (MSE) loss between predicted and actual noise:

$$\mathcal{L} = \text{MSE}(\epsilon_\theta(x_t, t), \epsilon) \tag{4}$$

where ϵ is the actual noise added and $\epsilon_\theta(x_t, t)$ is the model's prediction.

2.2.3 Noise Scheduling

We use the following noise scheduling methods to add noise to the input images during training as well as sampling:

2.2.4 Linear Schedule

$$\beta_t = \beta_1 + \frac{t}{T-1}(\beta_T - \beta_1) \tag{5}$$

$$\alpha_t = 1 - \beta_t \tag{6}$$

$$\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s \tag{7}$$

where $\beta_1 = 10^{-4}$ and $\beta_T = 2 \times 10^{-2}$.

2.2.5 Cosine Schedule

$$\bar{\alpha}_t = \frac{f(t)}{f(0)} \tag{8}$$

$$f(t) = \cos\left(\frac{\pi}{2} \cdot \frac{t+s}{T+s}\right)^2 \tag{9}$$

where $s = 0.008$ for numerical stability.

2.2.6 Forward Diffusion Process

The forward process gradually adds Gaussian noise:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{10}$$

The reparameterization trick allows sampling x_t directly from x_0:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{11}$$

2.2.7 Training Algorithm

Algorithm 3 DDPM Training Loop

Initialize model θ, optimizer, scheduler
for epoch = 1 to max_epochs **do**
  for batch in dataloader **do**
    Scale data to $[-1, 1]$ range: $x_0 = 2x - 1$
    Sample timesteps $t \sim \text{Uniform}(0, T)$
    Sample noise: $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
    Add noise: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$
    Forward pass: $\hat{\epsilon} = \text{model}(x_t, t)$
    Compute loss: $\mathcal{L} = \text{MSE}(\hat{\epsilon}, \epsilon)$
    Backward pass and update θ
  end for
end for

2.3 Sampling Procedure

2.3.1 Reverse Diffusion Process

Sampling follows the reverse process using the learned model:

Algorithm 4 DDPM Sampling

1: Initialize $x_T \sim \mathcal{N}(0, \mathbf{I})$
2: **for** $t = T - 1$ down to 0 **do**
3: Predict noise: $\hat{\epsilon} = \text{model}(x_t, t)$
4: Compute predicted mean: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \hat{\epsilon} \right)$
5: Sample next x: $x_{t-1} \sim \mathcal{N}(\mu_\theta(x_t, t), \sigma_t^2 \mathbf{I})$
6: **end for**
7: Scale back to image range ($[0, 1]$): $x_0 = \text{clamp}(\frac{x_0+1}{2}, 0, 1)$

2.3.2 Conditional Sampling

For conditional generation, class labels are provided at each timestep:

$$\hat{\epsilon} = \text{model}(x_t, t, \text{class\_label}) \tag{12}$$

2.3.3 Reverse Process Variance

The reverse process variance is computed as:

$$\sigma_t^2 = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \tag{13}$$

2.4 Experimental Setup

2.4.1 Hyperparameters

All experiments use the following base configuration:

- **Epochs**: 300
- **Batch Size**: 64
- **Learning Rate**: 0.0001
- **Optimizer**: Adam
- **Evaluation Samples**: 1000
- **FID Batches**: 5 (64 samples each)

2.4.2 Experimental Variables

We vary the following parameters:

- **Diffusion Steps**: 100, 500, 1000
- **Scheduler Type**: Linear, Cosine
- **Model Type**: Unconditional DDPM, Conditional DDPM

2.5 Implementation Details

2.5.1 Noise Scheduler Implementation

The `NoiseSchedulerDDPM` class handles the following:

- Precomputation of α_t, $\bar{\alpha}_t$, β_t
- Forward process: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$
- Reverse process: $x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z$

2.5.2 Model Architecture Details

Same as the D3PM model described in the previous section.

2.6 Results and Analysis

2.6.1 FID Score Results

Table 2 presents the FID scores for all experimental configurations:

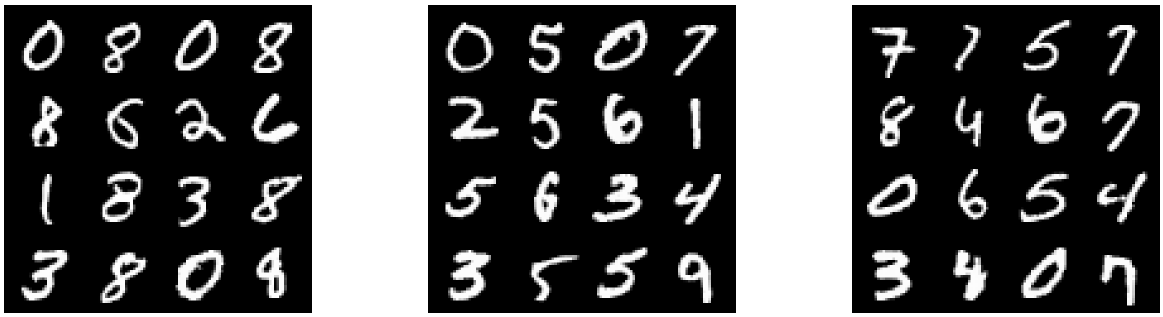
Table 2: FID Scores Across All Experimental Configurations

Model Type	Scheduler	Steps	FID Score
DDPM (Unconditional)	Cosine	1000	52.39
	Cosine	500	52.86
	Cosine	100	53.07
	Linear	1000	139.99
DDPM (Conditional)	Cosine	1000	54.26
	Cosine	500	55.01
	Cosine	100	55.24
	Linear	1000	155.29

2.6.2 Key Observations

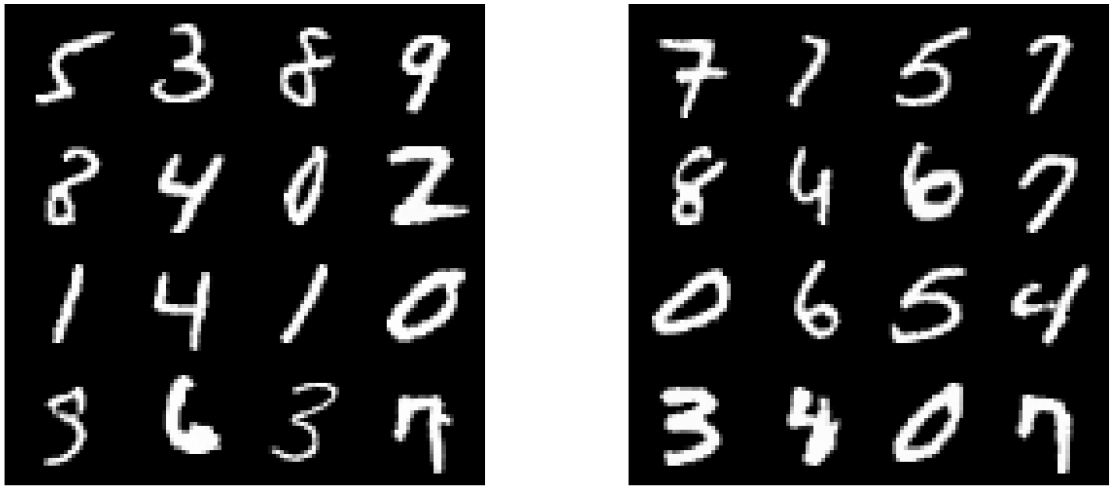
- Cosine scheduler is generally superior for both model types but more prominently seen in conditional models
- More diffusion steps generally improve quality for both model types however the improvement is not as significant as the number of steps increases

2.6.3 Sample Visualizations



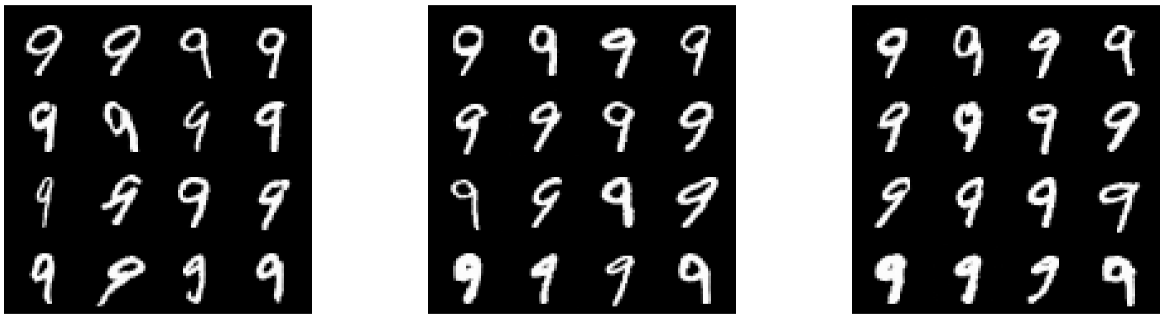
(a) T=100 (b) T=500 (c) T=1000

Figure 5: Final outputs of the unconditional DDPM with cosine schedule for different diffusion steps



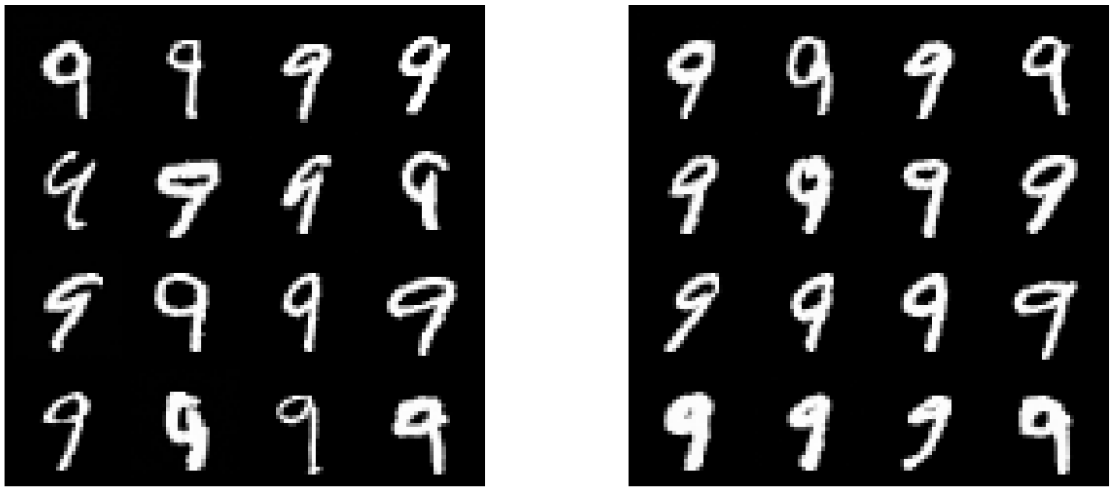
(a) Linear schedule (b) Cosine schedule

Figure 6: Final outputs of the unconditional DDPM with different schedules for 1000 steps



(a) T=100 (b) T=500 (c) T=1000

Figure 7: Final outputs of the conditional DDPM with cosine schedule for different diffusion steps.



(a) Linear schedule (b) Cosine schedule

Figure 8: Final outputs of the conditional DDPM with different schedules for 1000 steps

3 References

- Denoising Diffusion Probabilistic Models - Hugging Face
- Denoising Diffusion PyTorch - GitHub
- Denoising Diffusion Model Implementation from Scratch - Medium
- D3PMs - GitHub
- ChatGPT & Google Gemini