Assignment 2 Report
CS-726: Advanced Machine Learning

Deeptanshu Malu Deevyanshu Malu Neel Rambhia

1 Denoising Diffusion Probabilistic Models

1.1 Architecture

The implemented Denoising Diffusion Probabilistic Model (DDPM) architecture consists of several key components:

1.1.1 Time Embedding

We use sinusoidal position embeddings to encode timestep information:

- The `SinusoidalPositionEmbeddings` module transforms scalar timesteps into 16 dimensional embeddings
- This creates a unique representation for each timestep that preserves the notion of time progression
- The embedding uses a combination of sine and cosine functions at different frequencies, allowing the model to distinguish between timesteps

1.1.2 Network Architecture

The model consists of:

- An input linear layer that maps the concatenation of data and time embeddings to a hidden dimension (128 units)
- A series of 5 `DiffusionBlocks`, each containing a linear layer of 128 units followed by a ReLU activation
- An output linear layer that maps back to the data dimensionality

Rather than using convolutional layers as in image-based diffusion models, we use fully connected layers which is appropriate for dataset without temporal or spatial structure. The time embedding ensures the model can adapt its behavior based on the specific noise level at each timestep.

1.2 Results

For NLL caluclation, the temperature is set to 0.1. For EMD calculation, the number of subsamples is set to 250 and the number of iterations is set to 5.

For all training runs, the hyperparameters used are as follows:

- **Epochs**: 100

- **Batch Size**: 64
- **Learning Rate**: 1e-3
- **Number of Samples**: 5000

1.2.1 Varying Timesteps

Here, lbeta = 0.0001 and ubeta = 0.02.

Dataset	**Metric**	**Timesteps**					
		10	**50**	**100**	**150**	**200**	**500**
Moons	EMD	39.99	28.95	**27.40**	29.64	30.55	44.90
	NLL	1.02	0.96	0.94	**0.93**	0.94	0.95
Circles	EMD	34.60	**31.48**	33.24	34.41	38.62	42.04
	NLL	1.05	0.99	1.00	**0.98**	1.01	1.03
Blobs	EMD	88.78	43.69	20.08	18.36	**17.17**	19.83
	NLL	0.34	0.12	0.04	0.03	**0.01**	0.04
Manycircles	EMD	33.65	**26.41**	27.77	27.98	30.98	30.34
	NLL	0.66	**0.54**	**0.54**	**0.54**	**0.54**	0.58
Helix	EMD	56.00	59.33	57.15	**56.13**	58.87	60.49
	NLL	1.55	1.53	1.52	1.52	1.53	**1.51**

Table 1: EMD and NLL values for DDPM with varying timesteps.

For most datasets, there's a sharp improvement in both EMD and NLL when moving from T=10 to T=50 or T=100. This suggests that too few timesteps lead to poor approximation of the reverse diffusion process. But after reaching an optimal point, further increases in timesteps provide marginal or even negative returns.

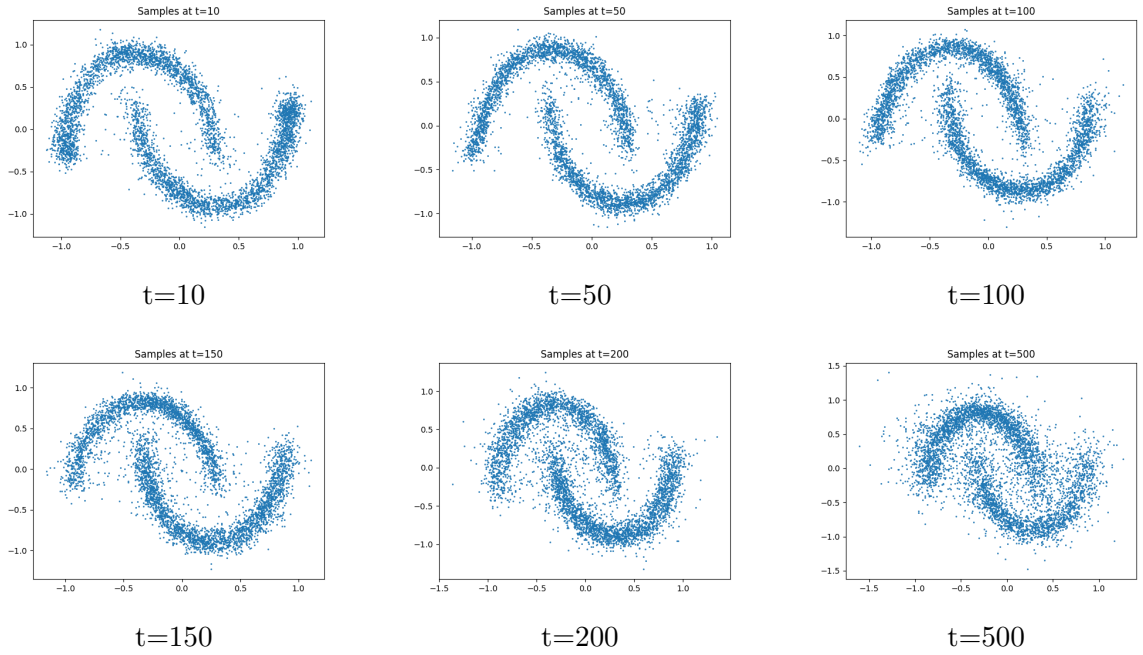
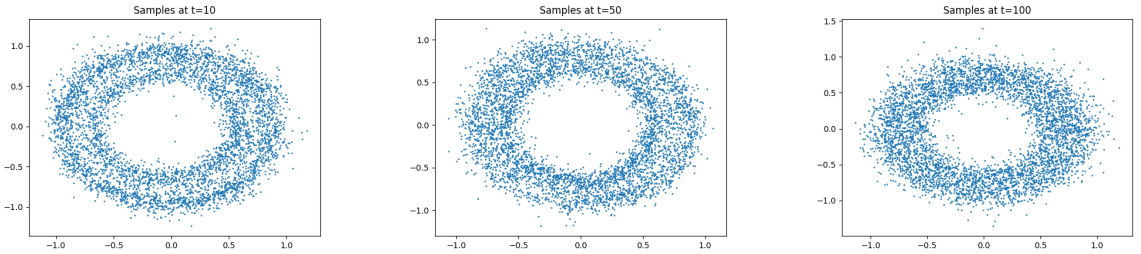
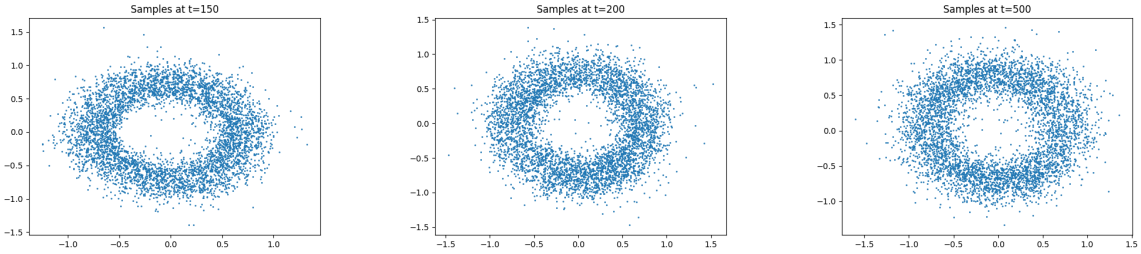


Figure 1: Samples generated by DDPM with **varying timesteps** for the Moons dataset.

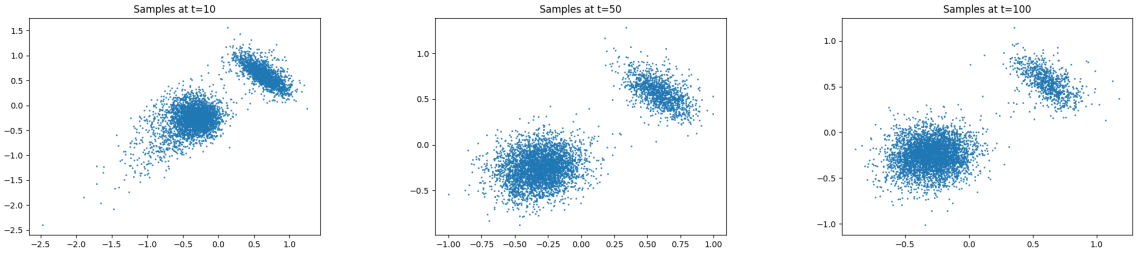


t=10 t=50 t=100

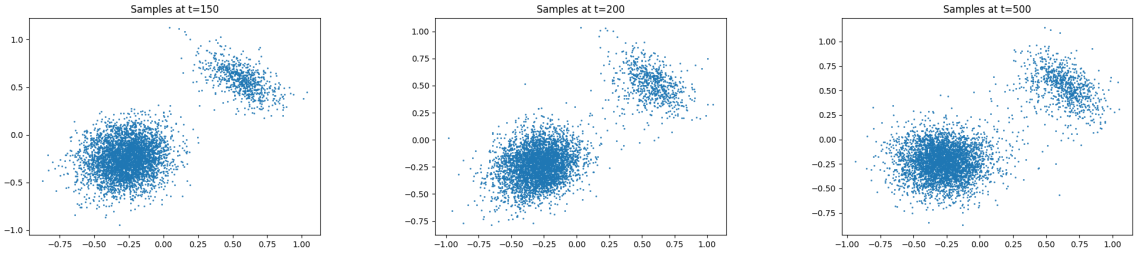


t=150 t=200 t=500

Figure 2: Samples generated by DDPM with **varying timesteps** for the Circles dataset.

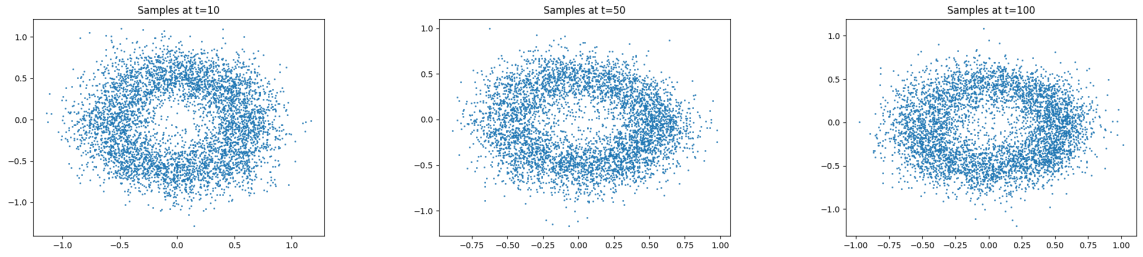


t=10 t=50 t=100

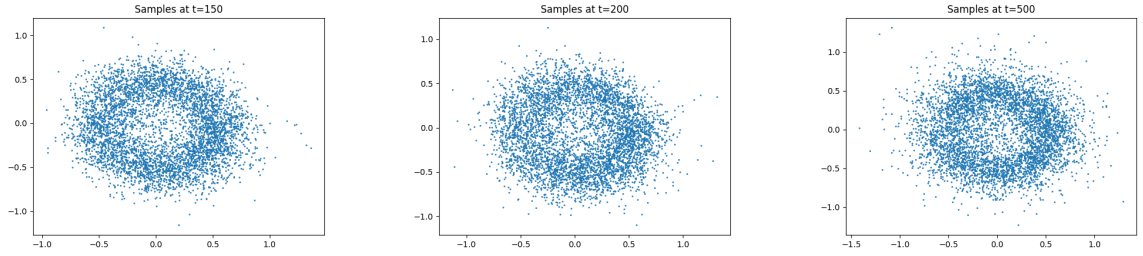


t=150 t=200 t=500

Figure 3: Samples generated by DDPM with **varying timesteps** for the Blobs dataset.

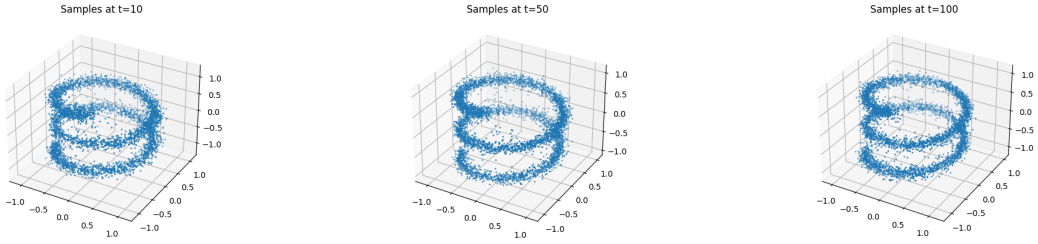


t=10 t=50 t=100

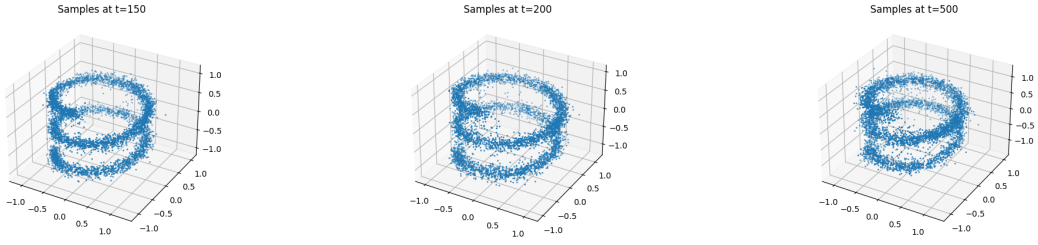


t=150 t=200 t=500

Figure 4: Samples generated by DDPM with **varying timesteps** for the Manycircles dataset.



t=10 t=50 t=100



t=150 t=200 t=500

Figure 5: Samples generated by DDPM with **varying timesteps** for the Helix dataset.

1.2.2 Varying Beta Schedule

Here, timesteps = 200.

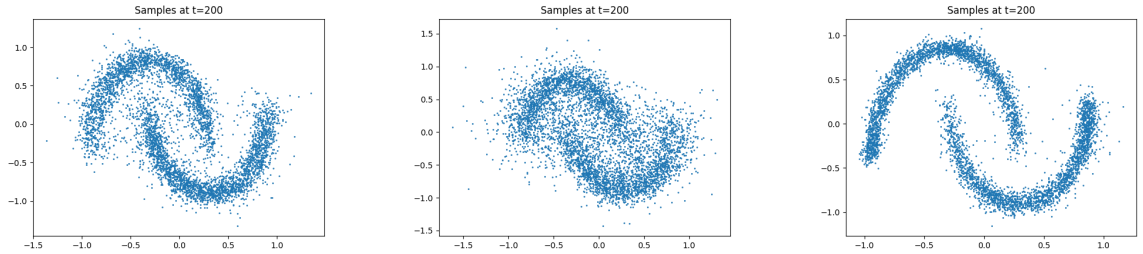
Dataset	**Metric**	**Beta Schedule (lbeta→ubeta)**					
		1e-4→0.02	**1e-3→0.2**	**1e-5→0.002**	**1e-5→0.02**	**1e-4→0.2**	**1e-5→0.2**
Moons	EMD	30.55	35.57	34.76	**30.48**	33.48	33.62
	NLL	0.94	**0.93**	0.98	0.95	0.95	0.94
Circles	EMD	38.62	38.03	**33.03**	38.36	36.52	37.45
	NLL	1.01	**1.00**	1.02	**1.00**	1.01	1.01
Blobs	EMD	**17.17**	20.35	62.98	**17.17**	19.59	19.62
	NLL	0.01	0.02	0.20	0.01	0.01	**0.00**
Manycircles	EMD	30.98	28.75	**26.55**	31.82	29.40	29.92
	NLL	0.54	**0.52**	0.58	0.53	0.53	0.54
Helix	EMD	58.87	**56.25**	57.31	58.29	57.34	58.72
	NLL	1.53	**1.52**	1.54	1.53	1.53	1.52

Table 2: EMD and NLL values for DDPM with varying beta schedules.

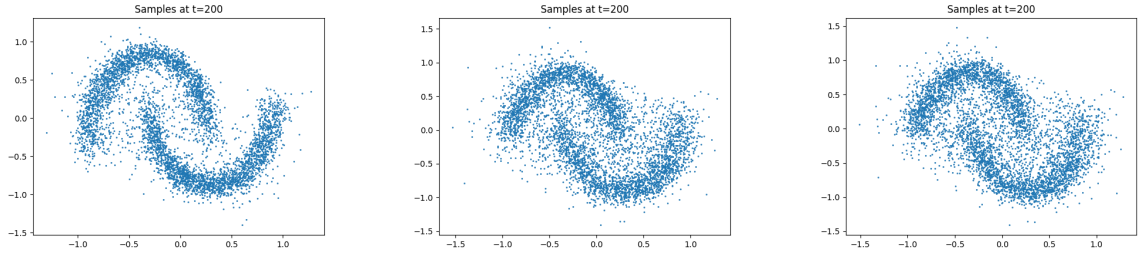
For the beta schedule comparison, there isn't a single schedule that works universally best across all datasets. However, there are some general trends:

- For Moons dataset: 1e-5→0.02 works best for EMD (30.48)
- For Circles dataset: 1e-5→0.002 works best for EMD (33.03)
- For Blobs dataset: 1e-4→0.02 and 1e-5→0.02 work best for EMD (17.17)
- For Manycircles dataset: 1e-5→0.002 works best for EMD (26.55)
- For Helix dataset: 1e-3→0.2 works best for EMD (56.25)

For NLL metrics, slightly higher upper beta values tend to work better, with 1e-3→0.2 achieving the best NLL for several datasets.

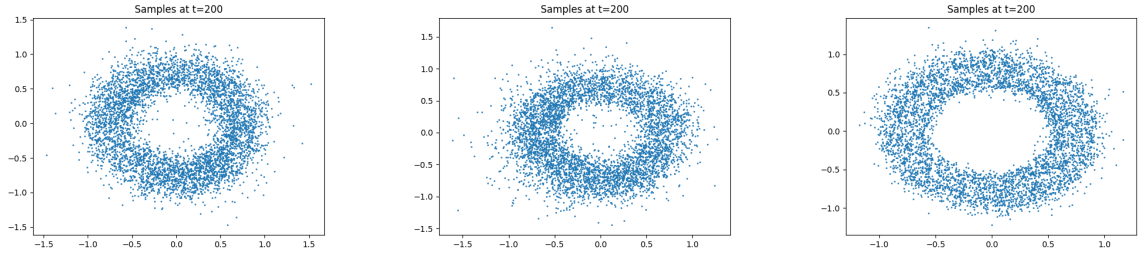


$\beta = \text{1e-4} \rightarrow 0.02$ $\beta = \text{1e-3} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.002$

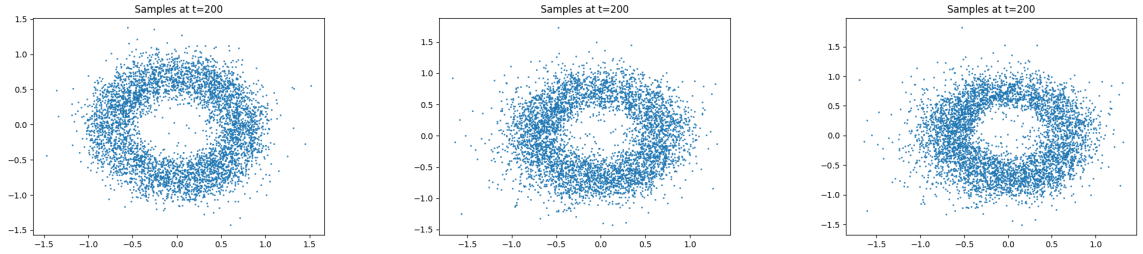


$\beta = \text{1e-5} \rightarrow 0.02$ $\beta = \text{1e-4} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.2$

Figure 6: Samples generated by DDPM with **varying beta schedules** for the Moons dataset.

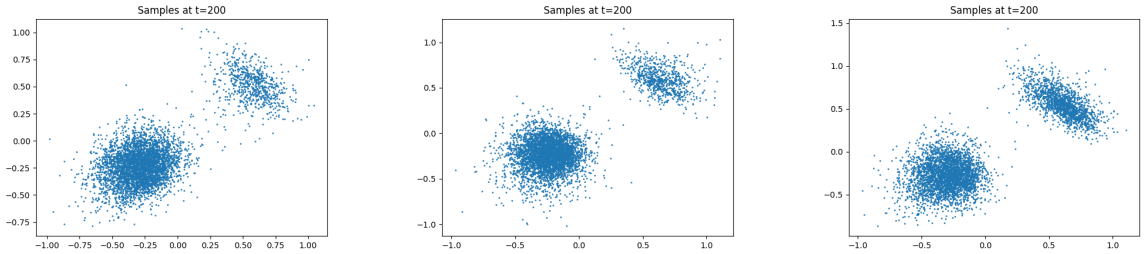


$\beta = \text{1e-4} \rightarrow 0.02$ $\beta = \text{1e-3} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.002$

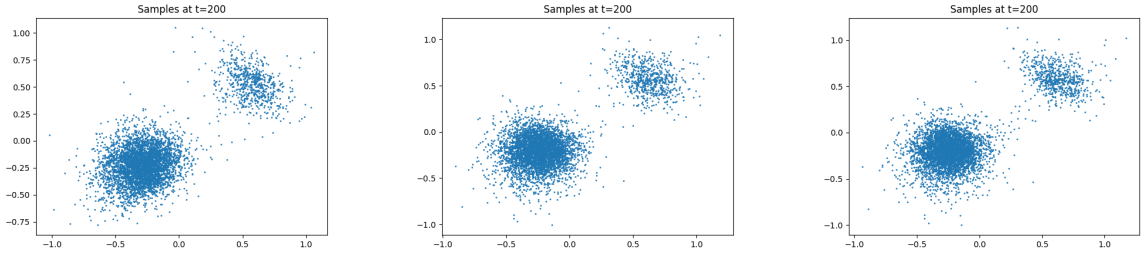


$\beta = \text{1e-5} \rightarrow 0.02$ $\beta = \text{1e-4} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.2$

Figure 7: Samples generated by DDPM with **varying beta schedules** for the Circles dataset.

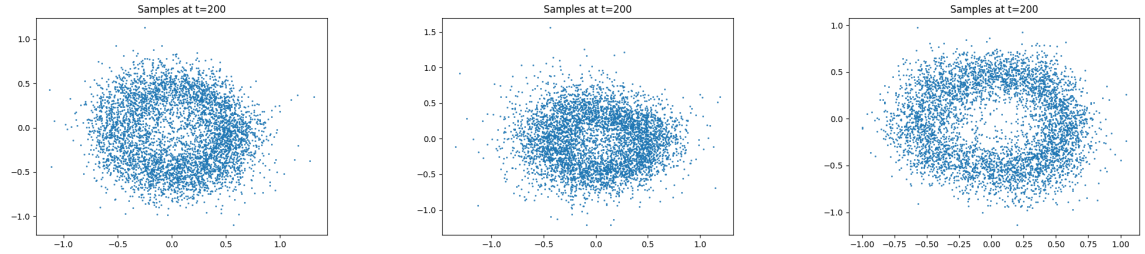


$\beta = \text{1e-4} \rightarrow 0.02$ $\beta = \text{1e-3} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.002$

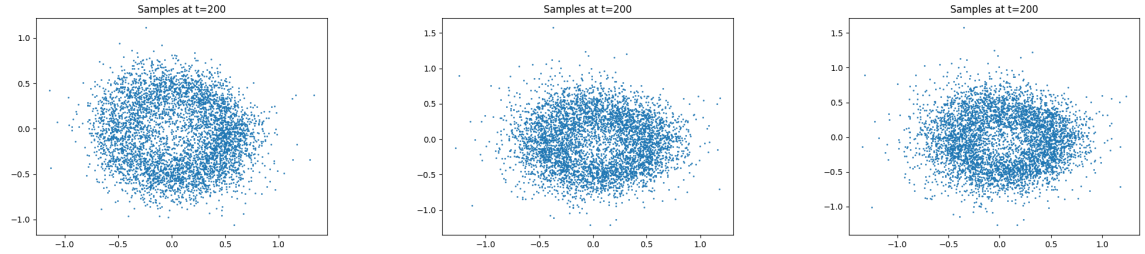


$\beta = \text{1e-5} \rightarrow 0.02$ $\beta = \text{1e-4} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.2$

Figure 8: Samples generated by DDPM with **varying beta schedules** for the Blobs dataset.

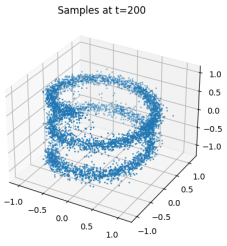


$\beta = \text{1e-4} \rightarrow 0.02$ $\beta = \text{1e-3} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.002$

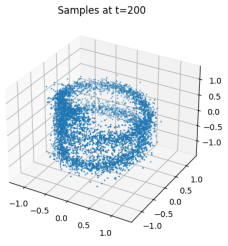


$\beta = \text{1e-5} \rightarrow 0.02$ $\beta = \text{1e-4} \rightarrow 0.2$ $\beta = \text{1e-5} \rightarrow 0.2$

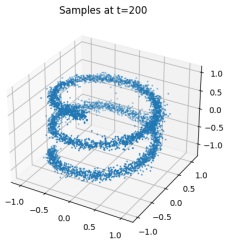
Figure 9: Samples generated by DDPM with **varying beta schedules** for the Manycircles dataset.



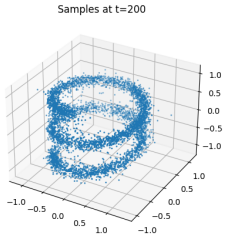
$\beta = \text{1e-4} \rightarrow 0.02$



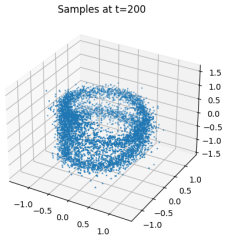
$\beta = \text{1e-3} \rightarrow 0.2$



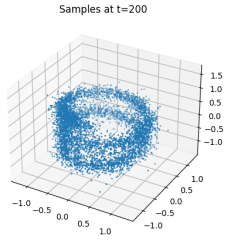
$\beta = \text{1e-5} \rightarrow 0.002$



$\beta = \text{1e-5} \rightarrow 0.02$



$\beta = \text{1e-4} \rightarrow 0.2$



$\beta = \text{1e-5} \rightarrow 0.2$

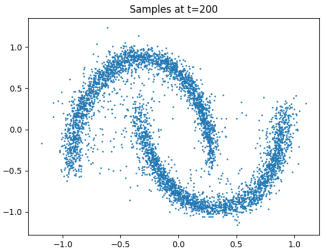
Figure 10: Samples generated by DDPM with **varying beta schedules** for the Helix dataset.

1.2.3 Cosine and Sigmoid schedules

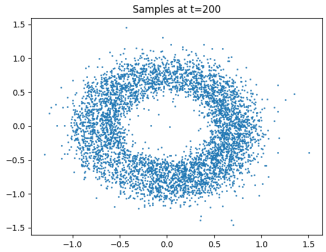
Here, timesteps = 200.

Dataset	Metric	Schedule	
		Cosine	**Sigmoid**
Moons	EMD	**28.29**	30.88
	NLL	0.96	**0.93**
Circles	EMD	**39.77**	41.68
	NLL	**1.01**	1.02
Blobs	EMD	15.54	**15.18**
	NLL	**0.00**	**0.00**
Manycircles	EMD	**30.80**	33.37
	NLL	0.54	**0.53**
Helix	EMD	56.88	**54.38**
	NLL	**1.54**	**1.54**

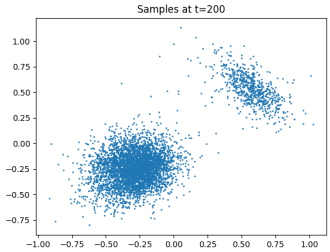
Table 3: EMD and NLL values for DDPM with cosine and sigmoid schedules.



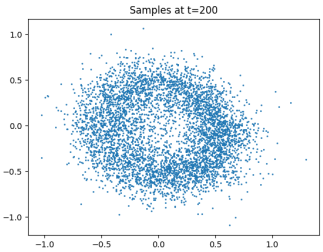
Moons



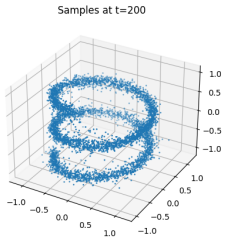
Circles



Blobs

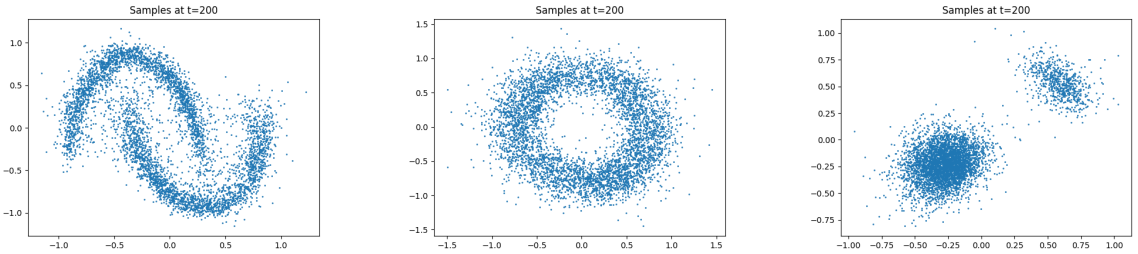


Manycircles

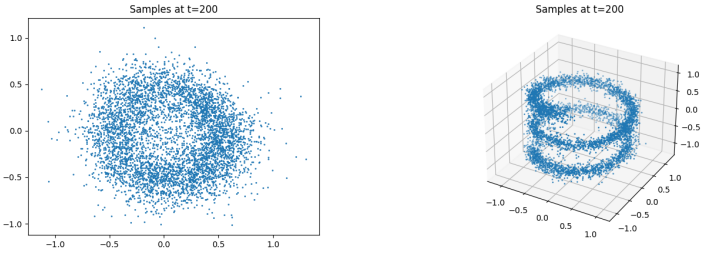


Helix

Figure 11: Samples generated by DDPM with **cosine schedule** for various datasets.



Moons Circles Blobs



Manycircles Helix

Figure 12: Samples generated by DDPM with **sigmoid schedule** for various datasets.

1.2.4 Training Albatross

The hyperparameters used for training Albatross are as follows:

- **Timesteps**: 150
- **Beta Schedule**: 1e-4→0.02

2 Classifier-Free Guidance

For EMD calculation, the number of subsamples is set to 250 and the number of iterations is set to 5.

For all training runs, the hyperparameters used are as follows:

- **Epochs**: 100
- **Batch Size**: 64
- **Learning Rate**: 1e-3
- **Number of Samples**: 20000
- **lbeta**: 0.0001
- **ubeta**: 0.02
- **p_uncond** (required for CFG training): 0.2

2.1 Guided vs. Conditional Sampling

- **Training:** Conditional sampling requires the model to be trained with explicit labels, while guided sampling can work even with an unconditional model.
- **Control:** Conditional sampling directly conditions the model on labels, whereas guided sampling modifies the noise prediction using a guidance scale.
- **Flexibility:** Conditional sampling is fixed after training, while guided sampling allows dynamic tuning of the guidance strength during inference.
- **Diversity:** Conditional sampling maintains diversity in generated samples, but guided sampling can reduce diversity if the guidance scale is too high.
- **Best Use Case:** Conditional sampling is ideal when training a model from scratch, while guided sampling is useful when modifying an already trained model.

2.2 Architecture

The model architecture for Classifier-Free Guidance (CFG) is similar to DDPM, with the modification of concatenating a class embedding (16 dimensional) to the input layer. The class embedding is a sinusoidal position embedding similar to the time embedding used in DDPM. The unconditional training is done by setting the class label to the number of classes.

2.3 Unguided Conditional DDPM Results

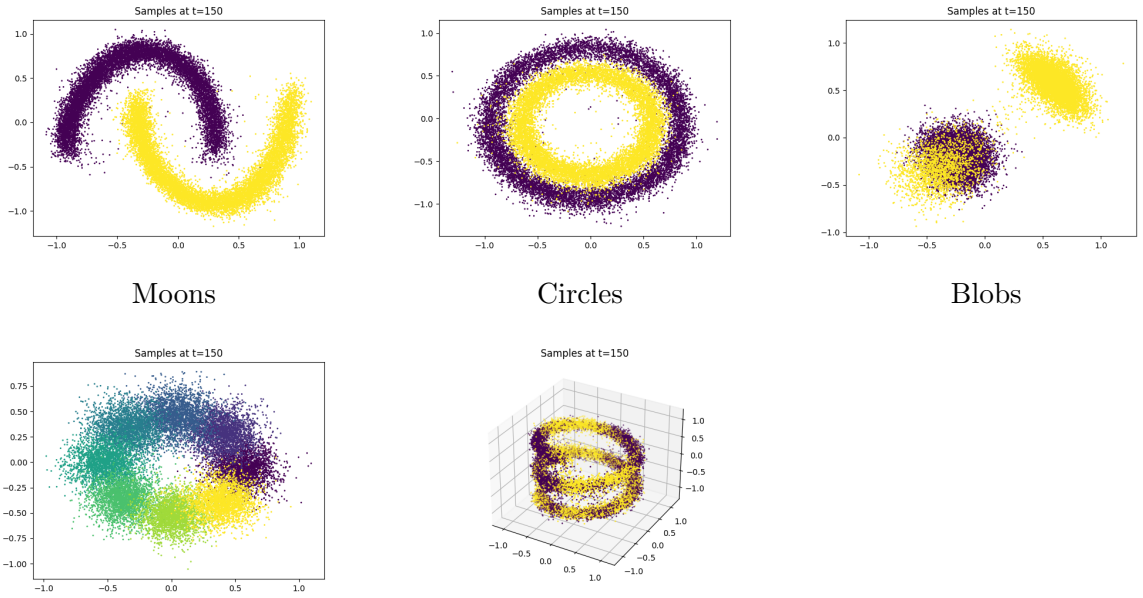


Figure 13: Samples generated by **conditional** DDPM for various datasets.

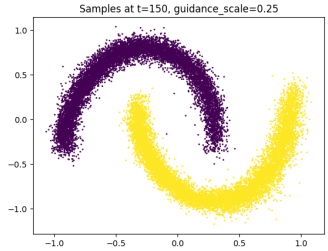
2.4 Classifier-Free Guidance Results

Dataset	Metric	Guidance Scale				
		0.25	**0.5**	**1.0**	**2.0**	**4.0**
Moons	EMD	**27.23**	31.77	44.57	63.59	82.62
	Accuracy	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Circles	EMD	40.60	**40.33**	40.65	42.87	56.30
	Accuracy	0.97	0.99	**1.00**	**1.00**	**1.00**
Blobs	EMD	**20.27**	26.16	39.27	55.48	77.69
	Accuracy	0.95	0.98	0.99	**1.00**	**1.00**
Manycircles	EMD	**12.16**	14.62	18.97	25.14	34.04
	Accuracy	0.92	0.96	0.98	**1.00**	**1.00**
Helix	EMD	**56.82**	58.13	63.17	74.67	97.19
	Accuracy	0.73	0.76	0.82	0.88	**0.94**

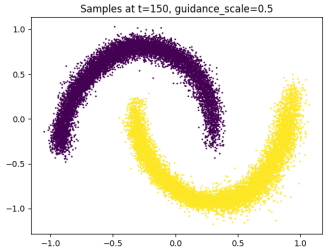
Table 4: EMD and accuracy values for CFG with varying guidance scales.

Earth Mover's Distance (EMD) is used because it measures the minimum cost to transform one distribution into another, accounting for geometric relationships between points. EMD evaluates the entire distribution, effectively capturing how well generated samples match the true data.

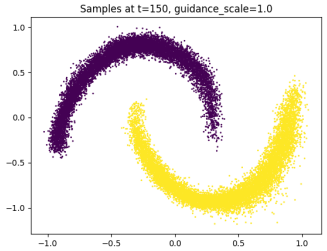
We observe a clear trade-off between sample quality (measured by EMD) and conditional accuracy across all datasets. As the guidance scale increases, the accuracy generally improves. However, this comes at the cost of increasing EMD values, suggesting that the generated samples may stray further from the true data distribution.



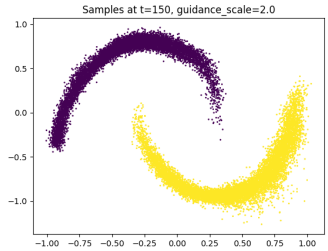
guidance=0.25



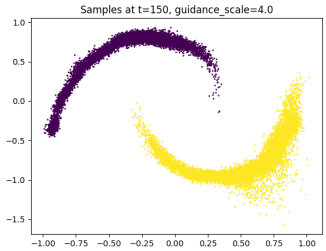
guidance=0.5



guidance=1.0

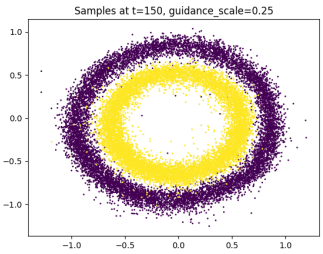


guidance=2.0

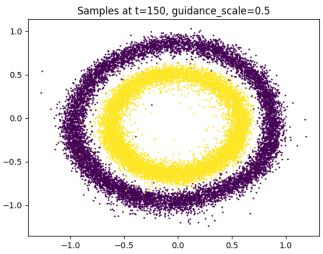


guidance=4.0

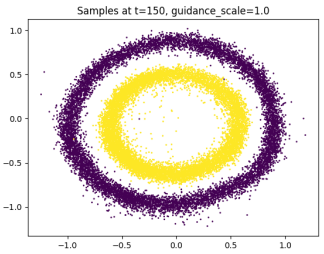
Figure 14: Samples generated by DDPM with **varying guidance scales** for the Moons dataset.



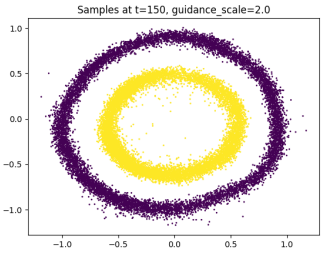
guidance=0.25



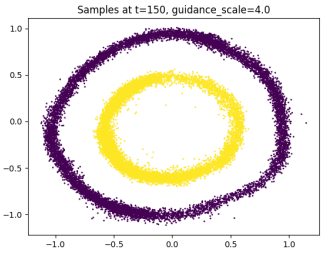
guidance=0.5



guidance=1.0

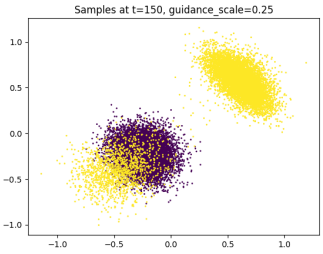


guidance=2.0

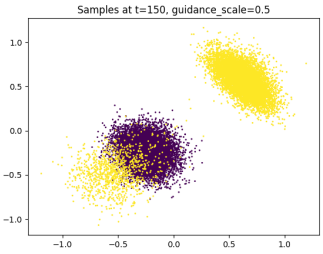


guidance=4.0

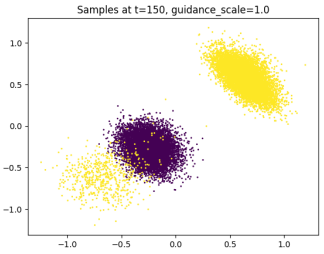
Figure 15: Samples generated by DDPM with **varying guidance scales** for the Circles dataset.



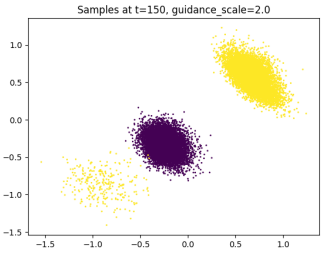
guidance=0.25



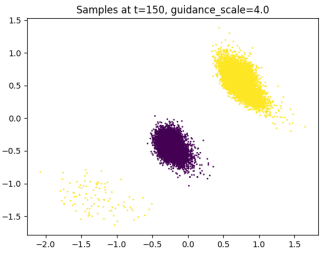
guidance=0.5



guidance=1.0

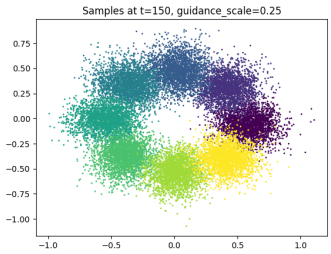


guidance=2.0

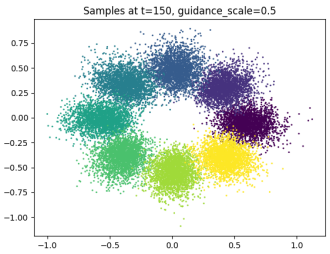


guidance=4.0

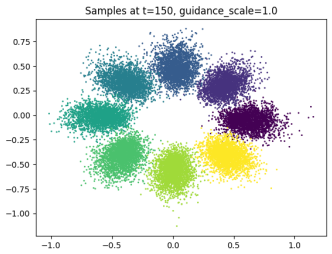
Figure 16: Samples generated by DDPM with **varying guidance scales** for the Blobs dataset.



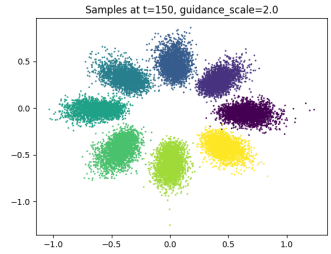
guidance=0.25



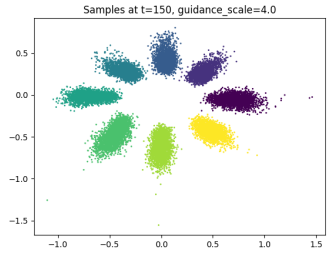
guidance=0.5



guidance=1.0

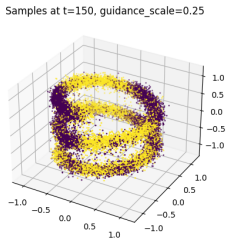


guidance=2.0

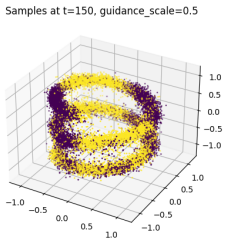


guidance=4.0

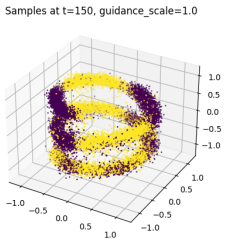
Figure 17: Samples generated by DDPM with **varying guidance scales** for the Manycircles dataset.



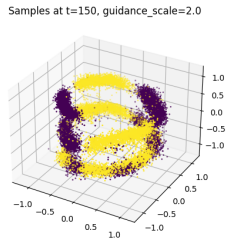
guidance=0.25



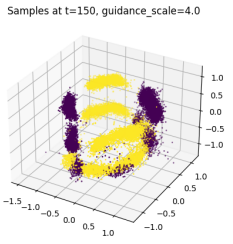
guidance=0.5



guidance=1.0



guidance=2.0



guidance=4.0

Figure 18: Samples generated by DDPM with **varying guidance scales** for the Helix dataset.

2.5 Training Free Classifier

For this we are using the KNN classifier with 100 neighbors. The classifier uses 20000 samples made using the ConditionalDDPM model. We are comparing its performance with an MLP classifier with 2 hidden layers of 100 and 50 units respectively.

Dataset	KNN	MLP
Moons	1.00	1.00
Circles	0.99	1.00
Blobs	0.87	0.86
Manycircles	0.95	0.97
Helix	0.77	0.80

Table 5: Accuracy of KNN and MLP classifiers

3 SVDD-PM

For EMD calculation, the number of subsamples is set to 250 and the number of iterations is set to 5.

The reward scale is set to 0.1. Given a sample the reward function returns the probability of the sample being in a particular class.

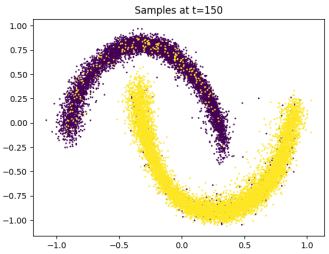
3.1 PM Approximation

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

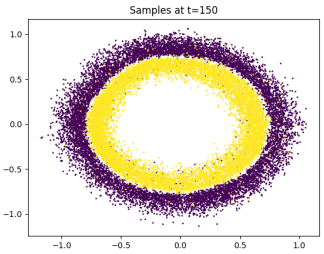
3.2 SVDD-PM Results

Dataset	Metric	SVDD-PM	CFG (guidance=1.0)
Moons	EMD	**36.44**	44.57
	Accuracy	0.96	**1.00**
Circles	EMD	**32.21**	40.65
	Accuracy	0.98	**1.00**
Blobs	EMD	63.07	**39.27**
	Accuracy	0.83	**0.99**
Manycircles	EMD	22.96	**18.97**
	Accuracy	0.89	**0.98**
Helix	EMD	64.24	**63.17**
	Accuracy	**0.93**	0.82

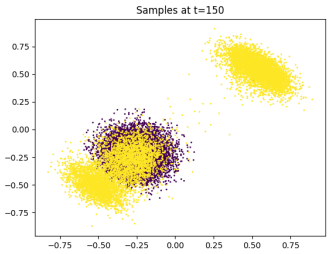
Table 6: EMD and accuracy values for SVDD-PM and CFG with guidance=1.0.



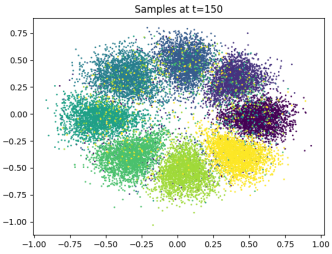
Moons



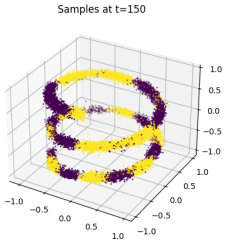
Circles



Blobs



Manycircles



Helix

Figure 19: Samples generated by **SVDD-PM** for various datasets.

Contributions

- **Deeptanshu Malu**:
  1. Ran experiments for varying timesteps for DDPM and analyzed results.
  2. Ran experiments for varying guidance scales for CFG and analyzed results.
  3. Implemented the SVDD-PM model and ran experiments.
- **Deevyanshu Malu**:
  1. Decided on the DDPM architecture and implemented it.
  2. Ran experiments for varying beta schedules for DDPM and analyzed results.
  3. Implemented the SVDD-PM model and ran experiments.
- **Neel Rambhia**:
  1. Decided on the DDPM architecture and implemented it.
  2. Trained MLP classifier and compared its performance with KNN classifier.
  3. Ran experiments for cosine and sigmoid schedules for DDPM and analyzed results.

Acknowledgements

- We took inspiration for the diffusion architecture from here.
- We have also used Copilot for faster coding and not for direct logic.
- We have used ChatGPT for generating a few parts of the code like the KNN classifier.