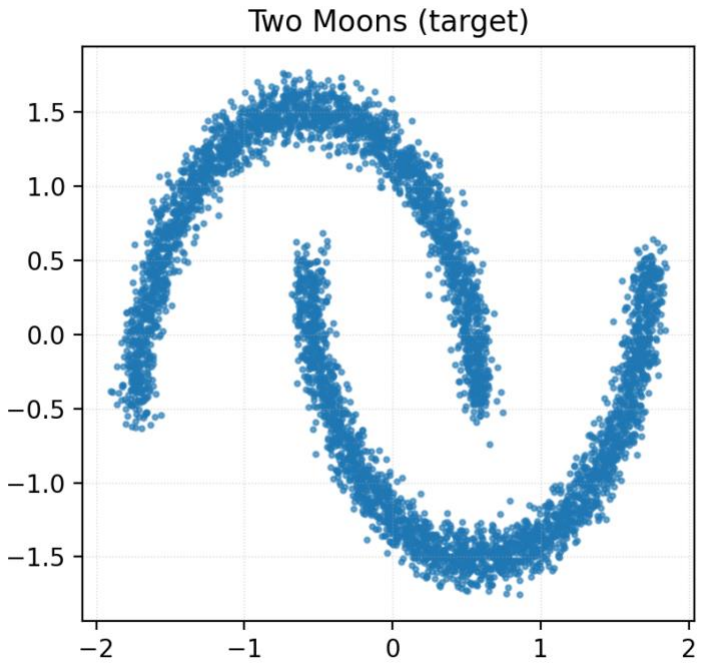
Rishi Desai
Applied ML/Research Engineer take-home assignment.
Code: https://github.com/RishiDesai/fal-task

Goal

Develop a simple test bed for diffusion models and evaluate their results on the two moons distribution.



Evaluation Metrics

The quality of the generated points is evaluated with four metrics:

1. Maximum Mean Discrepancy
2. Energy Distance
3. Wasserstein Distance
4. Classifier Two-Sample Test

Maximum Mean Discrepancy

The MMD compares the kernel-based discrepancy between two distributions, where we use a standard Radial Basis Function (RBF) kernel. I use multiple values for gamma, which allows us to capture differences across local and global structure in the distribution.

$$\mathrm{MMD}(P, Q) = \sqrt{\mathbb{E}[k(X, X')] + \mathbb{E}[k(Y, Y')] - 2\,\mathbb{E}[k(X, Y)]}$$

where $k(u, v) = \exp(-\gamma \|u - v\|^2)$ is the RBF kernel.

MMD is appropriate because we can use the raw data points rather than relying on features or classifier embeddings like KID, which would be overkill for a simple distribution like two moons.

Energy Distance

The Energy Distance compares two distributions with the pairwise Euclidean distance. It's simpler than MMD because it doesn't require hyperparameters. While MMD is more sensitive to local differences, ED emphasizes differences in spread and scale.

$$\mathrm{ED}(P,Q) = \sqrt{2\,\mathbb{E}\|X-Y\| - \mathbb{E}\|X-X'\| - \mathbb{E}\|Y-Y'\|}$$

Wasserstein Distance

The Wasserstein distance computes how much probability "mass" you'd need to move to turn one distribution into another. Computing it exactly in 2D is expensive, so we compute the sliced Wasserstein-1 distance. We project both samples (generated and ground-truth) onto random 1D directions and average the Wasserstein-1 distance between them. I use L=256 random unit vectors, which is more than enough for the two moons.

$$\mathrm{SW}_1(P,Q) = \mathbb{E}_{\theta\sim\mathbb{S}^{d-1}}\; W_1\big(\pi_\theta P,\; \pi_\theta Q\big)$$

where $\pi_\theta(z) = \theta^\top z$ is the projection.

Classifier Two-Sample Test

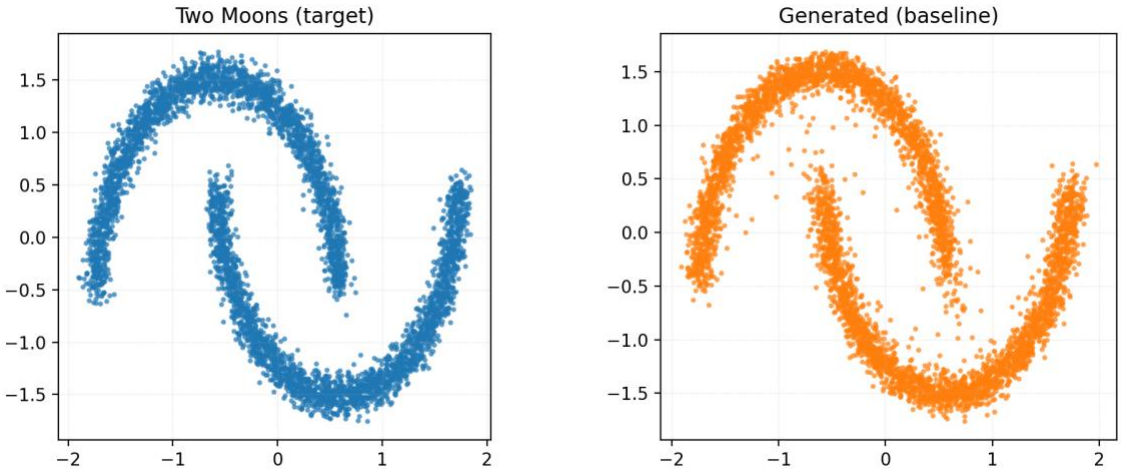
The final metric is training a classifier to predict real vs generated points. I trained a simple logistic regression model with an equal real and generated data split and computed the ROC-AUC and accuracy. If these values are close to .50 then we know that the distributions are very similar. I found that logistic regression was too weak (due to the linear decision boundary) for the two moons distribution, so I swapped it for a support vector machine classifier. The closer the two statistics are to .50 the better. For the ROC-AUC, I report 2 * |ROC-AUC – 0.5| to rescale the value to be between 0 and 1.0, where lower is better. I did not rescale accuracy; 50% accuracy is the target value, which indicates random guessing.

Note that it's difficult to rely on metrics like FID score or Inception score because they rely on pretrained image features or labels, so they aren't particularly useful for simple 2D point distributions.

Baseline Model

I trained a simple score network MLP with 34K parameters with sinusoidal timestep embeddings. The model learns the two moons distribution starting from a standard 2D Gaussian. I used a standard epsilon prediction objective, where the model predicts the added noise with a mean-squared error loss function. I trained the model with the DDPMScheduler and sampled with the EulerDiscreteScheduler from *diffusers*.

The model was trained for 40k steps with the AdamW optimizer, a 1e-3 learning rate, and 512 batch size. I maintain these hyperparameters during the future experiments for a fairer comparison. The next page displays the statistics, showing this simple baseline is very strong, with an MMD < .05 and Energy distance and Wasserstein distances < .06.

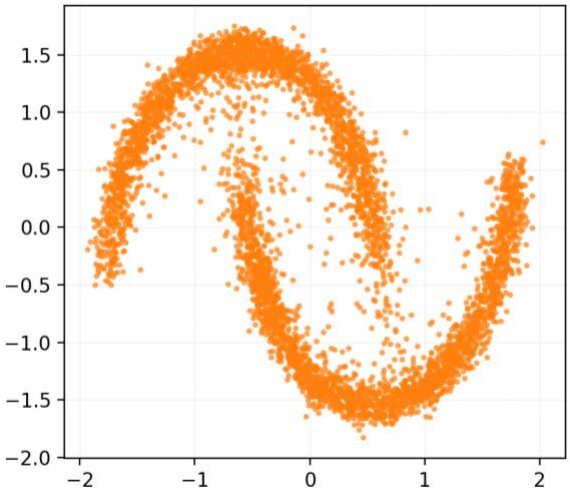


Improvements

I improved upon my simple baseline in a series of six modifications. All the metrics for the various enhancements are listed below for convenience.

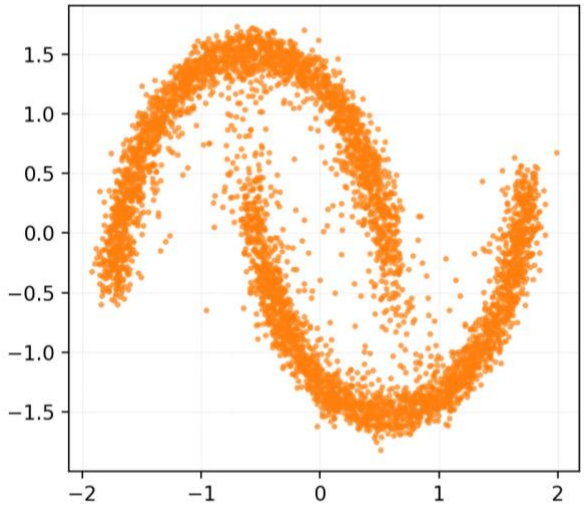
	MMD ↓	ED ↓	WD ↓	C2ST ROC ↓	C2ST Accuracy
Baseline	0.043403	0.059351	0.053859	0.1904	0.4076
1. DDPM	0.036139	0.051655	0.043603	0.2074	0.3968
2. EMA	0.000001	0.020453	0.016328	0.2570	0.3697
3. Cosine betas	0.000001	0.019954	0.015846	0.2606	0.3677
4. Larger MLP	0.000001	0.017666	0.014260	0.2526	0.3667
5.1 SNR weighting (no EMA)	0.034003	0.048948	0.041441	**0.1742**	**0.4200**
5.2 SNR weighting (with EMA)	0.021315	0.034615	0.028897	0.2039	0.3974
6. Residual Blocks	**0.000001**	**0.015938**	**0.012952**	0.2488	0.3623

1. DDPM Scheduler



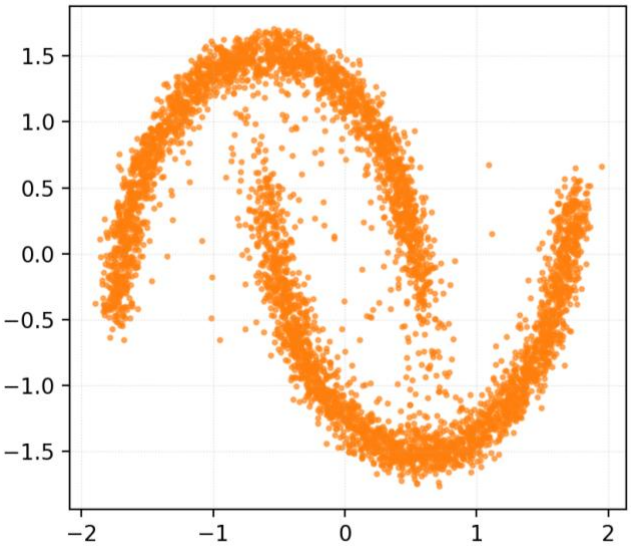
I swapped the Euler sampler for the DDPM sampler, so the training and sampling objectives are consistent. The Euler sampler used an ODE solver, which requires more care to get strong results than when training with DDPM. Overall, this scheduler is more forgiving with step size and score errors, because the noise levels during sampling match what the model saw during training. The DDPM sampler without any hyperparameter tuning easily decreased the MMD, energy and Wasserstein distance by around 20%.

2. Exponential Moving Average



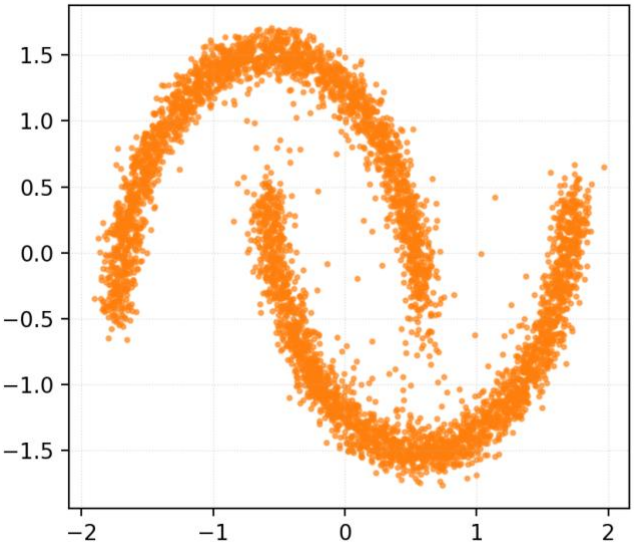
To increase training stability, I added in an EMA on the parameters during training and sampling. I use a conservative value of beta=.999. The EMA lowers the variance of the training at the expense of a small bias. The smaller parameter variance leads to smaller prediction variance. The MMD, ED, and WD all decrease by a quarter from the previous version (DDPM scheduler), which is remarkable. Visually, however, the plot looks indistinguishable.

3. Cosine Schedule



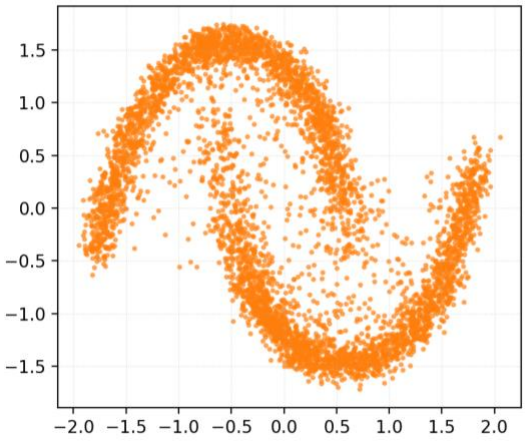
I switched the linear beta schedule to a cosine beta schedule for training and sampling. The linear beta schedule leads to higher variance gradients towards the end of the timesteps due to the more difficult epsilon target. The cosine schedule leads to more stable training and higher quality generation due to the more forgiving noise allocation.

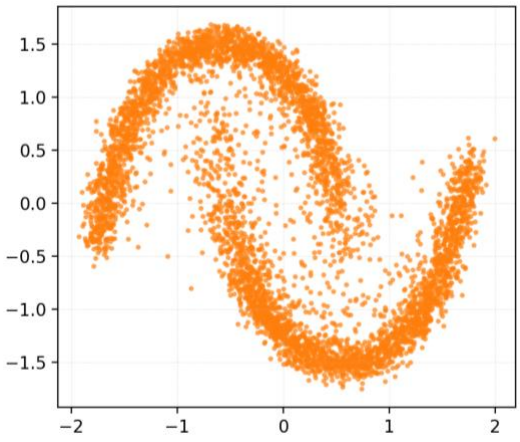
4. Larger MLP Score network



I upgraded the simple MLP with a few more blocks from 34K to 50K parameters. This marginally increased training time and decreased the ED and WD by around 5%. I'm hesitant to continue adding more layers as it's approaching an overengineered network for a simple distribution.

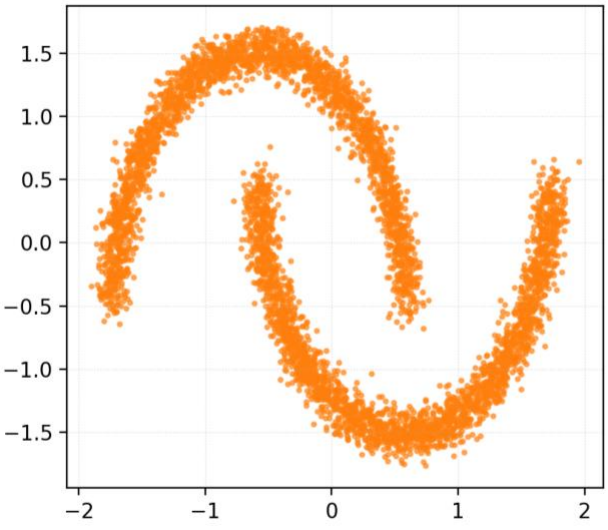
5. **SNR Weighting** (left: without EMA, right: with EMA)





SNR weighting helps balance gradients across timesteps. Even with the cosine beta schedule, the unweighted epsilon loss fluctuates between the easy timesteps and hard timesteps. Weighting the loss by 1/(SNR+1) helps balance the training. There is greater stability as the gradient variance is also more balanced. Notice that the C2ST values improved, whereas the three distance metrics did not. I suspect the pure-noise steps were over-weighted and think this is an example of over-optimization on a simple distribution. Hence, I removed SNR weighting in the final version.

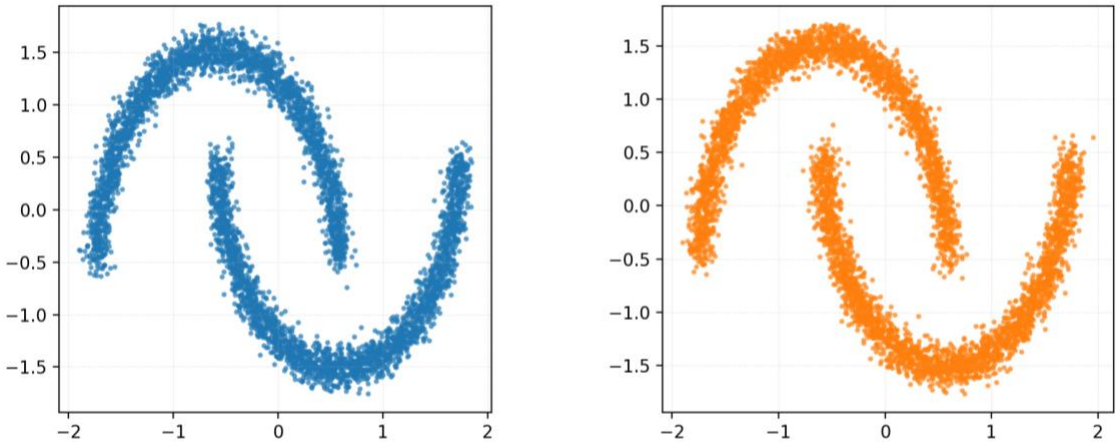
6. **Residual Blocks in Score Network**



I further enhanced the score network to have 134K parameters, where I include 3 residual blocks, each with LayerNorm and Dropout (in a similar vein to transformer blocks). This doubled training time to 7.5 minutes. Seeing the continued improvement in the metrics shows that the model still has some capacity for learning.

Conclusion

After the six enhancements, I found using a DDPM scheduler, cosine beta schedule, EMA, and large score network with residual blocks gave the best results. This image shows the results of the target distribution (blue) and generated points (orange) side-by-side.



	MMD ↓	ED ↓	WD ↓	C2ST ROC ↓	C2ST Accuracy
Baseline	0.043403	0.059351	0.053859	**0.1904**	**0.4076**
Final Version	**0.000001**	**0.015938**	**0.012952**	0.2488	0.3623

This model was trained with the same hyperparameters as the baseline. The MMD is very small (~1e-6) with the RBF settings used, which indicates a larger variety of bandwidth values could have been selected. Nevertheless, the final version improves upon the baseline in all the distance metrics, except for the C2ST metrics. I hypothesize the true distribution has more variance and thicker arcs, compared to the generated points, which are more concentrated. The SVM classifier latches onto noisier arcs of the true distribution, which leads to a slightly lower ROC score and accuracy.

In conclusion, the simplicity of the two moons distribution enables a simple MLP score network and DDPM scheduler to generate high quality points.