Paper Study - Consistency Models

Paul-Henri Pinart
MVA - École polytechnique

Adrien Ramanana Rahary
MVA - École polytechnique

Abstract

Diffusion models, best-in-class generative models for image synthesis, can output high fidelity images but it entails with a trade-off in generation time. Consistency models, explored in this paper study, are a novel class of image generative models strongly inspired from diffusion models which try to address the challenge of generating high fidelity images faster. We first provide a brief recap on diffusion models and introduce the theory behind consistency models. Then, we propose a comparison of a diffusion model, a consistency model distilled from a diffusion model and a consistency model trained from scratch on low dimensional data. Lastly, we explore how consistency models can be used for zero-shot image editing on an inpainting task and compare our results with those obtained with concurrent works on diffusion models.

1 Introduction

1.1 A recap on diffusion models

Diffusion models, first introduced in 2015 [1] and adapted to images in 2020 [2], are generative models which have gained a lot of attention in the recent years because of the high fidelity of the data they can output. In particular, in the context of image synthesis, diffusion models were proven to surpass other image generative models such as GANs which were previously known as the best-in-class models for image synthesis [3].

The core idea behind diffusion models is to learn how to sequentially denoise a Gaussian noise so as to obtain a clear sample resembling training data. Thus, during training, the original data distribution is corrupted with Gaussian perturbations and the the diffusion model is trained to predict the perturbations which were added. In the specific scenario we are interested in, the corruption process follows the stochastic differential equation (SDE) 1:

$$\mathrm{d}\mathbf{x}_t = \sqrt{2t}\,\mathrm{d}\mathbf{w}_t \tag{1}$$

where $t \in [0, T], T > 0$ represents the progress of the diffusion process, $\{\mathbf{x}_t\}_{t\in[0,T]}$ represents the corrupted data at time t and $\{\mathbf{w}_t\}_{t\in[0,T]}$ denotes the standard Brownian motion. This SDE is thus designed such that $p_0(\mathbf{x})$ is the original data distribution and $p_T(\mathbf{x})$ is close to a Gaussian distribution, with $p_t(\mathbf{x})$ denoting the distribution of $\mathbf{x}_t$. This formulation, found in [4] and [5], is referred to as "variance-exploding" in the litterature, and differs from "variance-preserving" formulations proposed in DDPM [2] [6] for instance. [4] prove that the ordinary differential equation (ODE) 2, called the Probability Flow, has solution trajectories which evaluated at time t follow $p_t(\mathbf{x})$:

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\,\nabla \log p_t(\mathbf{x}_t) \tag{2}$$

A neural network is trained to evaluate $\nabla \log p_t(\mathbf{x}_t)$, enabling the resolution of the ODE 2 using a numerical solver. At inference, the ODE is solved by starting from a pure Gaussian noise.

1.2 Consistency models

Sampling from diffusion models requires several numerical solver steps and can therefore take some time compared to single-step generation methods like GANs. Consistency models, proposed in [7], are a novel kind of generative models strongly inspired from diffusion models but with faster sampling.

Core idea. Consistency models are built around the concept of *consistency function*, that aims at mapping pairs of $(\mathbf{x}_t, t)$ which belong to the same probability flow trajectory to the same output x_ϵ ($\epsilon > 0$ is a small time which is used instead of 0 for stability reasons). In other words, a consistency function $\mathbf{f}$ verifies the property of *self-consistency* 3:

$$\forall\, t, t' \in [\epsilon, T]\ \ \mathbf{f}(\mathbf{x}_t, t) = \mathbf{f}(\mathbf{x}_{t'}, t') = \mathbf{x}_\epsilon \tag{3}$$

In practice, consistency models can be parametrized with skip connections to enforce the boundary condition at ϵ (*i.e.* $\mathbf{f}(\cdot, \epsilon)$ is the identity function). More precisely, let $c_{skip}(t)$ and $c_{out}(t)$ be two differentiable functions such that $c_{skip}(\epsilon) = 1$ and $c_{out}(\epsilon) = 0$, let F_θ be a neural network, then the consistency model $\mathbf{f}_\theta$ can be built as in the following equation 4

$$\mathbf{f}_\theta(\mathbf{x}, t) = c_{skip}(t)\mathbf{x} + c_{out}(t)F_\theta(\mathbf{x}, t) \tag{4}$$

Similarly to what can be found in the diffusion models framework proposed in [5], the time horizon $[\epsilon, T]$ of the consistency model is discretized into $N-1$ intervals, the schedule of which can be found in [5] and [7].

Distillation training. Given their close proximity with diffusion models due to the reliance on the probability flow trajectory, consistency models can in fact be distilled from pre-trained diffusion models. The idea behind distillation is to first generate adjacent data points in the time horizon by following the denoising trajectory of a pre-trained diffusion model. Then, the consistency model is trained in order to enforce the self-consistency property on those points. More precisely, a time index n is sampled uniformly from the $N-1$ first indices, a noisy data point $\mathbf{x}_{t_{n+1}}$ is generated by injecting noise into training data up to the time t_{n+1} (*i.e.* $\mathbf{x}_{t_{n+1}} = \mathbf{x} + t_{n+1}\mathbf{z}$ where z is a centered and standard Gaussian noise), and subsequently fed to a pre-trained diffusion model of parameters ϕ to be denoised to the time t_n, thus creating an adjacent point $\hat{\mathbf{x}}^\phi_{t_n}$ on the probability flow trajectory. The consistency model is then trained with respect to a distillation loss $\mathcal{L}_{CD}$ 5 which aims at minimizing a distance d (ℓ_2 distance, ℓ_1 distance, LPIPS, ...) between the model's output for $\mathbf{x}_{t_{n+1}}$ and $\hat{\mathbf{x}}^\phi_{t_n}$:

$$\mathcal{L}_{CD} := \mathbb{E}[d(\ \mathbf{f}_\theta(\mathbf{x}_{t_{n+1}}, t_{n+1})\ ,\ \mathbf{f}_\theta(\hat{\mathbf{x}}^\phi_{t_n}, t_n))] \tag{5}$$

In practice, exponential moving average (EMA) can be leveraged to improve training stability, by introducing an online and a target network.

Consistency training. Consistency models can also be trained as independant generative models. Similarly to distillation from diffusion models, a random time index is sampled from the $N-1$ first indices. Unlike distillation however, the number of discretization time steps is not constant in consistency training: N increases during training, following a specific schedule detailled in the Appendix of [7]. From a same noiseless data point, two noisy data points corresponding to adjacent times in the time-horizon are generated using a centered and standard Gaussian noise z. The consistency model's objective is to minimize the distance between the two outputs obtained when applying the consistency model to these noisy data points, through the loss $\mathcal{L}_{CT}$ defined in equation 6

$$\mathcal{L}_{CT} := \mathbb{E}[d(\ \mathbf{f}_\theta(\mathbf{x} + t_{n+1}\mathbf{z}, t_{n+1})\ ,\ \mathbf{f}_\theta(\mathbf{x} + t_n\mathbf{z}, t_n, t_n))] \tag{6}$$

Similarly to distillation from diffusion models, EMA is used in practice to stabilize training.

Sampling from consistency models The motivation for consistency models lies in their capacity to generate coherent outputs from noisy input in a single step. To do so, a Gaussian noise of variance T^2 is sampled and fed through the consistency model. While this simple approach allows for a single-step and thus fast generation of data, the fidelity is sub-par when compared to techniques with multiple steps.

Thankfully, the authors of [7] propose a multistep sampling algorithm which allows to improve the fidelity of the output data at the cost of more consistency steps. The idea behind multistep sampling is to have data undergo successive rounds of denoising using the consistency model, followed by controlled reintroduction of noise with reduced variance. This cyclic refinement continues until an ultimate denoising with the consistency model.

2 Low dimensional implementation and experiments

To get a better understanding of both diffusion models and consistency models, we have implemented them from scratch and trained them on a low dimensional dataset: two moons. In this section, we provide the details of our implementation, a comparison of the results obtained with a diffusion model, a consistency model trained from distillating a pre-trained diffusion model and a consistency model trained in isolation.

2.1 Implementation details

Backbone and time awareness. When dealing with images, the backbone of diffusion models and consistency models is the U-Net architecture [8]. When dealing with low-dimensional and relatively simple underlying data as in the two moons dataset however, a much simpler architecture can be used. Thus, we implemented a feedforward network with 5 hidden layers and 512 neurons each as the backbone for both the diffusion model and the two consistency models. While we observed that even less complex models could be used (*e.g.* 3 hidden layers of 128 neurons each), this configuration was the one for which we stopped observing significant improvements in fidelity as complexity increased. To account for time-dependency, we use time positional embedding as proposed in [9] and used in diffusion models [2] [5]. More precisely, we compute a time embedding vector with the hidden layer dimension and add it to each hidden representation in the feedforward architecture. This ensures that the model is well aware of time, while also allowing for unseen times to be understood by the model.

Training hyperparameters. For the diffusion model and the consistency model trained from distillation, the time horizon is discretized into 8 intervals ($N = 9$ possible times, as opposed to $N = 18$ in the original paper [7]). Using a finer discretization did not yield better results while increasing sampling time for the diffusion model; and using a coarser discretization reduced the output data quality. For the other time horizon discretization hyperparameters, we use the same values as proposed in [7]. In particular, we follow the same discretization schedule as [7] for training the consistency model in isolation. We train the diffusion model and the distilled consistency model for 10,000 gradient steps with a batch size of 512. The isolated consistency model is trained for 100,000 gradient steps with a batch size of 256. The ADAM optimizer is used with a learing rate of 0.001. While we use EMA on the consistency models following the schedule proposed in [7] (for the sake of self-interest and learning, it was not necessary given the simplicity of the training data...), we do not use it when training the diffusion model. The models are trained using an 8GB NVIDIA GeForce RTX 3070 Laptop GPU.

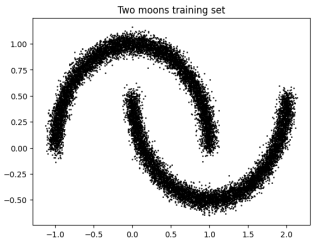
Dataset and evaluation procedure. Training data corresponds to 16384 points sampled from the two moons dataset with a noise level of 0.05. On image synthesis task, the most common metrics used to evaluate the quality of generated data are the Inception Score (IS) or the Fréchet Inception Distance (FID) [10]. Such metrics are used because they allow for a comparison of the training images manifold to the generated images manifold using image features which are assumed to be relevant to the image content, rather than by looking at pixels directly. In the case of the low dimensional data we are interested in, such an approach does not make sense as the actual coordinate values correspond to the position of the data point on the manifold. Therefore, we propose to use the Wasserstein-2 metric W_2 to compare the training distribution and the output distribution. The lower this metric, the better the model is able to sample from the desired distribution. Given the goal of consistency models to generate data faster than diffusion models, we also measure sampling times on a batch of 16384 data points. To have statistically relevant results, we repeat each sampling 100 times.

2.2 Results

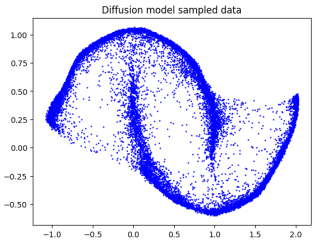
Figure fig. 1 illustrates the training distribution as well as the output distributions from the trained diffusion model, a distilled consistency model (single-step and multisteps) and a consistency model trained in isolation (single-step and multisteps).

Table table 1 provides quantitative results obtained with our experiments.

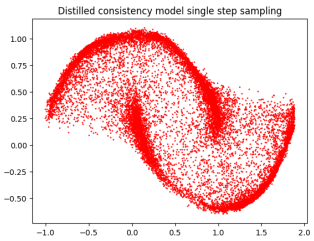
Additionally, we provide the training losses for each model (Appendix, fig. 3) as well as intermediate sampling steps for the diffusion model (Appendix, fig. 4) and the consistency models multisteps sampling (Appendix, fig. 5 and fig. 6).



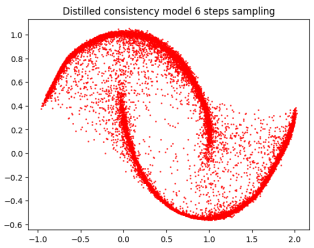
(a) Training distribution (Two moons with 0.05 noise level)



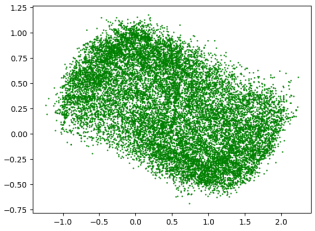
(b) Diffusion model output distribution



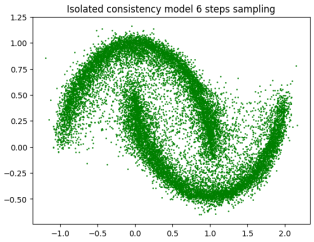
(c) Distilled consistency model (single-step) output distribution



(d) Distilled consistency model (multisteps) output distribution



(e) Isolated consistency model (single-step) output distribution



(f) Isolated consistency model (multisteps) output distribution

Figure 1: Training distribution and output distributions

Model	Sampling time (s)	W_2 metric to training data
Diffusion model (8 ODE steps)	0.109 ± 0.006	0.0330
Distilled consistency model (single-step)	$\mathbf{0.017 \pm 0.003}$	**0.0284**
Isolated consistency model (single-step)	0.017 ± 0.003	0.0514
Distilled consistency model (5 steps)	0.089 ± 0.002	0.1031
Isolated consistency model (5 steps)	0.084 ± 0.004	0.0317

Table 1: Sampling time and data generation fidelity

Our experiments do corroborate the faster sampling of consistency models compared to diffusion models. While, as expected, it is particularly true for single-step sampling (about an order of magnitude faster than sampling from the diffusion model), it is also true for multisteps sampling.

In our experiments, the distilled consistency model sampled in a single-step appears to be the best model both in terms of sampling speed and in numerical output fidelity. While it is natural for the single-step sampling from consistency models to be the fastest strategy out of all the ones we explored, it is quite surprising that it is also the one which yields the highest fidelity. In particular, we could have expected the diffusion model to yield better results than the consistency models.

The visually disappointing outcomes from the single-step isolated consistency training are notable. These results challenge the idea that consistency models, when regarded as a new class of independent generative models, excel as single-step generative models. However, the significantly enhanced results from multi-step sampling suggest that they possess promising capabilities as few-step generative models.

Although the distribution appears visually superior with the multistep distilled consistency model compared to the single-step isolated consistency model, the W_2 metrics indicate otherwise. This raises two possible interpretations: 1) visual confirmation could be deceptive in determining which model produces superior results and/or 2) the W_2 metric may not be the most suitable measure in this context.

The final moons in both the diffusion model and the distilled consistency model appear thinner than those in the training distribution. This observation could suggest two things: 1) there might be an appropriate model architecture or set of hyperparameters that we have not found and thus our results are suboptimal, or 2) these models may struggle to capture a manifold where data points are not evenly weighted (the interior of the moons has a higher density of points than their perimeter). We can also assume that the quality of the diffusion model and its features directly reflect the output of the distilled consistency model.

2.3 Miscellaneous observations

While training these three models, we observed several elements worth noting:

- Training a consistency model in isolation required a lot more time than training the diffusion model or the distilled consistency model. To get results of a comparable quality, the training time for isolated consistency models ($\approx$ 15 minutes) was about an order of magnitude longer than for training the diffusion model ($\approx$ 1 minute) or the distilled consistency model ($\approx$ 2 minutes).
- For both the diffusion model and the consistency models, the training loss seems to be hardly interpretable. In particular, we observed that even though the training loss was stagnating after a certain number of epochs, the corresponding results could greatly vary in quality. For a given training loss value, some models were able to output high fidelity data while others could not.
- Contrary to what we have observed with GANs in practical sessions, diffusion models and consistency models training is extremely stable. In particular, we have not observed mode collapse which compromised the model generation capabilities.

3 Zero-Shot Image Editing and Comparison to DDPM

One of the interesting applications of consistency models is zero-shot image editing. The principle behind zero-shot tasks is to test the ability of a certain model to handle those without being specifically trained for it. There are two main reasons to test these types of models which use a latent space representation. The first is practical : image editing is very useful to enhance photos and correct imperfections for photographers, graphic designers or even for personal use. Another key reason is to assess how well the model captures the underlying structure or distribution of the data. By testing this model on zero-shot image editing, we can evaluate their ability to learn meaningful representations of the input data in the latent space. The task we have decided to investigate is image inpainting.

3.1 Description of the algortihm

Suppose we have an image y and that a part of this image is missing, which we represent by a binary mask Ω. The value of any Ω coefficient is 1 if among the unknown values and 0 otherwise.

The approach behind zero-shot inpainting (see $algorithm$ 1) proposed by the paper requires the knowledge of the binary mask and relies on a multistep sampling procedure. It starts by sampling a noisy version of the image with the first time step t_1 and denoising it with the consistency model f_θ. The resulting latent image variable x then gets progressively refined throughout the algorithm by a sequence of sampling-denoising steps with decreasing noise.

In the **for** loop, the operation $x \leftarrow x \odot \Omega + y \odot (1 - \Omega)$ ensures that the original image information is retained in known areas, while the model's output fills the inpainted regions.

(a) Original LSUN generated image

(b) Partially masked image

Figure 2: Example of an inpainting task where we wish to generate a coherent image from fig. 2b

Algorithm 1: Zero-shot Inpainting

Data: Consistency model f_θ, sequence of time points $t_1 > t_2 > \cdots > t_N$, image y, binary image mask Ω

Result: Output result

$y \leftarrow y \odot (1 - \Omega)$;

Sample $x \sim \mathcal{N}(y, t_1^2 I)$;

$x \leftarrow f_\theta(x, t_1)$;

$x \leftarrow x \odot \Omega + y \odot (1 - \Omega)$;

for *n=2 to N* **do**

  Sample $x \sim \mathcal{N}(x, (t_n^2 - \epsilon^2)I)$;

  $x \leftarrow f_\theta(x, t_n)$;

  $x \leftarrow x \odot \Omega + y \odot (1 - \Omega)$;

end

3.2 Experiments

In order to deepen our understanding of the problem, we tested the inpainting algorithm on the LSUN Bedroom-256 dataset. This dataset consists of a large collection of high-resolution (256x256 pixels) bedroom images and was used for training a consistency model by OpenAI. The checkpoints are available in the original GitHub repository, and we used the weights obtained with LPIPS metric in consistency distillation training. As recommended in the repository, we used Diffusers, an open-source library from Hugging Face which provides tools and pre-trained models for various diffusion-based tasks [11].

We implemented the inpainting algorithm described above and tried different sizes of masks : from 40x40 rectangles to 130x130, which is approximately one fourth of the total surface of the image.

In order to analyse the results, we used several tools. First, we looked at the visual aspect for each timestep of the generated images and their recomposition with the mask (see fig. 7). We usually observe at first that the denoised samples are typical bedrooms but with few common features shared with the original image. As we progress, common features appear such as the type of bed, window, lamp and other key objects or colors. At the end of the iteration, we obtain a coherent generated image, although some pathological examples are provided for large masks

To numerically analyse those results, we also plotted the L2 losses between the denoised sample after mask recomposition and the original image (see fig. 9). We expect this loss to decrease over time, however two things are worth noting. First, we used a model which used the LPIPS as a metric for the loss function. Second, a major L2 loss does not necessarily mean that the generated image is bad. When the mask is relatively small we can expect the generated mask to be very similar to the original one, but when we use large rectangles (such as 130x130), there might not be enough context to determine which elements (lamp, chair, pillow or other decorations) were in the original picture. We removed the loss from the first generated sample and plot different evolutions. There is a general tendancy to decrease but it is not always the case; however, we always observe a convergence which shows that the image is stabilized at a certain point.

A third interpretation of the results consists in computing the Sobel filter (see fig. 8) for the masked image as well as the generated image. The goal is to assess how smooth the borders of the mask are.

We observe that the original rectangle mask is very hard to detect in the generated image, which is positive. However the generated images are not always satisfactory, for example in the 5th and 6th row.

3.3 Comparison to DDPM

In the 6th TP of the course, we discovered and implemented Denoising Diffusion Probabilistic Models (DDPMs) [2]. The approach explored during the practical session (based on [12]) consisted in adapting diffusion models for conditional sampling in reconstruction tasks, in our case inpainting.

The core idea was to modify the standard sampling process by adding a correction term that considers the difference between the predicted image and the actual observation. This correction helped the model generate samples that are more likely to be the true underlying image given the noisy observation. The experiments were done on the FFHQ datasets, which are specifically focused on human faces. However, the complexity of FFHQ and LSUN datasets are very different. The first focuses on faces and their details (highly symmetric, somewhat uniform colours, ...), while LSUN bedrooms contain a large number of objects with the need to understand their relationships and spatial arrangements.

In order to compare both experiments, we compared the runtimes and the associated visual results (see fig. 10). It has a certain coherence in the sense that the images are both of size 256x256, and that one of the key aspects of consistency models is the fast one-step generation. In DDPM experiments, the number of steps can be changed so we tested the results for 5 images with steps from 100 to 600. The runtimes are showed in table 2. For consistency models, we ran 20 samples and obtained an execution time of $16.54s \pm 0.04$. Both were run with GPUs on T4 machines.

Steps	**100**	**200**	**300**	**400**	**500**	**600**
Idx 0	19.43	39.53	59.63	79.31	98.76	119.00
Idx 1	19.74	40.04	59.24	79.54	98.96	118.99
Idx 2	19.65	40.47	59.18	79.51	99.09	119.56
Idx 3	19.63	40.65	59.68	79.55	99.83	122.55
Idx 4	19.65	40.07	59.40	79.72	99.38	119.11
Total	19.62 ± 0.09	40.15 ± 0.35	59.43 ± 0.18	79.53 ± 0.12	98.60 ± 0.28	119.84 ± 1.21

Table 2: Time execution in DDPM in function of the number of steps on T4 GPU

In terms of computational time, consistency models offer a more efficient inpainting method. For consistency models, the results are very often visually satisfying when using a rectangle mask up to a size of 90x90. However, in the last examples where we used 130x130 masks, some examples show that the model is not always able to complete the task properly. For DDPM examples, using 300 steps or less is often unsufficient if we don't use interpolation methods. Therefore, the generation of proper inpainting samples is more time consuming in DDPM.

4 Conclusion

We have presented and compared consistency models, a novel type of generative models, with diffusion models, currently recognized as the top image generative models. In our investigation, we evaluated their ability to produce high-fidelity data in a single step or a few steps, as well as their effectiveness in zero-shot image editing tasks. In our experiments in a low-dimensional setting, consistency models exhibited approximately 10 times faster inference compared to their diffusion model counterparts. Consistency models derived from diffusion models excelled in single-step generation, whereas those trained independently required multiple steps to generate convincing data. In the zero-shot image editing task we examined -— specifically, image inpainting -— consistency models demonstrated the ability to generate convincing data by leveraging multistep sampling. In comparison to concurrent techniques such as Posterior Sampling based on diffusion models, we demonstrated that consistency models were significantly faster without compromising on generation quality.

Diffusion models are notorious for their excellent performance on conditional generation (image-to-image generation, text-to-image generation ...). It would be an interesting challenge to investigate how popular conditioning techniques, like classifier-free guidance [13], can be adapted to consistency models.

References

[1] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[2] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[3] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[4] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2020.

[5] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[6] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models, 2021.

[7] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

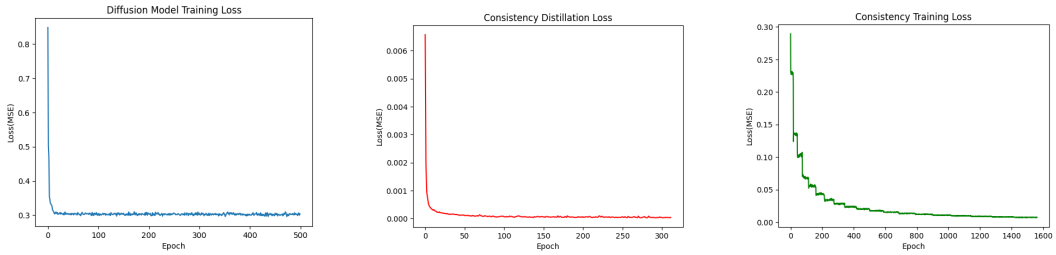
[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. 2017.

[11] Hugging Face. Consistency models. `https://huggingface.co/docs/diffusers/api/pipelines/consistency_models`.

[12] Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

A Low dimensional experiments



(a) Diffusion model (b) Distilled consistency model (c) Isolated conistency model

Figure 3: Training losses on low dimensional data

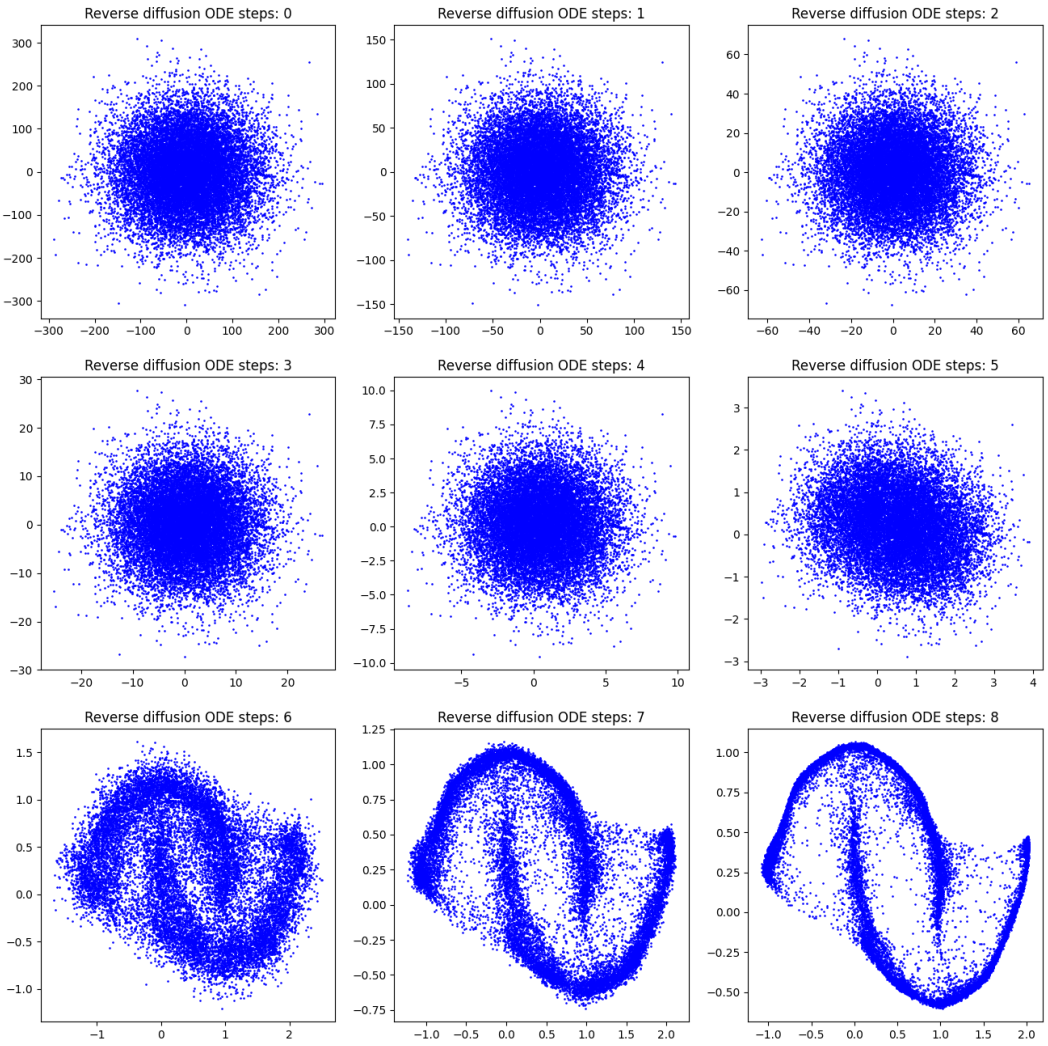


Figure 4: Diffusion sampling trajectory

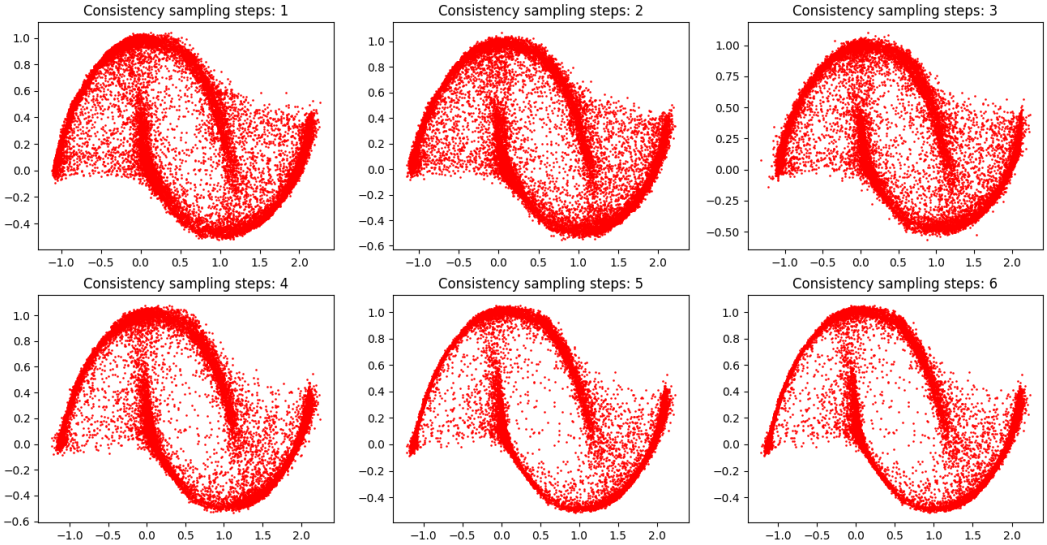


Figure 5: Multistep sampling on a distilled consistency model

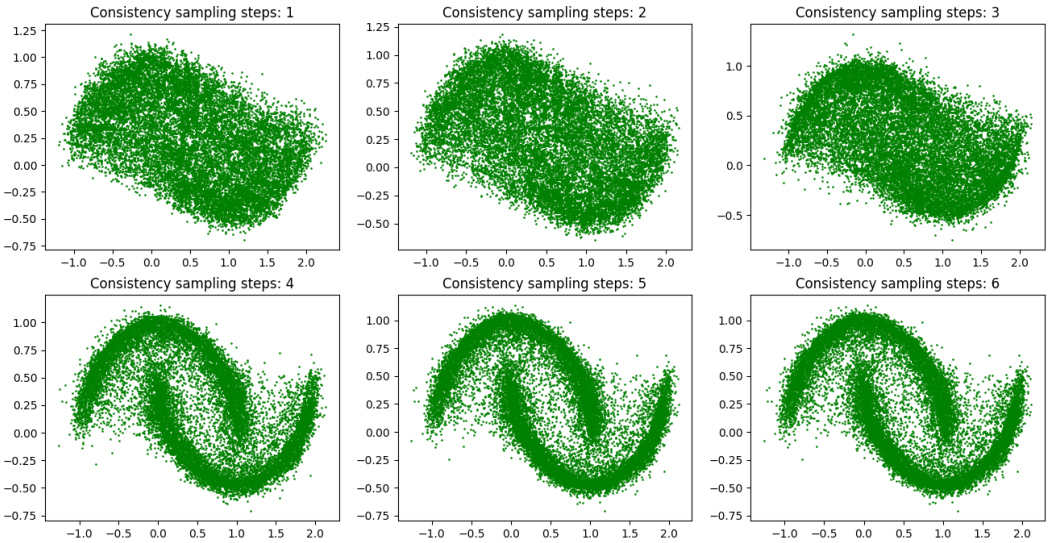


Figure 6: Multistep sampling on an isolated consistency model

B Inpainting examples

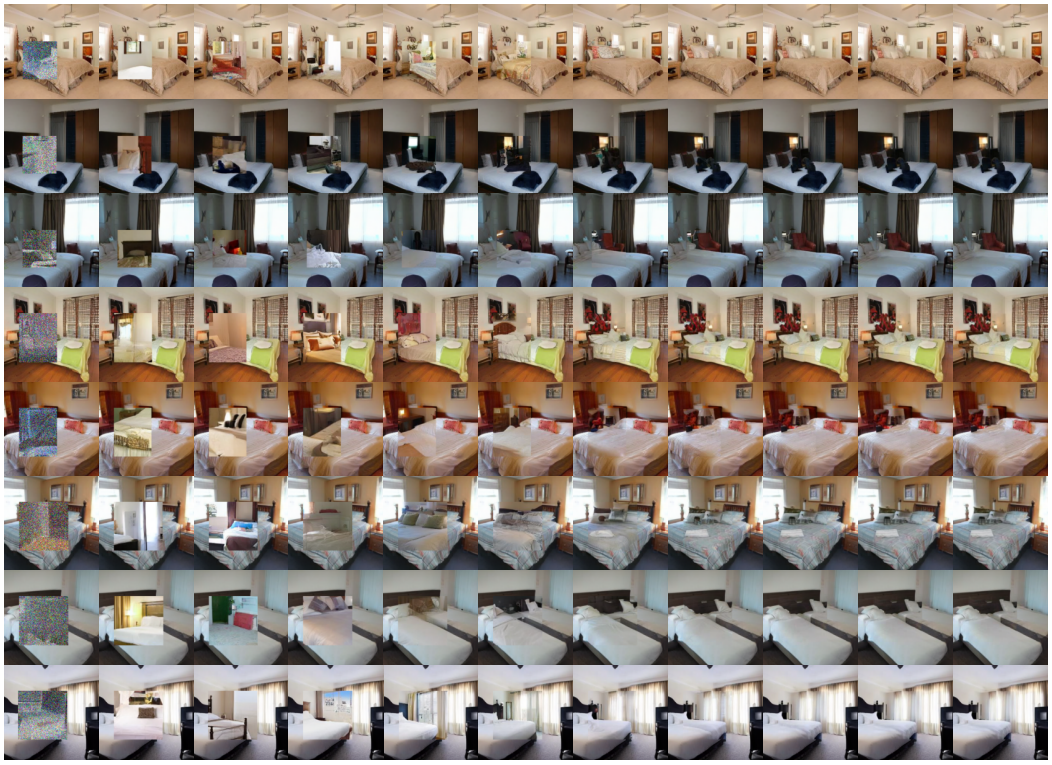


Figure 7: Grid of inpainting examples

C Sobel filters

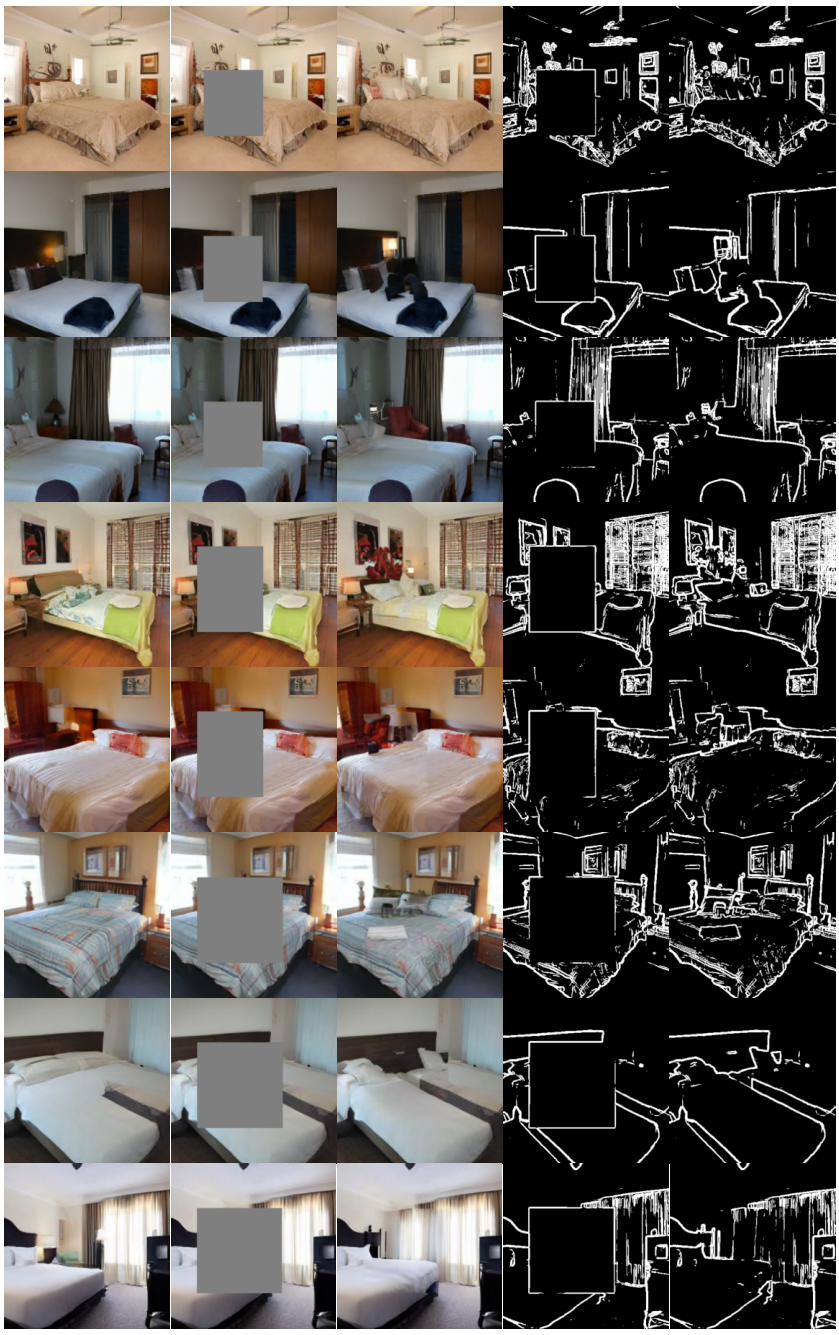


Figure 8: Grid of inpainting results with Sobel filters

D Losses

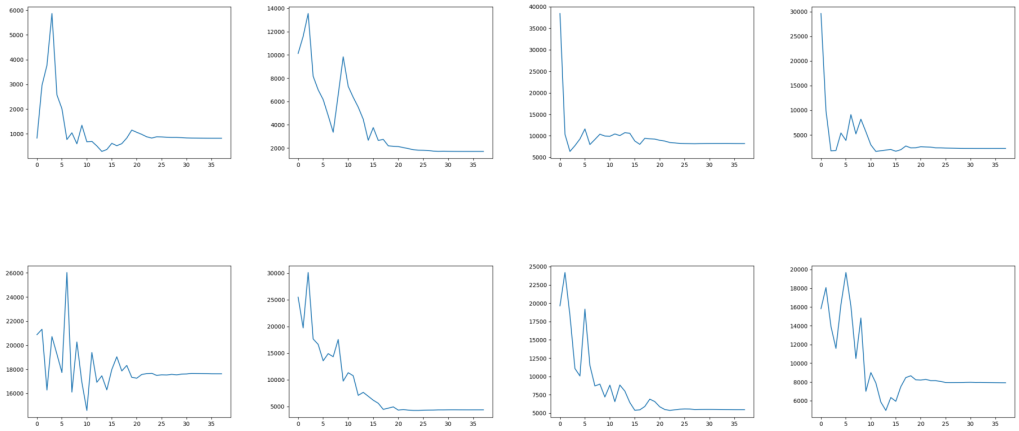


Figure 9: L2 losses evolution

E DDPM

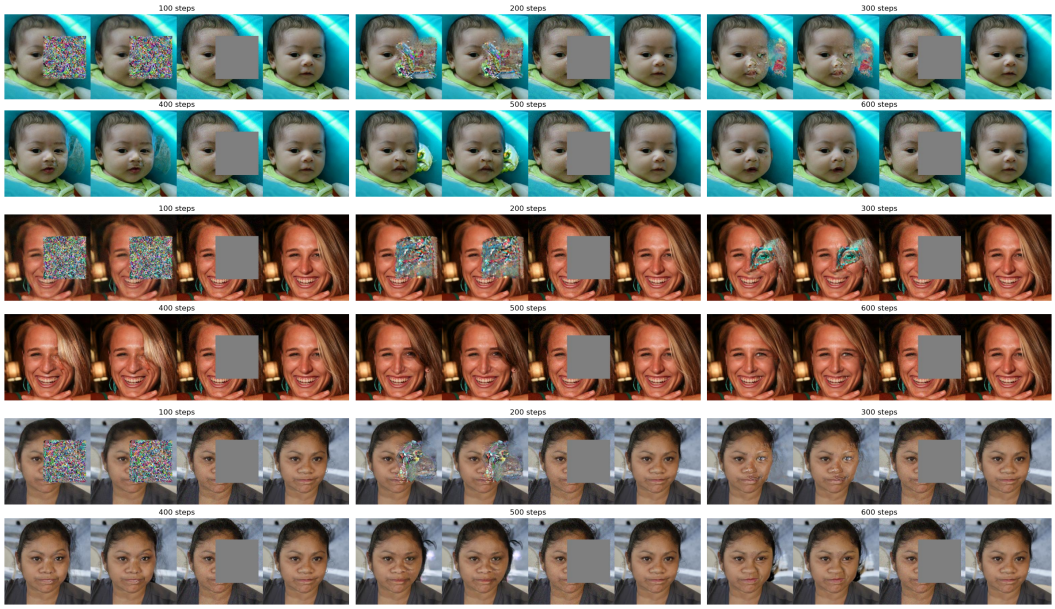


Figure 10: Grid of DDPM examples