Consistency models with learned idempotent boundary conditions

Gianluigi Silvestri [1 2] **Luca Ambrogioni** [2]

Abstract

Consistency Models have recently emerged as an alternative to Diffusion Models, being capable of generating high-quality data while keeping the computational cost low, often requiring only one or two network evaluations. At their core, Consistency Models learn to approximate an *ODE flow*, and their architecture is constrained to respect the boundary conditions of such ODE. In this work, we propose a novel method to train Consistency Models by *learning* the boundary conditions, resulting in a model that acts as an identity only for inputs that are on the support of the data, becoming an *idempotent* function. We compare our method with Consistency Models on simple tabular and image benchmarks, showing competitive sample quality and confirming the potential of the introduced training technique.

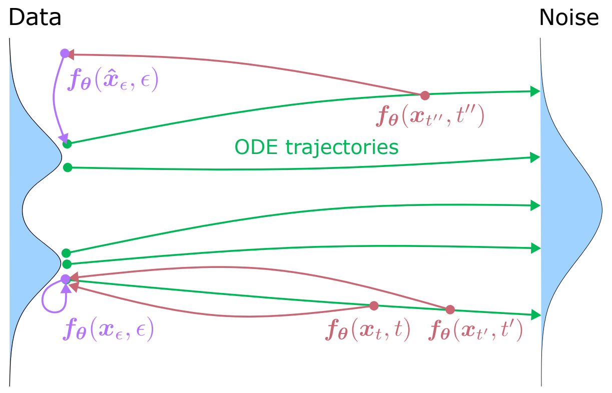

Figure 1. A schematic representation of Idempotent Consistency Models. As in Consistency Models, a neural network is trained to map adjacent points on the same Probability Flow trajectory to the same initial conditions. However, here the boundary conditions are not enforced but are learned with the same network, which results in an idempotent function (purple arrows) that maps points to the support of the data, while acting as the identity for points that are already on the support.

1. Introduction

Generative models synthesize data by learning a mapping between noise and the data distribution. Among various methods, Diffusion Models (DMs) (Ho et al., 2020; Song et al., 2020b) have achieved state-of-the-art performance in several domains, such as images (Dhariwal & Nichol, 2021), video (Ho et al., 2022) and audio (Kong et al., 2020). However, a drawback of DMs is the need for several sampling steps, resulting in a slow and computationally intensive data generation procedure. Since the introduction of DMs, many works have proposed strategies to speed up the sampling process while retaining the same generation capabilities (Song et al., 2020a; Salimans & Ho, 2021; Liu et al., 2021). In particular, faster sampling can be achieved by converting the generative stochastic process into deterministic dynamics described by a system of ordinary differential equations (ODEs) (Song et al., 2020b). In a variance-exploding setting, the system of ODEs is

$$\dot{\boldsymbol{x}}_t = -\sigma^2(t)\ \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\ , \tag{1}$$

where $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ denotes the learned score function and $\sigma^2(t)$ determines the instantaneous noise schedule. Note that the generative dynamics evolve backwards in time, which explains the negative sign in front of the score.

1.1. Consistency models

More recently, Consistency Models (CMs) have been proposed in (Song et al., 2023; Song & Dhariwal, 2023) as an alternative to DMs capable of generating high-quality samples in one or few function evaluations. Instead of using a network to learn the score function (i.e. the vector field of the ODE), the architecture of a CMs learns to approximate the *ODE flow*, which directly maps the noisy state $\boldsymbol{x}_t$ to the end point of the time-reversed generative process:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \boldsymbol{x}_\epsilon\ , \tag{2}$$

where $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, T)$ is a trained approximation of the ODE flow parameterized by $\boldsymbol{\theta}$. In this expression, $\boldsymbol{x}_\epsilon$ is defined

[1]OnePlanet Research Center, imec-the Netherlands; [2]Donders Institute for Brain, Cognition and Behaviour, Radboud University;. Correspondence to: Gianluigi Silvestri <gianluigi.silvestri@imec.nl>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024. Copyright 2024 by the author(s).

as the starting point of the unique ODE trajectory passing through $\boldsymbol{x}_t$ and time t. From the definition, it is easy to see that, when two points $\boldsymbol{x}_{t_1}$ and $\boldsymbol{x}_{t_2}$ are on the same ODE trajectory, the network should respect the following *consistency property*:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_1}, t_1) = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_2}, t_2)\ , \tag{3}$$

which leads to the following (partial) loss function:

$$\mathcal{L}_{ct}(\boldsymbol{\theta}) = d(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{x}_{t_1}, t_1), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_2}, t_2)) \tag{4}$$

where $d(\cdot,\cdot)$ is a divergence function and the minus sign in $\boldsymbol{\theta}^-$ indicates that the gradient is not propagated along this branch of the computation graph. In this form, the loss can only be evaluated when pairs of points on the same ODE trajectory are available, which is the case when distilling an existing DM (Song et al., 2023). However, it is also possible to show that the loss is unbiased when $\boldsymbol{x}_{t_1} = \boldsymbol{x} + \sigma(t_1)\boldsymbol{z}$ and $\boldsymbol{x}_{t_2} = \boldsymbol{x} + \sigma(t_2)\boldsymbol{z}$ at the limit of $t_2 - t_1 \to 0$, where $\boldsymbol{x}$ is a sample from the dataset and $\boldsymbol{z}$ is sampled noise. This observation leads to pure consistency training, where the consistency map is learned without using a pre-trained score function.

The resulting training scheme works by bootstrapping the data $\boldsymbol{x}$ forward in time from the $t = \epsilon$ to $t = T$ along the ODE trajectories, where ϵ is a small boundary time. In order to work, this requires that initial conditions are enforced through the following parameterization:

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, t) = c_{skip}(t)\boldsymbol{x} + c_{out}(t)\boldsymbol{F}_{\boldsymbol{\theta}}(\boldsymbol{x}, t) \tag{5}$$

Where c_{skip} and c_{out} are differentiable functions such that $c_{skip}(\epsilon) = 1$ and $c_{out}(\epsilon) = 0$, and $\boldsymbol{F}_{\boldsymbol{\theta}}$ is an unconstrained neural network.

2. The geometric inconsistency of the boundary condition

In this section, we will show that the boundary constrain given in Eq. 5 is incorrect for any value of ϵ when $\boldsymbol{x}$ lies outside of the support of the data. Consider the simple scenario where the dataset consists of a single data point $\boldsymbol{y}$. In this case, the score $\boldsymbol{s}(\boldsymbol{x})$ is simply $(\boldsymbol{y} - \boldsymbol{x})/\sigma^2(t)$, which diverges for $t \to 0$. Moreover, the ODE flow is given by the constant function $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}, t) = \boldsymbol{y}$, which does not reduce to Eq. 5 for $\epsilon \to 0$ for any $\boldsymbol{x}$ different from $\boldsymbol{y}$. More generally, if the data is supported on a m-dimensional manifold, the score will diverge for any point outside of the manifold, leading to a divergence from the constrained boundary for any point outside of the manifold.

3. Contributions

In this paper, we relax the boundary conditions by replacing the architectural boundary constrains with additional loss terms inspired by the principle of idempotency (Shocher et al., 2023). A function $\boldsymbol{g}(\cdot)$ is said to be idempotent when $\boldsymbol{g}(\boldsymbol{g}(\boldsymbol{x})) = \boldsymbol{g}(\boldsymbol{x})$. A variation of this property approximately applies to consistency models since $\boldsymbol{f}(\boldsymbol{f}(\boldsymbol{x}, t), \epsilon) \approx \boldsymbol{f}(\boldsymbol{x}, t)$ for small values of ϵ, where $\boldsymbol{f}$ is the true consistency function. This is the case because $\boldsymbol{f}(\boldsymbol{x}, t)$ is by definition in the support of the distribution, and therefore the initial condition in Eq. 5 applies. Following (Shocher et al., 2023), we use this principle as additional loss, together with another loss term that enforces the identity at the boundary.

4. Method

Our method, Idempotent Consistency Models (ICM), consists of training consistency models by learning the boundary conditions (Figure 1), rather than enforcing them as commonly done in CMs. In the following, we refer to $\boldsymbol{f}_{\boldsymbol{\theta}}$ as an unconstrained neural network. First, we can replace the hard-coded boundary condition with a boundary loss

$$\mathcal{L}_{bc}(\boldsymbol{\theta}) = d(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_\epsilon, \epsilon), \boldsymbol{x}_\epsilon) \tag{6}$$

where $\boldsymbol{x}_\epsilon$ is sampled from the dataset and perturbed with small noise (see Algorithm 1). While this loss may at first seem equivalent to the hard-coded condition in Eq. 5, it is only applied in the support of the data and it therefore does not introduce the geometric inconsistencies discussed in section 2.

We supplement this loss with an idempotency loss, which directs the consistency networks towards the support of the data:

$$\mathcal{L}_{id}(\boldsymbol{\theta}) = d(\boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \epsilon), \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)) \tag{7}$$

When $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ correctly maps the noisy state to the support of the data, the idempotency loss will be low as a consequence of the boundary loss. On the other hand, the loss will generally be high outside of the support of the data as in our model the boundary condition is not enforced in this range. By combining all the loss terms, we obtain the following total loss:

$$\mathcal{L} = \lambda_{bc}\mathcal{L}_{bc} + \lambda_{ct}\mathcal{L}_{ct} + \lambda_{id}\mathcal{L}_{id}, \tag{8}$$

where $\lambda_{bc}, \lambda_{ct}, \lambda_{id}$ are scalar scaling factors. The pseudocode for this modified training procedure is reported in Algorithm 1. The algorithm for sampling remains the same as for CMs (Song et al., 2023) and is reported in Algorithm 2.

5. Experiments

We test the performance of ICM against CT on 2D toy datasets as well as on the image datasets: MNIST, Fashion-Mnist, and the more complex Cifar10 and CelebA (32×32).

Algorithm 1 Idempotent Consistency Training

```
Input: dataset $\mathcal{D}$, initial model parameter $\boldsymbol{\theta}$, learning rate $\eta$, step schedule $N(\cdot)$, EMA rate $\mu$, $d(\cdot,\cdot)$, $\lambda(\cdot)$, scalars $\lambda_{bc}, \lambda_{ct}, \lambda_{id}$
$\boldsymbol{\theta}_{EMA} \leftarrow \boldsymbol{\theta}$ and $k \leftarrow 0$
repeat
    Sample $\boldsymbol{x} \sim D$ and $n \sim U[1, N(k) - 1]$
    Sample $\boldsymbol{z} \sim N(0, \boldsymbol{I})$
    $\boldsymbol{x}_\epsilon \leftarrow \boldsymbol{x} + \epsilon \boldsymbol{z}$
    $\boldsymbol{x}_{tn'} \leftarrow \boldsymbol{x} + t_{n+1}\boldsymbol{z}$
    $\boldsymbol{x}_{tn} \leftarrow \boldsymbol{x} + t_n \boldsymbol{z}$
    $\mathcal{L}_{ct}(\boldsymbol{\theta}) \leftarrow \lambda(t_n) d(\boldsymbol{f_\theta}(\boldsymbol{x}_{tn'}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{x}_{tn}, t_n))$
    $\mathcal{L}_{id}(\boldsymbol{\theta}) \leftarrow d(\boldsymbol{f_\theta}(\boldsymbol{x}_{tn'}, t_{n+1}), \boldsymbol{f}_{\boldsymbol{\theta}^-}(\boldsymbol{f_\theta}(\boldsymbol{x}_{tn'}, t_{n+1}), \epsilon))$
    $\mathcal{L}_{bc}(\boldsymbol{\theta}) \leftarrow d(\boldsymbol{f_\theta}(\boldsymbol{x}_\epsilon, \epsilon), \boldsymbol{x}_\epsilon)$
    $\mathcal{L}(\boldsymbol{\theta}) \leftarrow \lambda_{ct}\mathcal{L}_{ct} + \lambda_{id}\mathcal{L}_{id} + \lambda_{bc}\mathcal{L}_{bc}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$
    $\boldsymbol{\theta}_{EMA} \leftarrow$ stopgrad$(\mu \boldsymbol{\theta}_{EMA} + (1-\mu)\boldsymbol{\theta})$
    $k \leftarrow k + 1$
until convergence
```

Algorithm 2 Multistep Consistency Sampling

```
Input: Consistency model $\boldsymbol{f_\theta}$, sequence of time points $\tau_1 > \tau_2 > \cdots > \tau_{N-1}$, initial noise $\hat{\boldsymbol{x}}_T$
$\boldsymbol{x} \leftarrow \boldsymbol{f_\theta}(\hat{\boldsymbol{x}}_T, T)$
for $n = 1$ to $N - 1$ do
    Sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$
    $\hat{\boldsymbol{x}}_{\tau_n} \leftarrow \boldsymbol{x} + \sqrt{\tau_n^2 - \epsilon^2}\boldsymbol{z}$
    $\boldsymbol{x} \leftarrow \boldsymbol{f_\theta}(\hat{\boldsymbol{x}}_{\tau_n}, \tau_n)$
end for
Output: $\boldsymbol{x}$
```

Both the baseline CM and ICM use the same settings as in (Song & Dhariwal, 2023) unless differently specified. In this work, we consider only Consistency Training, i.e. without using a pre-trained score model.

5.1. Toy Data

As a simple benchmark, we choose three 2D datasets, namely Circle, 8Dots and Moons. In all these experiments, $\lambda_{bc} = 1$, while $\lambda_{ct} = 0.1, \lambda_{id} = 0.1$ for Circle and 8Points and $\lambda_{ct} = 0.01, \lambda_{id} = 0.01$ for Moons. We also test ICM with $\lambda_{id} = 0$, to evaluate the effect of the idempotency loss. We report the results in Wasserstein Distance in table 1, and show samples in figure 2. ICM outperform CM, and the results highlight the effectiveness of the idempotency loss term. These simple benchmarks suggest that learning the boundary condition and adding the idempotency loss is more beneficial when the data live in a lower dimensional manifold.

	Circle	8Points	Moons
ICM	**0.00062**	**0.0132**	**0.0082**
ICM ($\lambda_{id} = 0$)	0.00145	0.0284	0.0092
CM	0.0023	0.0309	0.0084

Table 1. Results in Wasserstein Distance (lower is better) for one-step sampling. The best entry is highlighted in bold.

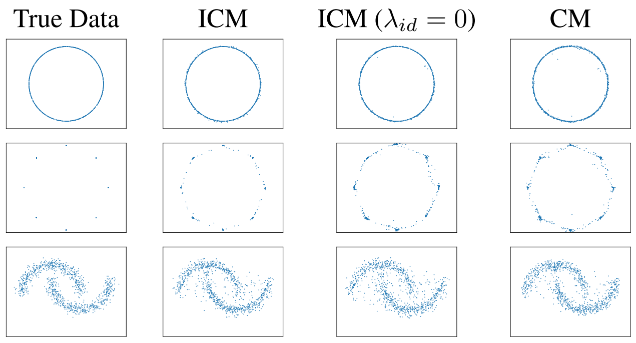

Figure 2. 1-Step samples on the 2D datasets, compared to the ground truth. The datasets are Circle, 8Points and Moons, from top to bottom.

5.2. Image Datasets - MNIST and F-MNIST

To assess the performance of ICM on more complex settings, we train the model on MNIST and Fashion MNIST. For all the ICM models we use $\lambda_{bc} = 1, \lambda_{ct} = 0.1$ and either $\lambda_{id} = 0.1$ or $\lambda_{id} = 0$. We used these values as they provided promising results on the 2D experiments, but we did not tune them to the image datasets. The results in FID score are reported in table 2, while samples from the models are reported in figure 3. ICM outperforms CM on both datasets, and the best performance is achieved when using the idempotency loss.

	MNIST	F-MNIST
ICM	**1.943**	**4.084**
ICM ($\lambda_{id} = 0$)	3.484	5.513
CM	7.823	5.855

Table 2. FID scores (lower is better) for one-step generation. The best entry is highlighted in bold.

5.3. Image Datasets - Cifar10 and CelebA

We additionally train the models on the higher-dimensional image datasets Cifar10 (Krizhevsky, 2009) and CelebA (Liu et al., 2021), using the same loss scaling factors as for MNIST. For CelebA, we preprocess the images by taking the center crop and rescaling them to the 32×32 resolution. For these datasets, we introduce an additional baseline,

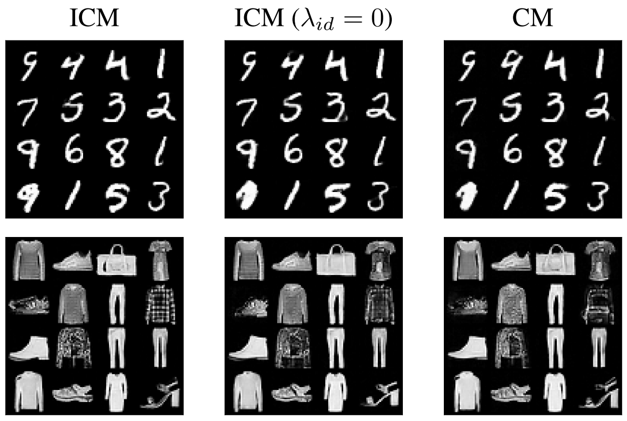

Figure 3. 1-Step samples on the MNIST (top) and Fashion MNIST (bottom) datasets.

where we use standard consistency training but with the simple boundary condition $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_\epsilon, \epsilon) = \boldsymbol{x}_\epsilon$, which enforces the identity only at time $t = \epsilon$ and uses the unconstrained network otherwise. We name this baseline CM-S. The best FID for 1-step generation for all the models is reported in table 3, while the samples from the models are shown in figures 4 and 5. CM outperforms the other models on both datasets, even though by just a small margin on CelebA. Interestingly, ICM with and without idempotency loss perform significantly better than CM-S, confirming the usefulness of learning the boundary condition in this case.

	Cifar10	CelebA
ICM	8.031	2.912
ICM ($\lambda_{id} = 0$)	8.402	3.197
CM	**7.062**	**2.832**
CM-S	12.795	6.467

Table 3. FID scores (lower is better) for one-step generation. The best entry is highlighted in bold.

5.4. Training details

On the 2D experiments, we use a residual MLP with two residual blocks, Positional Embedding for the scale parameter, EMA rate $\mu = 0.9999$ and train all the models for $200k$ iterations with batch size 512. All the datasets consist of $20k$ sampled points. For the MNIST and F-MNIST, we use the NCSN++ architecture as in (Song & Dhariwal, 2023), by changing the following parameters: $model\_channels = 96$, $channel\_mult = [2, 2, 2]$, and $num\_blocks = 2$, batch size 256 and EMA rate $\mu = 0.9999$. For Cifar10 and CelebA, we use the same settings used for Cifar10 in (Song & Dhariwal, 2023), but total batch size 128. We set the dropout rate to 0 for all the experiments, while the remaining hyperparameters and training techniques are the same as in (Song & Dhariwal, 2023).

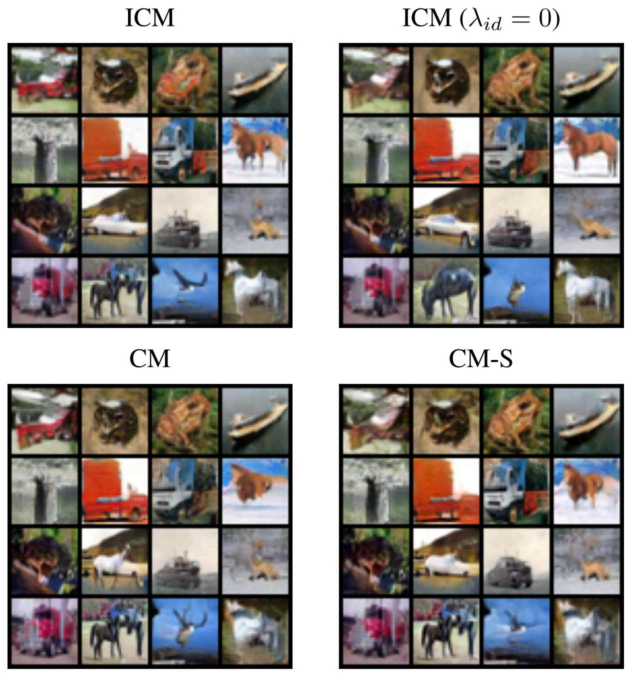

Figure 4. 1-Step samples on the Cifar10 datasets.

6. Related Work

CMs were introduced in (Song et al., 2023) and further improved in (Song & Dhariwal, 2023), and showed to be a valid alternative to DMs in terms of sample quality while requiring significantly less network evaluations. Our work reuses the building blocks from (Song & Dhariwal, 2023) while investigating the benefits of learning the boundary conditions.

The works from (Kim et al., 2023; Heek et al., 2024) focus on improving multi-step sample quality, by mapping points between arbitrary time steps of the ODE flow, or by employing a different training and sampling procedures similar to DDIM (Song et al., 2020a). The contribution from our work is rather focused on improving the performance with one sampling step for CMs, and can be seen as an orthogonal contribution.

A different family of models that share similarities with our method is Idempotent Generative Networks (IGN) (Shocher et al., 2023), where a neural network is trained to be idempotent on manifold, i.e. $F(x) = x, \forall x \in p(x)$. Our boundary and idempotency losses are similar to the losses proposed in IGN, but IGN differs from our method as it maps noise to data in an unstructured way, and it relies on an additional adversarial-style loss to prevent the estimated manifold from

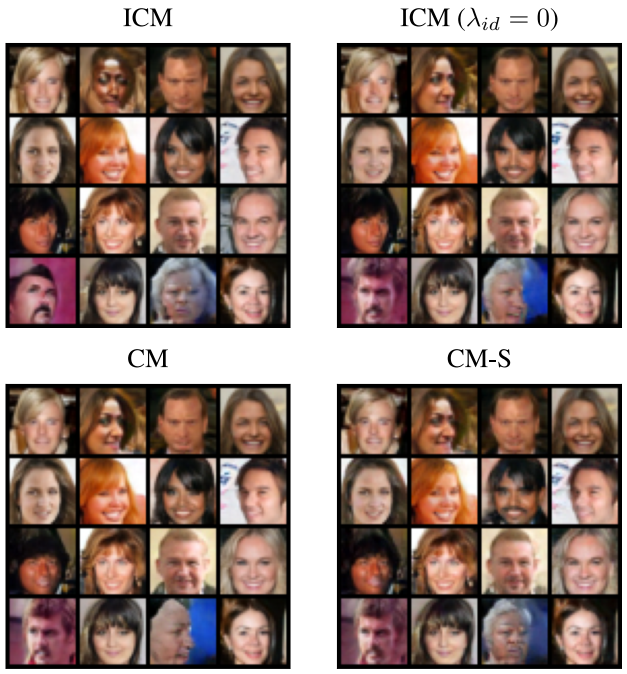

Figure 5. 1-Step samples on the CelebA datasets.

expanding.

7. Conclusion and Future Work

In this work, we introduced ICM, a novel method to train CMs by learning the boundary conditions, which leads to a novel loss formulation that can guide the model output towards the support of the data during training. We demonstrated the effectiveness of ICM on several 2D and image benchmarks, as well as providing a theoretical explanation for the improved performance. To further develop the method, a careful tuning of the loss scales and training procedure can result in improved performances. However, ICM did not outperform standard CM on higher resolution image datasets, and further research is required to verify whether such a model can scale to complex datasets, and to understand in which cases ICM should be the preferred solution.

References

Dhariwal, P. and Nichol, A. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Heek, J., Hoogeboom, E., and Salimans, T. Multistep consistency models. *arXiv preprint arXiv:2403.06807*, 2024.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Kim, D., Lai, C.-H., Liao, W.-H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., and Ermon, S. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *The Twelfth International Conference on Learning Representations*, 2023.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Liu, L., Ren, Y., Lin, Z., and Zhao, Z. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2021.

Salimans, T. and Ho, J. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2021.

Shocher, A., Dravid, A. V., Gandelsman, Y., Mosseri, I., Rubinstein, M., and Efros, A. A. Idempotent generative network. In *The Twelfth International Conference on Learning Representations*, 2023.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2020a.

Song, Y. and Dhariwal, P. Improved techniques for training consistency models. In *The Twelfth International Conference on Learning Representations*, 2023.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020b.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. Consistency models. In *International Conference on Machine Learning*, pp. 32211–32252. PMLR, 2023.